ECNU at SemEval-2016 Task 1: Leveraging Word Embedding from Macro and Micro Views to Boost Performance for Semantic Textual Similarity

Junfeng Tian[1], **Man Lan**[1,2*]
[1]Department of Computer Science and Technology,
East China Normal University, Shanghai, P.R.China
[2]Shanghai Key Laboratory of Multidimensional Information Processing
51151201048@ecnu.cn, mlan@cs.ecnu.edu.cn*

Abstract

This paper presents our submissions for semantic textual similarity task in SemEval 2016. Based on several traditional features (i.e., string-based, corpus-based, machine translation similarity and alignment metrics), we leverage word embedding from macro (i.e., first get representation of sentence, then measure the similarity of sentence pair) and micro views (i.e., measure the similarity of word pairs separately) to boost performance. Due to the various domains of training data and test data, we adopt three different strategies: 1) U-SEVEN: an unsupervised model, which utilizes seven straight-forward metrics; 2) S1-All: using all available datasets; 3) S2: selecting the most similar training sets for each test set. Results on test sets show that the unified supervised model (i.e., S1-All) achieves the best averaged performance with a mean correlation of 75.07%.

1 Introduction

Estimating the degree of semantic similarity between two sentences is the building block of many Natural Language Processing (NLP) applications, such as question answering, textual entailment, text summarization etc. Therefore, Semantic Textual Similarity (STS) has received an increasing amount of attention in recent years, e.g., the STS tasks in Semantic Evaluation Exercises have been held from 2012 to 2016.

To identify semantic similarity of sentence pairs, most existing works adopt at least one of the following feature types: 1) string based similarity (**?**; **?**) which employs common functions to calculate similarities over string sequences extracted from original strings, e.g., lemma, stem, or n-grams sequences; 2) corpus based similarity (**?**) where distributional models such as Latent Semantic Analysis (LSA), are used to derive the distributional vectors of words from a large corpus according to their occurrence patterns, afterwards, similarities of sentence pairs are calculated using these vectors; 3) knowledge based method (**?**) which estimates the similarities with the aid of external resources, such as WordNet. Among them, **?**) leverage different word alignment strategies to bring word-level similarity to sentence-level similarity.

Traditional NLP feature engineering often treat sentence as a bag of words or term frequency, and endeavor to evaluate the similarity according to the co-occurrence of words or other replacement words. For example, **?**) built a supervised model using ensemble of heterogeneous features and achieved great performance on STS Task 2014. However, it is difficult to evaluate semantic relatedness if all the word in both sentences is unique. For example: *A storm will spread snow over Shanghai; The earthquakes have shaken parts of Oklahoma*. These sentences have no words in common, although they convey the similar information.

In this work, we first borrow the aforementioned effective types of similarity measurements including string-based, corpus-based, machine translation similarity and alignment measures to capture the semantic similarity between two sentences. Besides, we also present our highly interpretable and hyper-parameter free word embedding features from

macro and micro views to boost the performance. Then we adopt three different strategies of the usage of training data: 1) U-SEVEN: an unsupervised model, which utilizes seven straight-forward metrics (i.e., longest common sequence, alignment feature, corpus-based feature, and others are all from word embedding features); 2) S1-All: use all available datasets and train a unified regression model after deleting unnecessary features; 3) S2: select the most similar training sets for each test set, according to the source of the dataset, average sentence length, and similarity distance (i.e., word mover's distance, discussed in Section 2.3).

The rest of this paper is organized as follows. Section 2 describes various similarity measurements used in our systems. Section 3 gives the datasets and system setups. Results on training set and test set will show in Section 4 and 5 respectively, and finally conclusion is given in Section 6.

2 Semantic Similarity Measurements

Previous excellent work (**?**; **?**) have shown great performance for STS tasks. Following their works, we engineer the traditional widely used features for semantic similarity measurements (i.e., string-based, corpus-based, machine translation similarity and alignment measures). In this work, we also present our highly interpretable and hyper-parameter free word embedding features from macro and micro views to boost the performance.

2.1 Preprocessing

Several text preprocessing operations are performed before feature engineering: 1) Converting the contractions to their formal writing, e.g., *"doesn't"* is rewritten as *"does not"*. 2) The WordNet-based Lemmatizer implemented in Natural Language Toolkit[1] is used to lemmatize all words to their nearest based forms in WordNet, e.g., *"was"* is lemmatized to *"be"*. 3) Stanford CoreNLP (**?**) is adopted to get the Part-Of-Speech (POS) tag and Named Entity Recognition (NER) tag.

