ECNU at SemEval-2017 Task 1: Leverage Kernel-based Traditional NLP features and Neural Networks to Build a Universal Model for Multilingual and Cross-lingual Semantic Textual Similarity

Junfeng Tian[1], **Zhiheng Zhou**[1], **Man Lan**[1,2*], **Yuanbin Wu**[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
{jftian, zhzhou}@stu.ecnu.edu.cn
{mlan, ybwu}@cs.ecnu.edu.cn

Abstract

To model semantic similarity for multilingual and cross-lingual sentence pairs, we first translate foreign languages into English, and then build an efficient monolingual English system with multiple NLP features. Our system is further supported by deep learning models and our best run achieves the mean Pearson correlation 73.16% in primary track.

1 Introduction

Sentence semantic similarity is the building block of natural language understanding. Previous Semantic Textual Similarity (STS) tasks in SemEval focused on judging sentence pairs in English and achieved great success. In SemEval-2017 STS shared task concentrates on the evaluation of sentence semantic similarity in multilingual and cross-lingual (Agirre et al., 2017). There are two challenges in modeling multilingual and cross-lingual sentence similarity. On the one hand, this task requires human linguistic expertise to design specific features due to the different characteristics of languages. On the other hand, lack of enough training data for a particular language would lead to a poor performance.

The SemEval-2017 STS shared task assesses the ability of participant systems to estimate the degree of semantic similarity between monolingual and cross-lingual sentences in Arabic, English and Spanish, which is organized into a set of six secondary sub-tracks (Track 1 to Track 6) and a single combined primary track (Primary Track) achieved by submitting results for all of the secondary sub-tracks. Specifically, track 1, 3 and 5 are to determine STS scores for monolingual sentence pairs in Arabic, Spain and English, respectively. Track 2, 4, and 6 involve estimating STS scores for cross-lingual sentence pairs from the combination of two particular languages, i.e., Arabic-English, Spanish-English and surprise language (here is Turkish)-English cross-lingual pairs. Given two sentences, a continuous valued similarity score on a scale from 0 to 5 is returned, with 0 indicating that the semantics of the sentences are completely independent and 5 signifying semantic equivalence. The system is assessed by computing the Pearson correlation between system returned semantic similarity scores and human judgements.

To address this task, we first translate all sentences into English through the state-of-the-art machine translation (MT) system, i.e., Google Translator[1]. Then we adopt a combination method to build a universal model to estimate semantic similarity, which consists of traditional natural language processing (NLP) methods and deep learning methods. For traditional NLP methods, we design multiple effective NLP features to depict the semantic matching degree and then supervised machine learning-based regressors are trained to make prediction. For neural networks methods, we first obtain distributed representations for each sentence in sentence pairs and then feed these representations into end-to-end neural networks to output similarity scores. Finally, the scores returned by the regressors with traditional NLP methods and by the neural network models are equally averaged to get a final score to estimate semantic similarity.

2 System Description

Figure 1 shows the overall architecture of our system, which consists of the following three modules:

[1]https://cloud.google.com/translate/

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 191–197,
Vancouver, Canada, August 3 - 4, 2017.

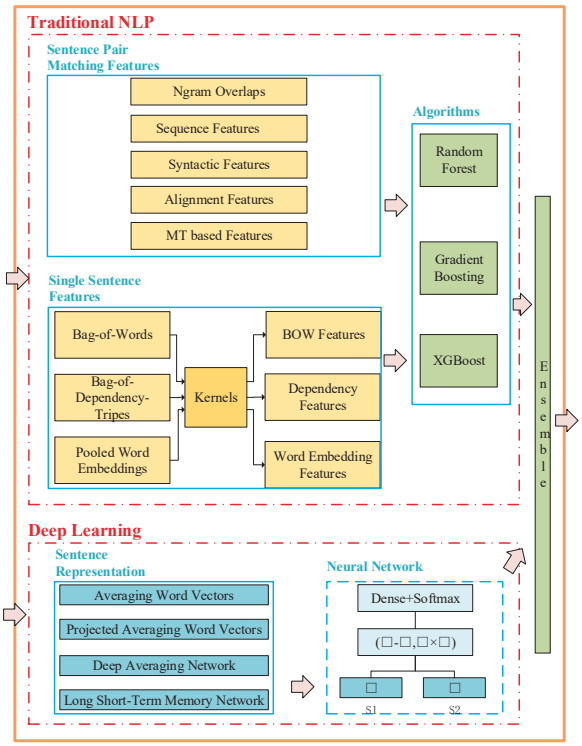

Figure 1: The system architecture

Traditional NLP Module is to extracts two kinds of NLP features. The sentence pair matching features are to directly calculate the similarity of two sentences from several aspects and the single sentence features are to first represent each sentence in NLP method and then to adopt kernel-based method to calculate the similarity of two sentences. All these NLP-based similarity scores act as features to build regressors to make prediction.

