HCTI at SemEval-2017 Task 1: Use convolutional neural network to evaluate Semantic Textual Similarity

Yang SHAO
Hitachi, Ltd. / Higashi koigakubo 1-280, Kokubunji-shi, Tokyo, Japan
yang.shao.kn@hitachi.com

Abstract

This paper describes our convolutional neural network (CNN) system for the Semantic Textual Similarity (STS) task. We calculated semantic similarity score between two sentences by comparing their semantic vectors. We generated a semantic vector by max pooling over every dimension of all word vectors in a sentence. There are two key design tricks used by our system. One is that we trained a CNN to transfer GloVe word vectors to a more proper form for the STS task before pooling. Another is that we trained a fully-connected neural network (FCNN) to transfer the difference of two semantic vectors to the probability distribution over similarity scores. All hyperparameters were empirically tuned. In spite of the simplicity of our neural network system, we achieved a good accuracy and ranked 3rd on primary track of SemEval 2017.

1 Introduction

Semantic Textual Similarity (STS) is the task of determining the degree of semantic similarity between two sentences. STS task is a building block of many natural language processing (NLP) applications. Therefore, it has received a significant amount of attention in recent years. STS tasks in SemEval have been held from 2012 to 2017 (Cer et al., 2017). Successfully estimating the degree of semantic similarity between two sentences requires a very deep understanding of both sentences. Well performing STS methods can be applied to many other natural language understanding tasks including paraphrasing, entailment detection, answer selection, hypothesis evidencing, machine translation (MT) evaluation and quality estimation, summarization, question answering (QA) and short answer grading.

Measuring sentence similarity is challenging for two reasons. One is the variability of linguistic expression and the other is the limited amount of annotated training data. Therefore, conventional NLP approaches, such as sparse, hand-crafted features are difficult to use. However, neural network systems (He et al., 2015a; He and Lin, 2016) can alleviate data sparseness with pre-training and distributed representations. We propose a convolutional neural network system with 5 components:

1) Enhance GloVe word vectors by adding handcrafted features.
2) Transfer the enhanced word vectors to a more proper form by a convolutional neural network.
3) Max pooling over every dimension of all word vectors to generate semantic vector.
4) Generate semantic difference vector by concatenating the element-wise absolute difference and the element-wise multiplication of two semantic vectors.
5) Transfer the semantic difference vector to the probability distribution over similarity scores by fully-connected neural network.

2 System Description

Figure 1 provides an overview of our system. The two sentences to be semantically compared are first pre-processed as described in subsection 2.1. Then the CNN described in subsection 2.2 combines the word vectors from each sentence into an appropriate sentence level embedding. After that, the methods described in subsection 2.3 are used to compute representations that compare paired sentence level embeddings. Then, a fully-connected neural network (FCNN) described in subsection 2.4 transfers the semantic difference vector to a probability distribution over similarity

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 130–133,
Vancouver, Canada, August 3 - 4, 2017.

scores. All hyperparameters in our system were empirically tuned for the STS task and shown in Table 1. We implemented our neural network system by using Keras[1] (Chollet, 2015) and TensorFlow[2] (Abadi et al., 2016).

2.1 Pre-process

Several text preprocessing operations were performed before feature engineering:

1) All punctuations are removed.
2) All words are lower-cased.
3) All sentences are tokenized by Natural Language Toolkit (NLTK) (Bird et al., 2009).
4) All words are replaced by pre-trained GloVe word vectors (Common Crawl, 840B tokens) (Pennington et al., 2014). Words that do not exist in the pre-trained embeddings are set to the zero vector.
5) All sentences are padded to a static length $l = 30$ with zero vectors (He et al., 2015a).

Several hand-crafted features are added to enhance the GloVe word vectors:

1) If a word appears in both sentences, add a TRUE flag to the word vector, otherwise, add a FALSE flag.
2) If a word is a number, and the same number appears in the other sentence, add a TRUE flag to the word vector of the matching number in each sentence, otherwise, add a FALSE flag.
3) The part-of-speech (POS) tag of every word according to NLTK is added as a one-hot vector.