[1] http://www.nltk.org

2.2 Traditional NLP Feature Engineering

2.2.1 String-Based Similarity

Length Features (len): We record the length information of given sentence pairs using the following eight measure functions: $|A|, |B|, |A-B|, |B-A|, |A \cup B|, |A \cap B|, \frac{|A-B|}{|B|}, \frac{|B-A|}{|A|}$, where $|A|$ stands for the number of non-repeated words in sentence A.

Syntactic Features (pos): Since two sentences with similar syntax structure convey similar meaning, we estimate the similarities of syntax structure. We firstly use Stanford CoreNLP toolkit (**?**) to obtain the POS tags of each sentence. Afterwards, we use eight measure functions mentioned in the Length Features on the sets of POS tags to calculate Syntactic Features.

Longest Common Sequence (lcs): In consideration of the different length of sentence pairs, we divide the maximum length of the common subsequence of two sentences by the length of the shorter one.

***n*-grams Overlap Features (*n*-grams):** We obtain n-grams at three different level (i.e., the original word level, the lemmatized word level and the character level). Then Jaccard similarity is used for calculating the similarity of these n-grams pairs. In our experiments, $n = \{1, 2, 3\}$ are used for the word level whereas $n = \{2, 3, 4\}$ are used for the character level.

Named Entities Features (ner): Besides of the surface similarities between words, we also calculate the relatedness of named entities in two sentences using *lcs* function. Seven types of named entities (i.e., *location, organization, data, money, person, time, percent*), recognized by Stanford CoreNLP toolkit (**?**), are considered.

2.2.2 Machine Translation Similarity

Machine Translation (MT) evaluation metrics are designed to assess whether the output of a MT system is semantically equivalent to a set of reference translations. The two given sentences are viewed as one input and one output of a MT system, then we get two MT scores of each MT measure (i.e., *WER, TER, PER, NIST, ROUGE-L, GTM-1*). Two strategies is employed to get MT similarity features, 1). average two MT scores in each MT measure; 2).

concatenate two MT scores in each MT measure.

2.2.3 Corpus-based Features

WordNet Rank Features (wordnet): The above semantic similarities only consider the surface similarities rather than their relations in corpus. Hence, we use graph-based lexical relatedness, which performs with a pre-existing Knowledge Base (KB) (i.e., WordNet), to get the relations of words. Then Personalized PageRank is applied on the Lexical Knowledge Base (LKB) to rank the vertices of the LKB. The details of the method are described in **?**). It outputs a ranking vector of the sentence over KB nodes and the values of the weights are normalized so that all link weights of particular headword sum to one. Finally, we calculate the `Cosine, Manhattan, Euclidean, Jaccard` of the two sentence vectors.

Vector Space Sentence Similarity (lsa): This measure is motivated by the idea of compositionality of distributional vector (**?**). we adopt two distributional word sets released by TakeLab (**?**), where Latent Semantic Analysis (LSA) was performed on the New York Times Annotated Corpus (NYT)[2] and Wikipedia. Then two strategies are used to convert the distributional meaning of words to sentence level: 1). simply summing up. 2). using *tf* to weigh each word vector.

2.2.4 Alignment Measures

(**?**) used delicate word aligner to compute proportion of aligned words across the two input sentences. It aligned words based on their semantic similarity in the two sentences, as well as the similarity between local semantic contexts, which relies on dependencies and surface-form neighbors. The paraphrase Database (PPDB) (**?**) was used to identify semantically similar words. Word pairs are aligned with greedy strategy, in descending order of their similarity.

Global Alignment Features (global): Given sentences S_1 and S_2, single proportion over all words is computed over all words:

$$sim(S_1, S_2) = \frac{n_a(S_1) + n_a(S_2)}{n(S_1) + n(S_2)} \quad (1)$$

where $n(S_i)$ is the number of non-repeated words in S_i, while $n_a(S_i)$ is the number of aligned content words in S_i.

Specific Alignment Features (pos-specific): Taking weight of POS tag of aligned words into consideration, score of aligned *noun* word pair is surely higher than the *adjective*. Using this property, we propose the specific alignment feature, to calculate the aligned words proportion specifically according to POS tag (i.e., noun, verb, adjective, adverb).

2.3 Word Embedding Feature Engineering

Recently, the distributed representations of words (i.e., word embedding) learned by neural networks over a large raw corpus have been shown that they performed significantly better than Latent Semantic Analysis for preserving linear regularities among words (**?**). The training on very large datasets allows the model to learn complex word relationships such as $vec(Berlin) - vec(Germany) + vec(France) \approx vec(Paris)$ (**?**).