Deep Learning Module is to encode input sentence pairs into distributed vector representations and then to train end-to-end neural networks to obtain similarity scores.

Ensemble Module is to equally average the above two modules to get a final score.

Next, we will describe the system in detail.

2.1 Traditional NLP Module

In this section, we give the details of feature engineering and learning algorithms.

2.1.1 Sentence Pair Matching Features

Five types of sentence pair matching features are designed to directly calculate the similarity of two sentences based on the overlaps of character/word/sequence, syntactic structure, alignment and even MT metrics.

***N*-gram Overlaps:** Let S_i be the sets of consecutive n-grams, and the n-gram overlap (denoted as ngo) is defined as (Šarić et al., 2012):

$$\text{ngo}(S_1, S_2) = 2 \cdot \left(\frac{|S_1|}{|S_1 \cap S_2|} + \frac{|S_2|}{|S_1 \cap S_2|}\right)^{-1}$$

We obtain n-grams at three different levels (i.e., the original and lemmatized word, the character level), where $n = \{1, 2, 3\}$ are used for word level and $n = \{2, 3, 4, 5\}$ are used for character level. Finally, we collect 10 features.

Sequence Features: Sequence features are designed to capture more enhanced sequence information besides the n-gram overlaps. We compute the *longest common prefix / suffix / substring / sequence* and *levenshtein distance* for each sentence pair. Note that the stopwords are removed and each word is lemmatized so as to estimate sequence similarity more accurately. As a result, we get 5 features.

Syntactic Parse Features: In order to model tree structured similarity between two sentences rather than sequence-based similarity, inspired by Moschitti (2006), we adopt tree kernels to calculate the similarity between two syntactic parse trees. In particular, we calculate the number of common substructures in three different kernel spaces, i.e., *subtree* (ST), *subset tree* (SST), *partial tree* (PT). Thus we get 3 features.

Alignment Features: Sultan et al. (2015) used word aligner to align matching words across a pair of sentences, and then computes the proportion of aligned words as follows:

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

where $n_a(S)$ and $n(S)$ is the number of aligned and non-repeated words in sentence S.

To assign appropriate weights to different words, we adopt two weighting methods: i) weighted by five POS tags (i.e., noun, verb, adjective, adverb and others; we first group words in two sentences into 5 POS categories, then for each POS category we compute the proportion of aligned words, and we get 5 features as a result. ii) weighted by IDF values (calculated in each dataset separately). Totally, we collect 7 alignment features.

MT based Features: Following previous work in (Zhao et al., 2014) and (Zhao et al., 2015), we use MT evaluation metrics to measure the semantic equivalence of the given sentence pairs. Nine

MT metrics (i.e., *BLEU, GTM-3, NIST, -WER, -PER, Ol, -TERbase, METEOR-ex, ROUGE-L*) are used to assess the similarity. These 9 MT based features are calculated using the Asiya Open Toolkit[2].

Finally, we collect a total of 34 sentence pair matching features.

2.1.2 Single Sentence Features

Unlike above sentence pair matching features to directly estimate matching score between two sentences, the single sentence features are to represent each sentence in the same vector space to calculate the sentence similarity. We design the following three types of features.

BOW Features: Each sentence is represented as a Bag-of-Words (BOW) and each word (i.e., dimension) is weighted by its IDF value.

Dependency Features: For each sentence, its dependency tree is interpreted as a set of triples, i.e., (dependency-label, governor, subordinate). Similar to BOW, we treat triples as words and represent each sentence as Bag-of-Triples.

Word Embedding Features: Each sentence is represented by concatenating *min/max/average* pooling of vector representations of words. Note that for each word, its vector is weighted by its IDF value. Table 1 lists four the state-of-the-art pretrained word embeddings used in this work.