2.2 Convolutional neural network (CNN)

Our CNN consists of $n = 300$ one dimensional filters. The length of the filters is set to be the same as the dimension of the enhanced word vectors. The activation function of the CNN is set to be $relu$ (Nair and Hinton, 2010). We did not use any regularization or drop out. Early stopping triggered by model performance on validation data was used to avoid overfitting. The number of layers is set to be 1. We used the same model weights to transfer each of the words in a sentence. Sentence level embeddings are calculated by max pooling (Scherer et al., 2010) over every dimension of the transformed word level embedding.

[1]http://github.com/fchollet/keras
[2]http://github.com/tensorflow/tensorflow

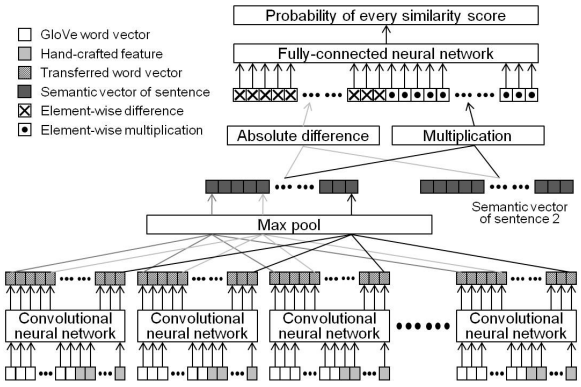

Figure 1: Overview of system

2.3 Comparison of semantic vectors

To calculate the semantic similarity score of two sentences, we generate a semantic difference vector by concatenating the element-wise absolute difference and the element-wise multiplication of the corresponding paired sentence level embeddings. The calculation equation is

$$\vec{SDV} = (|\vec{SV}1 - \vec{SV}2|, \vec{SV}1 \circ \vec{SV}2) \quad (1)$$

Here, $\vec{SDV}$ is the semantic difference vector, $\vec{SV}1$ and $\vec{SV}2$ are the semantic vectors of the two sentences, and $\circ$ is Hadamard product which generate the element-wise multiplication of two semantic vectors.

2.4 Fully-connected neural network (FCNN)

An FCNN is used to transfer the semantic difference vector (600 dimension) to a probability distribution over the six similarity labels used by STS. The number of layers is set to be 2. The first layer uses 300 units with a $tanh$ activation function. The second layer produces the similarity label probability distribution with 6 units combined with a $softmax$ activation function. We train without using regularization or drop out.

3 Experiments and Results

We randomly split all dataset files of SemEval-2012–2015 (Agirre et al., 2012, 2013, 2014, 2015) into ten. We used the preparation of the data from (Baudis et al., 2016). We used 90% of the pairs in each individual dataset file for training and the other 10% for validation. We tested our model in the English dataset of SemEval-2016 (Agirre et al., 2016). Our objective function is the Pearson correlation coefficient computed over each batch. ADAM was used as the gradient descent optimization method. All parameters are set to the values

Table 1: Hyperparameters

Sentence pad length	30
Dimension of GloVe vectors	300
Number of CNN layers	1
Dimension of CNN filters	1
Number of CNN filters	300
Activation function of CNN	$relu$
Initial function of CNN	$he\_uniform$
Number of FCNN layers	2
Dimension of input layer	600
Dimension of first layer	300
Dimension of second layer	6
Activation of first layer	$tanh$
Activation of second layer	$softmax$
Initial function of layers	$he\_uniform$
Optimizer	$ADAM$
Batch size	339
Max epoch	6
Run times	8

suggested by (P.Kingma and Ba, 2015): learning rate is 0.001, $\beta 1$ is 0.9, $\beta 2$ is 0.999, ϵ is 1e-08. $he\_uniform$ (He et al., 2015b) was used as the initial function of layers. We did the experiment 8 times and choose the model that achieved the best performance on the validation dataset. Our system got a Pearson correlation coefficient result of 0.7192±0.0062.

We also used the same model design to take part in all tracks of SemEval-2017. We submitted two runs. One with machine translation (MT) and another without (non-MT). In MT run, we translated all the other languages in the test dataset into English by Google Translate[3] and used the English model to evaluate all similarity scores. For the monolingual tracks, we also tried non-MT run, which means we trained the models directly from the English, Spanish and Arabic data. Here, we independently trained another English model for each run. The difference between English-English performance from MT and non-MT is caused by the random shuffling of data during training.