As discussed in Section 1, it is very hard to evaluate semantic similarity if no words in the sentence pair in common. Obviously, word embedding features supply the gap. For example, *A storm will spread snow over Shanghai; The earthquakes have shaken parts of Oklahoma*. while *storm* is similar to *earthquake* and *spread* is analogous to *shaken*, *Shanghai* and *Oklahoma* both are locations.

In order to evaluate semantic similarity of a sentence pair, we define the function *INFO* is the semantic information of a word or a sentence carried. Thus the semantic similarity of sentence pair can be regarded as the distance between their *INFOs*. In

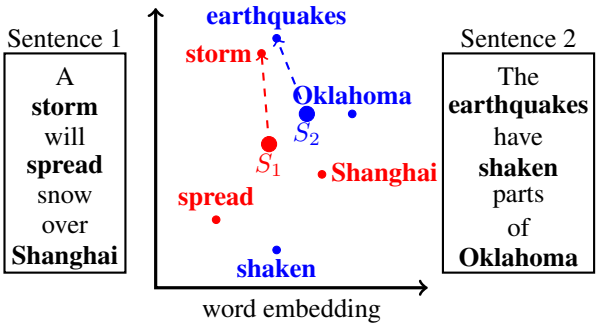

Figure 1: An illustration of the word centroid distance. Points in red is the word from sentence 1 (stopwords is ignored), while blue from sentence 2. S_i is the centroid of points from sentence i.

[2]https://catalog.ldc.upenn.edu/LDC2008T19

other words, given sentence S_1 and S_2,

$$sts(S_1, S_2) = INFO(S_1) - INFO(S_2) \quad (2)$$

where *INFO(S_i)* is the semantic information of sentence S_i. We study the above formula from macro and micro views.

Macro Information Distance: From macro view, we first get the *INFO* of each sentence (i.e., semantic information), and then calculate the distance between them. As follows:

$$\begin{aligned} sts(S_1, S_2) = INFO\left(f(w_1^{S_1}, w_2^{S_1}, \ldots, w_{len(S_1)}^{S_1})\right) \\ - INFO\left(f(w_1^{S_2}, w_2^{S_2}, \ldots, w_{len(S_2)}^{S_2})\right) \end{aligned} \quad (3)$$

where $w_i^{S_j}$ is the word embedding of word i in sentence j and function f is to obtain the sentence representation from word embeddings, such as sum, average or convolution. Assuming that if two sentences are similar, one word in a sentence should have the similar meaning word with another, we use the centroid of word embedding to symbolise the macro *INFO* of the sentence. As showed in Figure 1, S_i represents for sentence i, and the distance between S_1 and S_2 represents for the similarity of the sentence pair. What is more, *storm* and *earthquakes* are the most important word in the sentence pair, we surely should give them more weight. As they are similar, the distance centroid tend to be close (the dashed lines). We use *idf* from datasets to weigh the importance.

Micro Information Distance: As for micro view, we first get the *INFO* of each word, and evaluate the distance between the sentence pair according to the *INFO* of words. The formula is described as Equation 4.

$$\begin{aligned} sts(S_1, S_2) = & INFO(w_1^{S_1}) + INFO(w_2^{S_1}) + \ldots + INFO(w_{len(S_1)}^{S_1}) \\ & - INFO(w_1^{S_2}) - INFO(w_2^{S_2}) - \ldots - INFO(w_{len(S_2)}^{S_2}) \end{aligned} \quad (4)$$

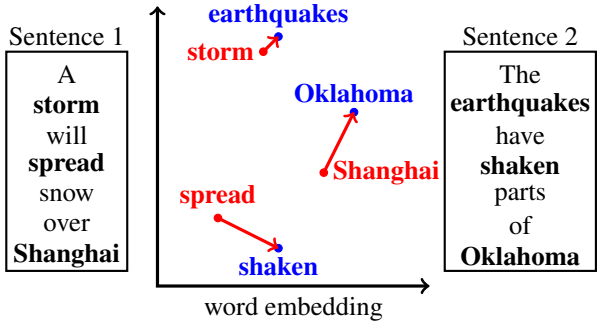

Figure 2: An illustration of the word mover's distance, an instance of micro information distance.

Our goal is to incorporate the semantic similarity between each word pairs into the micro information distance of sentence. Here, we adopted word mover's distance (**?**), the minimum cumulative distance that all word in sentence 1 need to travel to exactly match sentence 2, showed in Figure 2. For more details, see **?**).