Embedding	Dimension	Source
word2vec Mikolov et al. (2013)	300d	GoogleNews-vectors-negative300.bin
GloVe Pennington et al. (2014)	100d	glove.6B.100d.txt
	300d	glove.6B.300d.txt
paragram Wieting et al. (2015)	300d	paragram_300_sl999.txt

Table 1: Four pretrained word embeddings

However, in comparison with the number of sentence pair matching features (33 features), the dimensionality of single sentence features is huge (approximately more than $71K$ features) and thus it would suppress the discriminating power of sentence pair matching features. Therefore, In order to reduce the high dimensionality of single sentence features, for each single sentence feature, we use 11 kernel functions to calculate sentence pair similarities. Table 2 lists the 11 kernel functions we used in this work. In total we collect 33 single sentence features, which is of the same order of magnitude as sentence pair matching features.

Type	Measures
linear kernel	Cosine distance, Manhanttan distance, Euclidean distance, Chebyshev distance
stat kernel	Pearson coefficient, Spearman coefficient, Kendall tau coefficient
non-linear kernel	polynomial, rbf, laplacian, sigmoid

Table 2: List of 11 kernel functions

Finally, these 67 NLP features are standardized into $[0, 1]$ using max-min normalization before building regressor models.

2.1.3 Regression Algorithms

Five learning algorithms for regression are explored, i.e., Random Forests (RF), Gradient Boosting (GB) Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and XGBoost (XGB). Specially, the first four algorithms are implemented in scikit-learn toolkit[3], and XGB is implemented in xgboost[4]. In preliminary experiments, SVM and SGD underperformed the other three algorithms and thus we adopt RF, GB and XGB in following experiments.

2.2 Deep Learning Module

Unlike above method adopting manually designed NLP features, deep learning based models are to calculate semantic similarity score with the pretrained word vectors as inputs. Four pretrained word embeddings listed in Table 1 are explored and the paragram embeddings achieved better results in preliminary experiments. We analyze and find the possible reason may be that the paragram embeddings are trained on Paraphrase Database[5], which is an extensive semantic resource that consists of many phrase pairs. Therefore, we use paragram embeddings to initialize word vectors.

Based on pretrained word vectors, we adopt the following four methods to obtain single sentence vector as (Wieting et al., 2015):

(1) by simply averaging the word vectors in single sentence;

(2) after (1), the resulting averaged vector is multiplied by a projection matrix;

(3) by using deep averaging network (DAN, Iyyer et al. (2015)) consisting of multiple layers as well as nonlinear activation functions;

[2] http://asiya.cs.upc.edu/demo/asiya_online.php

[3] http://scikit-learn.org/stable/

[4] https://github.com/dmlc/xgboost

[5] http://www.cis.upenn.edu/~ccb/ppdb/

(4) by using long short-term memory network (LSTM, Hochreiter and Schmidhuber (1997)) to capture long-distance dependencies information.

In order to obtain the vector of sentence pair, given two single sentence vectors, we first use a element-wise subtraction and a multiplication and then concatenate the two values as the final vector of sentence pair representation. At last, we use a fully-connected neural network and output the probability of similarity based on a softmax function. Thus we obtain 4 deep learning based scores.

To learn model parameters, we minimize the KL-divergence between the outputs and gold labels, as in Tai et al. (2015). We adopt Adam (Kingma and Ba, 2014) as optimization method and set learning rate of 0.01.

2.3 Ensemble Module

The NLP-based scores and the deep learning based scores are averaged in the ensemble module to obtain the final score.

3 Experimental Settings

Datasets: SemEval-2017 provided 7 tracks in monolingual and cross-lingual language pairs. We first translate all sentences into English via Google Translator and then we build a universal model on only English pairs. The training set we used is all the monolingual English dataset from SemEval STS task (2012-2015) consisting of $13,592$ sentence pairs.

For each track, we grant the training datasets provided by SemEval-2017 as development set. Table 3 lists the statistics of the development and the test data for each track in SemEval-2017.

Track	Language Pair	Development		Test
		Pairs	Dataset	Pairs
Track 1	Arabic-Arabic (AR-AR)	1088	MSRpar, MSRvid, SMTeuroparl (2017)	250
Track 2	Arabic-English (AR-EN)	2176	MSRpar, MSRvid, SMTeuroparl (2017)	250
Track 3	Spanish-Spanish (SP-SP)	1555	News, Wiki (2014, 2015)	250
Track 4a	Spanish-English (SP-EN)	595	News, Multi-source (2016)	250
Track 4b	Spanish-English WMT news data (SP-EN-WMT)	1000	WMT (2017)	250
Track 5	English-English (EN-EN)	1186	Plagiarsism, Postediting, Ans.-Ans., Quest.-Quest., HDL (2016)	250
Track 6	English-Turkish (EN-TR)	-	-	500

Table 3: The statistics of development and test set.

