A Simple Neural Network for Evaluating Semantic Textual Similarity

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

Abstract

This paper describes a simple neural network system for Semantic Textual Similarity (STS) task. The basic type of the system took part in the STS task of SemEval 2017 and ranked 3^{rd} in the primary track. More variant neural network structures and experiments are explored in this paper. Semantic similarity score between two sentences is calculated by comparing their semantic vectors in our system. Semantic vector of every sentence is generated by max pooling over every dimension of their word vectors. There are mainly two trick points in our system. One is that we trained a convolutional neural network (CNN) to transfer GloVe word vectors to a more proper form for STS task before pooling. Another is that we trained a fully-connected neural network (FCNN) to transfer difference of two semantic vectors to the probability distribution over similarity scores. In spite of the simplicity of our neural network system, the best variant neural network achieved a Pearson correlation coefficient result of 0.7930 on the STS benchmark test dataset and ranked 3^{rd1}.

1 Introduction

Semantic Textual Similarity (STS) is a task of deciding a score that estimating 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 lot of attentions in recent years. STS tasks in SemEval have been held from 2012 to 2017 [Cer *et al.*, 2017]. In order to provide a standard benchmark to compare among meaning representation systems in future years, the organizers of STS tasks created a benchmark dataset in 2017. STS Benchmark[2] comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017 [Agirre *et al.*, 2012; 2013; 2014; 2015; 2016; Cer *et al.*, 2017]. The selection of datasets include text from image captions, news headlines and user forums. Estimating the degree of semantic similarity of two sentences requires a very deep understanding of both sentences. Therefore, methods developed for STS tasks could also be used for a lot of other natural language understanding tasks, such as ”Paraphrasing” tasks, ”Entailment” tasks, ”Answer Sentence Selection” tasks, ”Hypothesis Evidencing” tasks, etc..

Measuring sentence similarity is challenging mainly because of 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 pretraining and distributed representations. We propose a simple neural network system with 5 components:

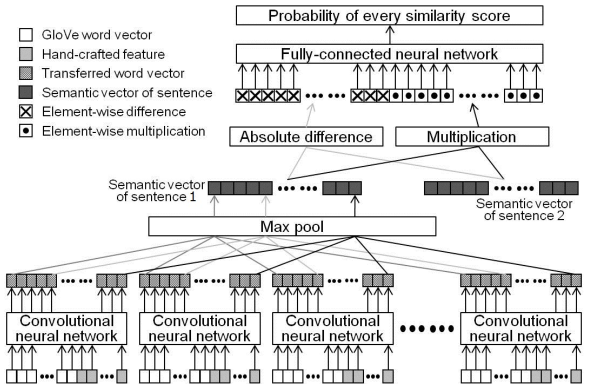
1) Enhance GloVe word vectors in every sentence by adding hand-crafted features.
2) Transfer the enhanced word vectors to a more proper form by convolutional neural network (CNN).
3) Max pooling over every dimension of all word vectors to generate semantic vector.
4) Generate semantic difference vector 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 (FCNN).

2 System Description

Figure 1 provides an overview of our system. The two sentences to be semantically compared are first pre-

In: Proceedings of IJCAI Workshop on Semantic Machine Learning (SML 2017), Aug 19-25 2017, Melbourne, Australia.

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



GloVe word vector
Hand-crafted feature
Transferred word vector
Semantic vector of sentence
Element-wise difference
Element-wise multiplication
Probability of every similarity score
Fully-connected neural network
Absolute difference
Multiplication
Semantic vector of sentence 1
Semantic vector of sentence 2
Max pool
Convolutional neural network

Figure 1: Overview of system

processed as described in subsection 2.1. Then the CNN described in subsection 2.2 transfers the word vectors to a more proper form for each sentence. After that, the processes introduced in subsection 2.3 is adopted to calculate semantic vector and semantic difference vector from the transferred word vectors. Then, an FCNN described in subsection 2.4 transfers the semantic difference vector to a probability distribution over similarity scores. We implemented our neural network system by using Keras[3] [Chollet, 2015] and TensorFlow[4] [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 word vectors are set to the zero vector.
5) All sentences are padded to a static length $l = 30$ with zero vectors [He *et al.*, 2015a].