Word Embedding Features: ?) shows that heterogenous feature outperform a single feature, and we use three embeddings (**?**; **?**; **?**) as our initial word vector input. Incidentally, the distance is substitutable, and we replace it with different measurements (i.e., *cosine distance, Manhanttan distance, Euclidean distance, Pearson coefficient, Spearman coefficient, Kendall tau coefficient*). Specially, because of the high time complexity of word mover's distance, we only train it on *word2vec* (**?**), although other embeddings are also plausible.

3 Experiments

3.1 Datasets

We collect all the datasets from 2012 to 2015 as training data. Each dataset consists of a number of sentence pairs and each pair has a human-assigned similarity score in the range[0,5] which increases with similarity. The datasets are collected from different but related domains. We briefly describe data in Table 1, Refer **?**) for details. We emphasize dataset with symbol * for that this dataset appears both in training and test sets, which is very useful to our third submission **S2** (see Section 3.4 for more details).

3.2 Evaluation Measurement

In order to evaluate the performance of different algorithms, we adopt the official evaluation measure, i.e, Pearson correlation coefficient for each individual test set, and a weighted sum of all correlations is used as final evaluation metric. It weights according to the number of gold sentence pairs. The weight of a test set is equal to the rate of the gold sentences pairs in all the gold sentences.

Training Set			Test Set		
Dataset	**Input**	**Gold**	**Dataset**	**Input**	**Gold**
MSRpar	1500	1500	answers-answers	1572	254
SMTeuroparl	1193	1193	plagiarism	1271	230
headlines*	3000	2250	headlines*	1498	249
SMTnews	399	399	postediting	3287	244
MSRvid	1500	1500	question-question	1555	209
OnWN	2061	2061	-	-	-
FNWN	189	189	-	-	-
images	2250	1500	-	-	-
deft-forum	450	450	-	-	-
deft-news	300	300	-	-	-
tweet-news	750	750	-	-	-
answers-forums	1500	375	-	-	-
answers-students	1500	750	-	-	-
belief	2000	375	-	-	-
All	19092	13592	All	9183	1186

Table 1: The statistics of all datasets for STS task. Dataset with symbol * represents that this dataset appears both in training and test sets.

3.3 Learning Algorithm

We conduct a series of experiments using all features discussed in Section 2.2 and 2.3 to obtain the optimized learning algorithm. Three supervised learning methods are explored: Support Vector Regression (SVR), Random Forest (RF) and Gradient Boosting (GB). These supervised learning algorithms are implemented using scikit-learn toolkit (**?**).we use all the datasets from STS Task 2015 as development data while others from STS Task 2012 to 2014 as training data.

To configure the parameters in the regression algorithm, i.e., the trade-off parameter c in SVR, the number of trees n_{RF} in RF and the number of boosting stages n_{GB} in GB, we make a grid search for c in $[0.01, 0.1, 1, 10]$, n_{RF} from 5 to 100 with step 5 and n_{GB} from 10 to 300 with step 10.

Regression	belief	answers -students	headlines	images	answers -forums	Weighted Mean
SVR(c=1)	0.7413	0.7359	0.8168	0.8660	0.7400	0.7898
RF(n=40)	0.7466	0.7100	0.8200	0.8534	0.7398	0.7816
GB(n=140)	**0.7655**	0.7484	**0.8439**	**0.8791**	**0.7469**	**0.8080**
DLSCU-S1	0.7491	**0.7725**	0.8250	0.8644	0.7390	0.8015

Table 2: Pearson coefficient of development data using different algorithms with different hyperparameter, as well as top rank results on STS 2015 test data.

Table 2 shows the best result of each algorithm , as well as the top runs on STS 2015 test data. GB(n=140) outperformed other algorithms on all datasets. We choose GB(n=140) as our final regression algorithm on our next series experiments. Also, without any specific training dataset selections or choosing suitable features, we achieve the considerable results compare to the top runs on STS 2015 Task. **?**) has shown that specific training datasets with similar domains and enough data will yield better results than an all-inclusive training datasets. And next we endeavor to select specific training sets and suitable features.

3.4 System Setups

We build three different systems according to the usage of training datasets as follows.

U-SEVEN: This is an unsupervised system based on the word aligner described in (**?**) without any training data. We evaluate semantic similarity by adopting straight-forward measurements (i.e., longest common sequence, alignment feature, corpus-based feature, all four features from word embedding features), which are averaged to get the final score. We adopt *cosine distance, Pearson coefficient, Spearman coefficient* as the distance measurements, which perform better results on our preliminary experiments.

S1-All: We use all the training datasets and build a single global regression model regardless of domain information of different test datasets. In order to make better use of these features and improve the performance, we construct feature selection procedure on development set (i.e., test set of STS 2015 Task) and vote for the preserved feature sets. As for feature selection strategy, we adopt *hill climbing*: keep adding one type feature at a time until no further improvement can be achieved.