Almost all test data is from *SNLI*, except for Track 4b from *WMT*. This can explain why on *Track 4b SP-EN-WMT*, the performance is very poor. So we perform $10 - fold$ cross validation (CV) on *Track 4b SP-EN-WMT*.

Preprocessing: All sentences are translated into English via Google Translator. The Stanford CoreNLP (Manning et al., 2014) is used for tokenization, lemmatization, POS tagging and dependency parsing.

Evaluation: For Track 1 to Track 6, Pearson correlation coefficient is used to evaluate each individual test set. For Primary Track, since it is achieved by submitting results of all the secondary sub-tracks, a macro-averaged weighted sum of all correlations on sub-tracks is used for evaluation.

4 Results on Training Data

A series of comparison experiments on *English STS 2016* training set have been performed to explore different features and algorithms.

4.1 Comparison of NLP Features

Table 4 lists the results of different NLP features with GB learning algorithm. We find that: (1) the simple *BOW Features* with kernel functions are effective for sentence semantic similarity. (2) The combination of all these NLP features achieved the best results, which indicates that all features make contributions. Therefore we do not perform feature selection and use all these NLP features in following experiments.

4.2 Comparison of Learning Algorithms

Table 5 lists the results of different algorithms using all NLP features as well as deep learning scores. We find:

(1) Regarding machine learning algorithms, RF and GB achieve better results than XGB. GB performs the best on 3 and RF performs the best on 2 of 5 datasets.

(2) Regarding deep learning models, DL-word and DL-proj outperform the other 2 non-linear models on all the 5 datasets. This result is consistent with the findings in (Wieting et al., 2015):"In out-of-domain scenarios, simple architectures such as word averaging vastly outperform LSTMs."

(3) All ensemble methods significantly improved the performance. The ensemble of 3 machine learning algorithms (RF+GB+XGB) outperforms any single learning algorithm. Similarly, the ensemble of the 4 deep learning models (DL-all) promotes the performance to 75.28%, which is sig-

English STS 2016						
NLP Features	**Postediting**	**Ques.-Ques.**	**HDL**	**Plagiarism**	**Ans.-Ans.**	**Weighted mean**
BOW features	0.8388	0.6577	0.7338	0.7817	0.6302	0.7322
Alignment Features	0.8125	0.6243	0.7642	0.7883	**0.6432**	0.7312
Ngram Overlaps	0.8424	0.5864	0.7581	0.8070	0.5756	0.7203
Sequence Features	**0.8428**	0.6115	0.7337	0.7983	0.4838	0.7000
Word Embedding Features	0.8128	0.6378	0.7625	0.7955	0.4598	0.6992
MT based Features	0.8412	0.5558	0.7259	0.7617	0.5084	0.6851
Dependency Features	0.7264	0.5381	0.4634	0.5820	0.3431	0.5328
Syntactic Parse Features	0.5773	0.0846	0.4940	0.3976	0.0775	0.3376
All Features	0.8357	**0.6967**	**0.7964**	**0.8293**	0.6306	**0.7618**
Rychalska et al. (2016)	0.8352	0.6871	**0.8275**	**0.8414**	**0.6924**	**0.7781**
Brychcín and Svoboda (2016)	0.8209	0.7020	0.8189	0.8236	0.6215	0.7573
Afzal et al. (2016)	**0.8484**	**0.7471**	0.7726	0.8050	0.6143	0.7561

Table 4: Feature comparison on English STS 2016, the last three are top three systems in STS 2016

English STS 2016							
Algorithm		**Postediting**	**Ques.-Ques.**	**HDL**	**Plagiarism**	**Ans.-Ans.**	**Weighted mean**
Single Model	RF	0.8394	0.6858	0.7966	0.8259	0.5882	0.7518
	GB	0.8357	0.6967	0.7964	0.8293	0.6306	0.7618
	XGB	0.7917	0.6237	0.7879	0.8175	0.6190	0.7333
	DL-word	0.8097	0.6635	0.7839	0.8003	0.5614	0.7283
	DL-proj	0.7983	0.6584	0.7910	0.7892	0.5573	0.7234
	DL-dan	0.7695	0.4200	0.7411	0.6876	0.4756	0.6274
	DL-lstm	0.7864	0.5895	0.7584	0.7783	0.5182	0.6921
Ensemble	RF+GB+XGB	0.8298	0.6969	0.8086	0.8313	0.6234	0.7622
	DL-all	0.8308	0.6817	0.8160	0.8261	0.5854	0.7528
	EN-seven	**0.8513**	**0.7077**	**0.8288**	**0.8515**	**0.6647**	**0.7851**

