**Siamese architectures applied to**

**sentence semantic similarity comparison tasks**

Michał Sitko

Fac. of Electronics and Information Technology

Warsaw University of Technology

Warsaw, Poland

msitko@mion.elka.pw.edu.pl

Pratiwi Widya Wahyuni

Fac. of Electronics and Information Technology

Warsaw University of Technology

Warsaw, Poland

pwahyuni@mion.elka.pw.edu.pl

*Abstract* — This document presents modifications introduced to Long Short-Term Memory Siamese Neural Network proposed by J. Mueller and A. Thyagarajan [1] at 13th AAAI Conference on Artificial Intelligence in Phoenix, USA in 2016. They applied aforementioned architecture to tasks of assessing syntactic and semantic similarity between sentences. The goal of our research was to check if a variant of the Siamese Network with two untied-variable (untied-weights) streams may yield better performance results. As well as the implementation and the testing process we tried to keep as much close to the baseline as it was possible.

Keywords: Natural Language Processing (NLP), Neural Networks (NN), Siamese Neural Networks, Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Word Embeddings

# Introduction

Computational methods of solving Natural Language Processing (NLP) tasks have been studied for decades. Some claim that the famous Turing Test formulated by himself as a main subject for his seminar, carried out in 1950, can be found the first theoretical work in this field of study. Other think that natural language science originates from works of his predecessors. Despite years of collaborative research of numerous excellent computer scientists, mathematicians and linguists, no complete approach was proposed, which can fully and successfully model domain of any natural language.

However, current level of understanding of NLP already resulted in preparing many astonishing applications.

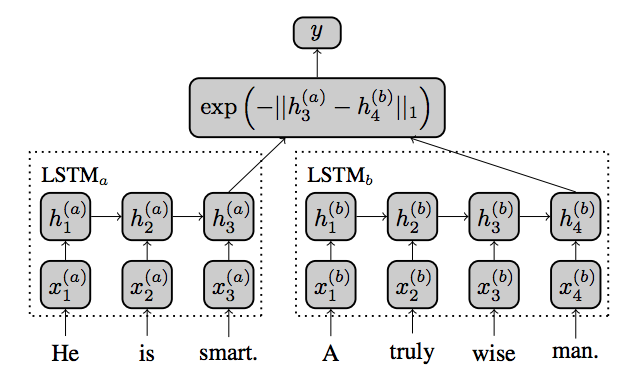
Siamese Networks, which are one of the key concepts of this work were used from almost 30 years. J. Bromley et al. (1994) [13] proposed an application for signature forgery identification. As for early 90s, they achieved surprisingly good performance results. Although whole training process took over a week, their application was able to detect forgeries with >95% accuracy. Modern applications of this family of networks include famous research result of Facebook AI Research group, so called DeepFace – Y. Taigman et al. (2014) [10]. Siamese Network played rather secondary role in their impressive ensemble model. However, its contribution in providing nearly-human level face recognition performance system is inevitable.

Untied (unbalanced) Siamese architectures seems to be also very good at finding similarities between items from different data sets, or even different data domains. These are e.g. comparison of images and text / sentences (image search with text), sketch-based image retrieval – SBIR (translation of one domain example to the other domain class representative) [9].

# Baseline Network Architecture Revised

The baseline model is presented in Figure 1. It consists of two LSTM networks, which operates on sequences of word embeddings. In the model, it is assumed that both network streams share weights, i.e.

|  |  |
| --- | --- |
|  | (1) |



1. The baseline model – sentence pairs are pushed through different LSTM streams of the siamese network. Then, outputted values distance is treated as a semantic similarity measure between inputs.

In the first step of pipeline, pairs of sentences are converted to sequences of word embeddings. Both networks learn a task of encoding input sequences, represented as matrices, into the space of vectors. Dimensions of the vectors generated by the subnetworks are kept the same. For a given pair of output vectors, similarity comparison function is applied. In the baseline solution a simple Manhattan distance metric is used. The model is learned to minimize this distance for pairs of sentences semantically or synthetically similar and maximize for those which tend to be dissimilar.