One hand-crafted feature is 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.

2.2 Convolutional neural network (CNN)

The number of our CNN layers is l. Every layer consists of k_l one dimensional filters. The length of filters are set to be same as the dimension of enhanced word vectors. The activation function of convolutional neural is set to be $tanh$. We did not use any regularization or drop out. Early stopping triggered by model performance on validation data was used to avoid overfitting. We used the same model weights to transfer each of the words in a sentence.

[3]http://github.com/fchollet/keras
[4]http://github.com/tensorflow/tensorflow

Table 1: Hyper parameters

Sentence pad length	30
Dimension of GloVe vectors	300
Number of CNN layers l	1
Number of CNN filters in layer1 k_1	300
Activation function of CNN	$tanh$
Initial function of CNN	$he\_uniform$
Number of FCNN layers n	2
Dimension of input layer	600
Dimension of layer1 m_1	300
Dimension of output layer	6
Activation of layers except output	$tanh$
Activation of output layer	$softmax$
Initial function of layers	$he\_uniform$
Optimizer	$ADAM$
Batch size	1500
Max epoch	25
Run times	8

2.3 Comparison of semantic vectors

The semantic vector of sentence is calculated by max pooling [Scherer *et al.*, 2010] over every dimension of the CNN transferred word vector. 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 two semantic vectors. The calculation equation is

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

Here,$\vec{SDV}$ is the semantic difference vector between two sentences, $\vec{SV}1$ and $\vec{SV}2$ are the semantic vectors of two sentences, $\circ$ 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 to a probability distribution over the six similarity labels used by STS. The number of layers is n. The dimension of every layer is m_n. The activation function of every layer except the last one is $tanh$. The activation function of the last layer is $softmax$. We train without using regularization or drop out.

3 Experiments and Results

The basic type of our neural network system took part in the STS task of SemEval 2017 and ranked 3^{rd} in the primary track [Shao, 2017]. The hyper parameters used in our basic type system were empirically decided for the STS task and shown in Table 1. Our objective function is the Pearson correlation coefficient. $ADAM$ [P.Kingma and Ba, 2015] was used as the gradient descent optimization method. All parameters of the optimizer are set to be followed with the original paper. The learning rate is 0.001,

Table 2: Increasing of dimensions of FCNN

Dimensions of FCNN layer1 m_1	Pearson correlation coefficient results
300	0.778679 ± 0.003508
600	0.776741 ± 0.002711
900	0.778596 ± 0.001876
1200	0.779059 ± 0.003414
1500	0.778852 ± 0.003400
1800	0.779247 ± 0.002261

Table 3: Increasing of filters of CNN

Number of CNN filters in layer1 k_1	Pearson correlation coefficient results
300	0.780586 ± 0.001843
600	0.785420 ± 0.002587
900	0.790137 ± 0.002325
1200	0.791042 ± 0.002557
1500	0.792357 ± 0.002256
1800	0.792580 ± 0.002613

$\beta1$ is 0.9, $\beta2$ is 0.999, ϵ is 1e-08. $he\_uniform$ [He *et al.*, 2015c] was used as the initial function of all layers. The basic model achieved a Pearson correlation coefficient result of 0.778679 ± 0.003508 and ranked 4^{th} on the STS benchmark[5]. We explore more variant neural network structures and experiments in this section.

3.1 Increasing of dimensions of FCNN

We run the experiment using more FCNN dimensions in this subsection. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the dimension of FCNN layer1 m_1. The dimensions of FCNN layer1 m_1 and the Pearson correlation coefficient results are shown in Table 2. Figure 2 shows the average results in every epoch with standard deviation error bar. The highest curve is the Pearson correlation coefficient results on the training data. The curve in the middle is the results on the validation data. The lowest curve is the results on the test data.

3.2 Increasing of filters of CNN

We run the experiment using more CNN filters in this subsection. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the number of CNN filters in layer1 k_1. The number of CNN filters in layer1 k_1 and the Pearson correlation coefficient results are shown in Table 3. Figure 3 shows the average results in every epoch with standard deviation error bar.