S2: **?**) has shown that taking all the training datasets into consideration may hurt the performance since training and test sets are from different domains. Hence, for each test set, we select the datasets which are most similar, taking source, average length of sentences and word mover's distance (discussed in Section 2.3) into consideration. For the data set with symbol * (i.e., *headlines*), we use all *headlines* pairs. For *answers-answers* and *question-question*, we use *belief, deft-forums, answers-students, answers-forums* pairs. For postediting, we use *SMTeuropar* and *MSRpar* pairs. For plagiarism, we use *onWN* and *FNWN* pairs.

4 Results on Training Data

According to the above preliminary experimental results, we employ GB(n=140) algorithm as our final

regression algorithm. In order to explore the influences of word embedding features and make better use of all the above features, we construct feature selection experiment on development set (i.e., test set of STS 2015 Task) and vote for the preserved feature sets.

Feature		belief	answers -students	headlines	image	answers -forums
String-based	len	-	-	√	√	-
	pos	-	-	√	-	√
	lcs	-	-	√	√	√
	n-grams	-	√	√	√	√
	ner	-	√	√	-	-
Machine Translation	average	√	-	√	√	-
	concat	-	-	-	-	-
Corpus-based	wordnet	√	√	√	-	√
	lsa	-	√	√	√	√
Alignment	global	-	√	√	√	√
	pos-specific	√	√	√	√	√
Word Embedding (Macro)	word2vec	√	√	√	√	√
	glove	-	-	-	-	-
	turian's	√	√	√	√	-
Word Embedding (Micro)	wmd	√	√	√	√	√
Our Results		**0.7835**	0.7713	**0.8455**	**0.8808**	**0.7636**
Best Scores		0.7717	**0.7879**	0.8417	0.8713	0.7390

Table 3: Results of feature selection experiments on STS 2015 test data. The last row shows the the best scores of all submitted system on STS 2015 task.

Table 3 shows the results of feature selection experiments on STS 2015 test data. From the table, we find that 1) Word embedding features, the positive complementary of macro perspective and micro perspective, indeed improve results. 2) Specific alignment features can compensate for the weaknesses of global alignment features. 3) Since concatenate MT metrics and glove features does not perform well on all five data sets, we remove them from our feature sets and train our S1-All model with the preserved features.

5 Results on Test Data

Table 4 summarizes the results of our submitted runs on test datasets officially released by the organizers, as well as the top runs. In terms of weighted mean of Pearson measurement, system S1-All performs the best while our corpus-specific system **S2** performs the worst. We think the measurement to choose training data from the candidate datesets in main task are ill-suited. It is noteworthy that on plagiarism and postediting, our unsupervised model U-SEVEN achieves much better results than the supervised model (i.e., S1-All, S2), which indicates the efficiency of the ensemble of similar measurements.

Dataset	Runs			Best Score
	U-SEVEN	S1-All	S2	
answers-answers	0.4774	0.5697	**0.5715**	0.6923
plagiarism	**0.8301**	0.8250	0.7733	0.8413
headlines	0.7668	**0.8121**	0.7903	0.8274
postediting	**0.8423**	0.8234	0.7496	0.8669
question-question	0.7191	**0.7311**	0.6763	0.7470
weighted mean	0.7242	**0.7507**	0.7116	0.7780

Table 4: The results of our three runs on STS 2016 test datasets.The rightmost column shows the best score by any system. The last row shows the value of the officially evaluation metric.

On *answer-answer* set, the gap between top systems and our systems is about 12%. According to our investigations on this set, we find that certain sentence pairs are similar in syntactical structure but express different meanings. For example,

$$\begin{cases} \textit{You should do it.} \\ \textit{You can do it, too.} \end{cases} \quad \begin{cases} \textit{It's pretty much up to you.} \\ \textit{It's much better to ask.} \end{cases}$$

Our assumption (i.e., two sentences with similar syntax structure convey similar meaning) does not apply to the above condition.

6 Conclusion

We use the traditional NLP features including string-based features, corpus-based features and alignment features for textual similarity estimation, as well as efficient word embedding features. It is also worth pointing out that our word embedding features are highly interpretable and hyper-parameter free, as well as they are straight-forward to measure semantic textual similarity. The difference between top system and our best system is about 2.8%, which means our systems are promising. Noticing the gap between top system and our systems on *answer-answer* set, we will explore to find the central words of sentences in future work.

Acknowledgments

This research is supported by grants from Science and Technology Commission of Shanghai Municipality (14DZ2260800 and 15ZR1410700), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213).