Table 5: Algorithms comparison on English STS 2016 datasets

nificantly better than single model and is comparable to the result using expert knowledge. Furthermore, the ensemble of 3 machine learning algorithms and 4 deep learning models by averaging these 7 scores (EN-seven), achieves the best results on all of the development set in *English STS 2016*. It suggests that the traditional NLP methods and the deep learning models are complementary to each other and their combination achieves the best performance.

4.3 Results on Cross-lingual Data

To address cross-lingual, we first translate cross-lingual pairs into monolingual pairs and then adopt the universal model to estimate semantic similarity. Thus, language translation is critical to the performance. The first straightforward way for translation (Strategy 1) is to translate foreign language into English. We observe that it is more likely to produce synonyms when using Strategy 1. For example: one English-Spanish pair

`The respite was short.`
`La tregua fue breve.`

is translated into English-English pair,

`The respite was short.`
`The respite was brief.`

where *short* and *brief* are synonyms produced by MT rather than their actual literal meaning expressed in original languages. Reminding that one MT system may be in favour of certain words and it also can translate English into foreign language. Thus we propose Strategy 2 for translation, i.e., we first translate the English sentence into foreign target language and then roll back to English via MT again. Under Strategy 2, the above example English-Spanish pair is translated into the same English sentence:

`The respite was brief.`

Table 6 compares the results of the two strategies on cross-lingual data. It is clear that Strategy 2 achieves better performance, which indicates that the semantic difference between synonyms in cross-lingual pairs resulting from MT are different from that in monolingual pairs.

4.4 Results on Spanish-English WMT

On Spanish-English WMT dataset, the system performance dropped dramatically. The possible reason may lie in that they are from different domains. Therefore, we use 10-fold cross validation on

Cross-lingual STS 2016				
Algorithm		**news**	**multisource**	**Weighted mean**
Stategy 1	RF[1]	0.9101	0.8259	0.8686
	GB[1]	0.8911	0.8220	0.8570
	XGB[1]	0.8795	0.7984	0.8394
Stategy 2	RF[2]	0.9009	0.8405	0.8711
	GB[2]	0.9122	0.8441	0.8786
	XGB[2]	0.8854	0.8265	0.8563
	RF+GB+XGB[2]	**0.9138**	**0.8474**	**0.8810**
	DL-all[2]	0.8016	0.7442	0.7732
	EN-seven[2]	0.8832	0.8291	0.8565
Brychcín and Svoboda (2016)		0.9062	0.8190	0.8631

Table 6: Pearson correlations on Cross-lingual STS 2016, the last row is the top system in 2016.

this dataset for evaluation. Table 7 list the results on Spanish-English WMT, where the last column (wmt(CV)) of show that using the in-domain dataset achieves better performance.

Take a closer look at this dataset, we find that several original Spanish sentences are meaningless. For example, the English-Spanish pair

`His rheumy eyes began to cloud.`
`A sus ojos rheumy comenzóa nube.`

has a score of `1` as the second is not a proper Spanish sentence. Since there are many meaningless Spanish sentences in this dataset sourced from MT evaluation, we speculate that these meaningless sentences are made to be used as negative training samples for MT model. Thus, only on this dataset, we grant Spanish as target language and translate English sentences into Spanish sentences. After that, we use 9 MT evaluation metrics (mentioned in Section 2.1) to generate **MT based Features**. Then these 9 MT metrics are averaged as the similarity score (MT(es)^3).

Spanish-English WMT		
Algorithm	**wmt**	**wmt(CV)**
RF[2]	0.1761	0.2635
GB[2]	0.1661	0.2053
XGB[2]	0.1627	0.2620
RF+GB+XGB[2]	0.1739	0.2677
DL-all[2]	0.0780	-
EN-seven[2]	0.1393	-
MT(es)^3	0.2858	0.2858
RF+GB+XGB[2]+MT(es)^3	**0.2889**	**0.3789**
EN-seven[2]+MT(es)^3	0.2847	-

Table 7: Pearson correlations on Spanish-English WMT. MT(es)^3 is calculated using their translated Spanish-Spanish form. We did not perform cross validation in deep learning models and did not ensemble them due to time constraint.