3.3 Increasing of layers of FCNN

We run the experiment using more FCNN layers in this subsection. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the number of FCNN layers n and the dimension of FCNN layers m_n. The number of FCNN layers n is set to be 3 in this subsection. The dimensions of FCNN layer$n$ m_n and the Pearson correlation coefficient results are shown in Table 4. The number of filters in CNN layer1 k_1 is set to be 1800 based on the previous experiments. Figure 4 shows the average results in every epoch with standard deviation error bar.

3.4 Increasing of layers of CNN

We run the experiment using more CNN layers in this subsection. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the number of CNN layers l and the number of filters in CNN layers k_l. The number of CNN layers l is set to be 2 in this subsection. The number of filters in CNN layer$l$ k_l and the Pearson correlation coefficient results are shown in Table 5. The dimensions of FCNN layer1 m_1 is set to be 1800 based on the previous experiments. Figure 5 shows the average results in every epoch with standard deviation error bar.

3.5 2 CNN layers with shortcut

We run the experiment using 2 CNN layers in this subsection. We add a shortcut [He *et al.*, 2015b] between input layer and the second layer. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the number of CNN layers l and the number of CNN filters in layers k_l. The number of CNN layers l is set to be 2. The number of CNN filters in layer2 k_2 is set to be 301, same with the dimensions of expanded GloVe word vectors. The number of filters in CNN layers$l$ k_1 and the Pearson correlation coefficient results are shown in Table 6. The dimensions of FCNN layer1 m_1 is set to be 1800 based on the previous experiments. Figure 6 shows the average results in every epoch with standard deviation error bar.

3.6 3 CNN layers with shortcut

We run the experiment using 3 CNN layers in this subsection. We add a shortcut [He *et al.*, 2015b] between the first layer and the third layer. The hyper parameters used in this subsection are same with the basic type system in Table 1 except the number of CNN layers l and the number of CNN filters in layers k_l. The number of CNN layers l is set to be 3. The number of filters in CNN layer3 k_3 is set to be same with the number of filters in CNN layer1 k_1. The number of CNN filters in layers k_l and the Pearson correlation coefficient results are shown in Table 7. The dimensions of FCNN layer1 m_1 is set to be 1800 based on the previous experiments. Figure 7 shows the average results in every epoch with standard deviation error bar. For this experiment, we also tried the model that without the hand-

[5]As of May 26, 2017

Table 4: Increasing of layers of FCNN

Dimensions of FCNN layer1	Dimensions of FCNN layer2	Pearson correlation coefficient results
300	300	0.788331 ± 0.004569
600	600	0.785838 ± 0.003565
900	900	0.789736 ± 0.002546
1200	1200	0.786109 ± 0.003820
1500	1500	0.789013 ± 0.001524
1800	1800	0.782995 ± 0.003396

Table 5: Increasing of layers of CNN

Number of filters in layer1	Number of filters in layer2	Pearson correlation coefficient results
300	301	0.762369 ± 0.002277
600	301	0.765034 ± 0.002445
900	301	0.765966 ± 0.003641
1200	301	0.761183 ± 0.004322
1500	301	0.764604 ± 0.004969
1800	301	0.766178 ± 0.004455

Table 6: 2 CNN layers with shortcut

Number of filters in layer1	Number of filters in layer2	Pearson correlation coefficient results
300	301	0.762030 ± 0.008716
600	301	0.768793 ± 0.003466
900	301	0.767369 ± 0.004021
1200	301	0.768415 ± 0.005799
1500	301	0.769528 ± 0.002299
1800	301	0.770214 ± 0.006707

Table 7: 3 CNN layers with shortcut

Number of filters in layer1	Number of filters in layer2	Pearson correlation coefficient results
1800	300	0.793013 ± 0.002325
1800	600	0.791661 ± 0.002444
1800	900	0.787749 ± 0.003798
1800	1200	0.785493 ± 0.002761
1800	1500	0.785675 ± 0.003413
1800	1800	0.783370 ± 0.004499

crafted feature. The purely sentence representation system achieved an accuracy of 0.788154 ± 0.003412.