We also trained another English model with same design to evaluate the STS benchmark dataset (Cer et al., 2017)[4]. We used only the Train part for training and the Dev. part to fine tune. We also run our system without any hand-crafted features. The purely sentence representation system also got a good accuracy. The results are shown in Table 2. Our model achieves 4^{th} place on the STS benchmark[5].

Table 2: Pearson correlation coefficient with the golden standard of 2017 test dataset

Tracks	CNN	Best	Diff.(Rank)
STS 2016	0.7192 ±0.0062	0.7781	0.0589(14^{th})
STS 2017 (MT)			
Primary	0.6598	0.7316	0.0718(3^{rd})
1 AR-AR	0.7130	0.7543	0.0413(6^{th})
2 AR-EN	0.6836	0.7493	0.0657(3^{rd})
3 SP-SP	0.8263	0.8559	0.0296(4^{th})
4a SP-EN	0.7621	0.8302	0.0681(5^{th})
4b SP-EN	0.1483	0.3407	0.1924(7^{th})
5 EN-EN	0.8113	0.8547	0.0434(8^{th})
6 EN-TR	0.6741	0.7706	0.0965(3^{rd})
STS 2017 (non-MT)			
1 AR-AR	0.4373	0.7543	0.3170(15^{th})
3 SP-SP	0.6709	0.8559	0.1850(15^{th})
5 EN-EN	0.8156	0.8547	0.0391(7^{th})
STS benchmark (hand-craft)			
Dev.	0.8343	0.8470	0.0127(4^{th})
Test	0.7842	0.8100	0.0258(4^{th})
STS benchmark (no hand-craft)			
Dev.	0.8236	0.8470	0.0234(4^{th})
Test	0.7833	0.8100	0.0267(4^{th})

4 Discussion

The difference between our model's performance and that of the best participating system are relative small for all tracks except track 4b and 6. We note that the sentences in track 4b are significantly longer than the sentences in other tracks. We speculate that the results of our system in track 4b were pulled down by the decision to use static padding of length 30 within our model.

Another trend that could be observed is that the results of non-MT were likely harmed by the smaller amounts of available training data. We had over 10,000 training pairs for English, but only 1634 pairs in Spanish and 1104 in Arabic. Correspondingly, for our non-MT models, we achieved our best Pearson correlation scores on English with diminished results on Spanish and our worst results on Arabic. Notably, the results obtained by combining our English model with MT to handle Spanish and Arabic were not affected by the

[3] http://translate.google.com

[4] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

[5] As of April 17, 2017

limited amount of training data for these two languages and provided better performance.

5 Conclusion

We proposed a simple convolutional neural network system for the STS task. First, it uses a convolutional neural network to transfer hand-crafted feature enhanced GloVe word vectors. Then, it calculates a semantic vector representation of each sentence by max pooling every dimension of their transformed word vectors. After that, it generates a semantic difference vector between two paired sentences by concatenating their element-wise absolute difference and the element-wise multiplication of their semantic vectors. Next, it uses a fully-connected neural network to transfer the semantic difference vector to a probability distribution over similarity scores.

In spite of the simplicity of our neural network system, the difference in performance between our proposed model and the best performing systems that participated in the STS shared task are less than 0.1 absolute in almost all STS tracks and result in our model being ranked 3rd on primary track of SemEval STS 2017.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. USENIX Association, Berkeley, CA, USA, OSDI'16, pages 265–283.

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*. pages 252–263.

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*. pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. Sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. pages 32–43.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. pages 385–393.

Petr Baudis, Jan Pichl, Tomas Vyskocil, and Jan Sedivy. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127* .

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 1–14. http://www.aclweb.org/anthology/S17-2001.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015a. Multi-perspective sentence similarity modelling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modelling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015b. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*. pages 1532–1543.

Diederik P.Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Dominik Scherer, Andreas C. Muller, and Sven Behnke. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of 20th International Conference on Artificial Neural Networks (ICANN)*. pages 92–101.