From Table 7, we see that the MT(es)^3 score alone achieves 0.2858 on *wmt* in terms of Pearson correlation, which even surpasses the best performance (0.2677) of ensemble model. Based on this, we also combine the ensemble model with MT(es)^3 and their averaged score achieves 0.3789 in terms of Pearson correlation.

4.5 System Configuration

Based on the above results, we configure three following systems:

Run 1: all features using RF algorithms. (RF)

Run 2: all features using GB algorithms. (GB)

Run 3: ensemble of three algorithms and four deep learning scores. (EN-seven)

Particularly, we train *Track 4b SP-EN-WMT* using the *wmt* dataset provided in SemEval-2017 and Run 2 and Run 3 on this track are combined with MT(es)^3 features.

5 Results on Test Data

Table 8 lists the results of our submitted runs on test datasets. We find that: (1) GB achieves slightly better performance than RF, which is consistent to that in training data; (2) the ensemble model significantly improves the performance on all datasets and enhance the performance of *Primary Track* by about 3% in terms of Pearson coefficient; (3) on *Track 4b SP-EN-WMT*, combining with MT(es)^3 significantly improves the performance.

The last three rows list the results of two top systems and one baseline system provided by organizer. The baseline is to use the *cosine* similarity of one-hot vector representations of sentence pairs. On all language pairs, our ensemble system achieves the best performance. This indicates that both the traditional NLP methods and the deep learning methods make contribution to performance improvement.

6 Conclusion

To address mono-lingual and cross-lingual sentence semantic similarity evaluation, we build a universal model in combination of traditional NLP methods and deep learning methods together and the extensive experimental results show that this combination not only improves the performance but also increases the robustness for modeling similarity of multilingual sentences. Our future work will concentrate on learning reliable sentence pair representations in deep learning.

Run	Primary	Track 1 AR-AR	Track 2 AR-EN	Track 3 SP-SP	Track 4a SP-EN	Track 4b SP-EN-WMT	Track 5 EN-EN	Track 6 EN-TR
Run 1: RF	0.6940	0.7271	0.6975	0.8247	0.7649	0.2633	0.8387	0.7420
Run 2: GB	0.7044	0.7380	0.7126	0.8456	0.7495	0.3311*	0.8181	0.7362
Run 3: EN-seven	**0.7316**	**0.7440**	**0.7493**	**0.8559**	**0.8131**	**0.3363***	**0.8518**	**0.7706**
Rank 2: BIT	0.6789	0.7417	0.6965	0.8499	0.7828	0.1107	0.8400	0.7305
Rank 3: HCTI	0.6598	0.7130	0.6836	0.8263	0.7621	0.1483	0.8113	0.6741
Baseline	0.5370	0.6045	0.5155	0.7117	0.6220	0.0320	0.7278	0.5456

Table 8: The results of our three runs on STS 2017 test datasets.

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) and NSFC (61402175).

References

Naveed Afzal, Yanshan Wang, and Hongfang Liu. 2016. Mayonlp at semeval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model. In *Proceedings of SemEval 2016*. San Diego, California.

Eneko Agirre, Daniel Cer, Mona Diab, Lopez-Gazpio Inigo, and Specia Lucia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SemEval 2017*.

Tomáš Brychcín and Lukáš Svoboda. 2016. Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. In *Proceedings of SemEval 2016*. San Diego, California, pages 588–594.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL 2015*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

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 ACL 2014*. pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*. pages 3111–3119.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML 2006*. Springer, pages 318–329.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*. pages 1532–1543.

Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak, and Piotr Andruszkiewicz. 2016. Samsung poland nlp team at semeval-2016 task 1: Necessity for diversity; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In *Proceedings of SemEval 2016*. San Diego, California, pages 602–608.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. Dls@cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of SemEval 2015*. Denver, Colorado.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of SemEval 2012*. Montréal, Canada, pages 441–448.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198* .

Jiang Zhao, Man Lan, and Jun Feng Tian. 2015. Ecnu: Using traditional similarity measurements and word embedding for semantic textual similarity estimation. In *Proceedings of SemEval 2015*. Denver, Colorado.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogeneous measures for semantic relatedness and textual entailment. In *Proceedings of SemEval 2014*. Dublin, Ireland.