4 Discussion

From the results of experiment 1, we can find that increasing the dimensions of FCNN does not have remarkable effect on the accuracy of evaluations. The curves in Figure 2 are almost coincidental. From the results of experiment 2, we can find that increasing the number of filters in CNN layer can improve the accuracy of evaluations. Although the size of training data (5749 records) is not very large, abstracting more features still benefits the evaluation results. By increasing the number of filters in CNN layer, we achieved a Pearson correlation coefficient result of 0.792580 ± 0.002613 and improve the rank from 4^{th} to 3^{rd}.

From the results of experiment 3, we can find that increasing the layer of FCNN is harmful to the evaluation results. But increasing the dimensions of FCNN layer has little effect on the accuracy of evaluations. From the results of experiment 4, we can find that increasing the layer of CNN could significantly pull down the accuracy of evaluations. However, changing the number of filters in CNN layers only changes the learning speed, has little effect on the final accuracy. The structure with smaller number of filters can learn faster.

From the results of experiment 5, we can find that adding a shortcut between input layer and the second CNN layer can slightly improve the accuracy of evaluations. From the results of experiment 6, we can find that adding a shortcut between the first CNN layer and the third CNN layer can get a result that close to the model that has only one CNN layer. Smaller number of filters in the second CNN layer can achieve better accuracy of evaluations. Comparing with the structure that has only one CNN layer, 3 CNN layers with shortcut structure can learn faster. 3 CNN layers with shortcut structure achieved a Pearson correlation coefficient result of 0.793013 ± 0.002325 and that is the best result in all of the variant neural networks.

5 Conclusion

We investigated a simple neural network system for the STS task. All variant models used convolutional neural network to transfer hand-crafted feature enhanced GloVe word vectors to a proper form. Then, the models calculated semantic vectors of sentences by max pooling over every dimension of their transferred word vectors. After that, semantic difference vector between two sentences is generated by concatenating the element-wise absolute difference and element-wise multiplication of their semantic vectors. At last, a fully-connected neural network was used to transfer the semantic difference vector to the probability distribution over similarity scores.

In spite of the simplicity of our neural network system, the basic type ranked 3^{rd} in the primary track of the STS task of SemEval 2017. On the STS benchmark test dataset, the basic model achieved a Pearson correlation coefficient result of 0.778679 ± 0.003508 and ranked 4^{th}. By investigating several variant neural networks in this research, we found that 3 CNN layers with shortcut between the first layer and the third layer structure achieved the best result, a result of 0.793013 ± 0.002325 improved our rank from 4^{th} to 3^{rd}. We also tried purely sentence representation system for this model and the result is 0.788154 ± 0.003412, also ranked 3^{rd}.

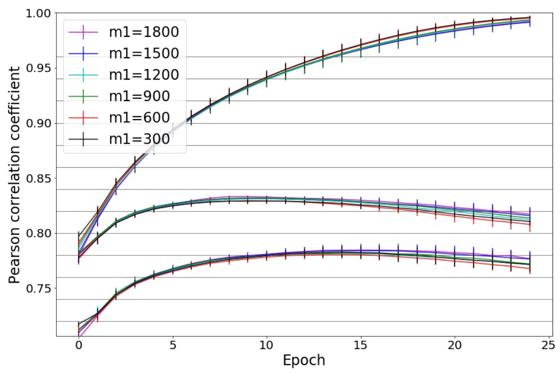


Figure 2: Increasing of dimensions of FCNN

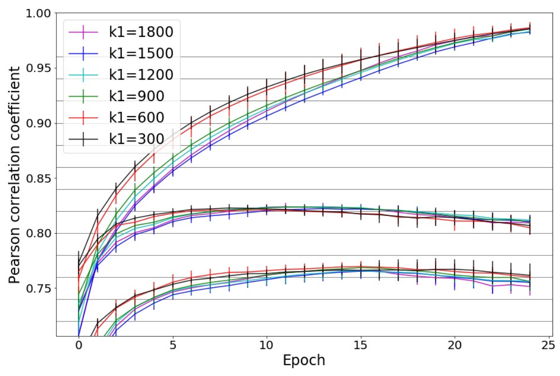


Figure 5: Increasing of layers of CNN

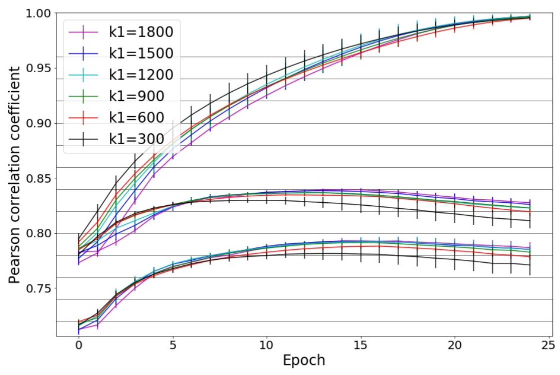


Figure 3: Increasing of filters of CNN

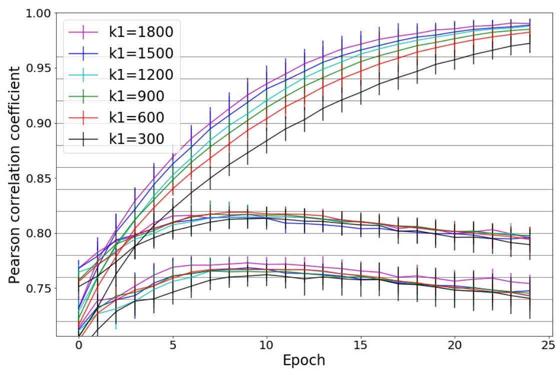


Figure 6: 2 CNN layers with shortcut

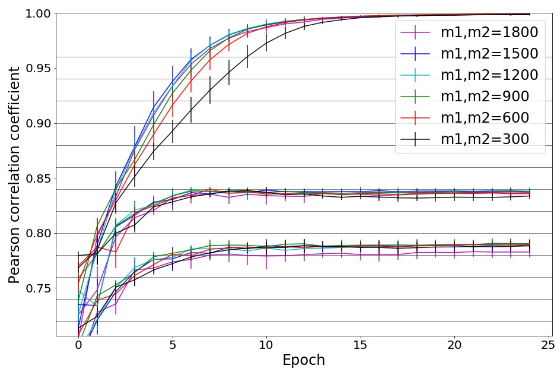


Figure 4: Increasing of layers of FCNN

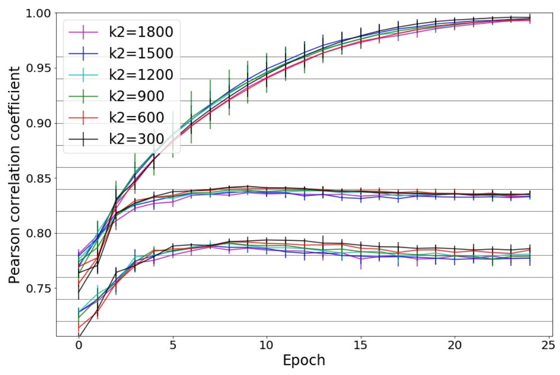


Figure 7: 3 CNN layers with shortcut

References

[Abadi *et al.*, 2016] 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. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[Agirre *et al.*, 2012] Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 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, 2012.

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

[Agirre *et al.*, 2014] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 81–91, 2014.

[Agirre *et al.*, 2015] 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. 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, 2015.

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

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

[Cer *et al.*, 2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics.

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

[He and Lin, 2016] Hua He and Jimmy Lin. 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*, 2016.

[He *et al.*, 2015a] Hua He, Kevin Gimpel, and Jimmy Lin. 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, 2015.

[He *et al.*, 2015b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

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

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

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

[Shao, 2017] Yang Shao. HCTI at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 130–133, Vancouver, Canada, August 2017. Association for Computational Linguistics.