Our approach introduces two modifications to the proposed architecture:

* The weights of both subnetworks can be untied:

|  |  |
| --- | --- |
|  | (2) |

* Instead of Manhattan based similarity distance – contrastive distance is used. The new metric implies calculating 2nd norm of distance vector (3.1) and then applying it in more complex equation (3.2).

|  |  |
| --- | --- |
|  | (3.1) |
|  | (3.2) |

Results of modifying the net in a described above way are presented in the next section.

# Results

In the last project phase – *train and test,* we prepared a set of slightly different architectures. Each differ from the others in at least one of the following characteristics:

* tied / untied weights,
* configuration of hidden layers – 3 different hidden nodes’ combinations: *50 / 50 / 50, 150 / 100 / 50, 500 / 150 / 50*,
* dropout – no dropout, *0.8* node keep probability.

In order to distinguish different configurations, we introduced a strict naming scheme for the results. This is:

|  |  |  |
| --- | --- | --- |
|  | “*{train\_set}\_{tied\_}\_ 1n-{nodes1}\_1d-{dropout1}\_ 2n-{nodes2}\_2d-{dropout2}”* | (4) |

where: *tied* phrase indicates if a tied-weights network version was used (and is optional), *nodes1* and *nodes2* present hidden nodes configuration in 1st and 2nd stream, *dropout1* and *dropout2* describe dropout keep probability <0, 1> accordingly. Nodes configuration can be any comma separated sequence of numbers *n1,n2,…nl*, e.g. “128,128,36,12”. Each number *ni* refers to the number of hidden nodes in layer *i*. It is assumed that the last number, representing an output layer, for *nodes1* and *nodes2* must be the same. *train\_set* is a code of a data set used for training. We used SICK2014 train set, containing 4427 distinct sentence pairs at first. However, we observed that due to limited amount of samples, trained models tend to overfit. Their performance on SICK2014 set was very high, but it was not on the other validation sets. Therefore, we switched to using SNLI corpus [14], which contains vastly more examples – over 350 000 sentence pairs. Such huge amount of training data would cause us to wait for the process of evaluating one single model very long. To prevent that happening, we implemented early stopping (parameterized with 50 000 iterations limit). The main disadvantage of learning the models with SNLI is that its samples’ similarity is rated using simple, binary – *similar / dissimilar* – metric, while the majority of STS exercises’ samples are rated with continuous or semi-continuous, multilevel metric. Thus, this data set is unlikely to result in models learned to discover partly similar sentence pairs.

The models trained with huge corpuses were then validated against STS exercises proposed at SemEval competitions in years 2014-2017. Each SemEval exercise is build based on relatively small size data set – usually less than 1000 samples, each models different language domain, i.e.: *track5 (2017), headlines (2016)*, *forum answers (2016)*, *questions (2016)*, *postediting* *(2016)*, answers *(2016)*, *plagiarism (2016)*, *headlines (2015)*, *answers (2015)*, *images (2015)*, *beliefs (2015)*, *SICK2014 (2014)*.

In order to compare different evaluations, we used the following comparison metrics:

* Pearson Correlation Coefficient (PCC) – measures level of linear correlation between predicted and expected distances. An equation version, which can be easily vectorized, was used:

|  |  |
| --- | --- |
|  | (5) |

* Mean Squared Error (MSE1),

|  |  |
| --- | --- |
|  | (6) |

* Mean Squared Error (MSE5), which is simply MSE1 multiplied by 5. The metric shows mean squared error in GS comparable domain.

Performance results, averaged over all datasets, achieved by different developed models are outlined in Table 1. Data sets’ wide model average results are outlined in Table 2.

1. Average Model Performance

| ***Model*** | PCC | MSE5 | MSE1 |
| --- | --- | --- | --- |
| snli\_tied\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | 0,3528 | 1,4201 | 0,2840 |
| snli\_tied\_1n-150,100,50\_1d-1.0\_2n-150,100,50\_2d-1.0 | 0,3525 | 1,4315 | 0,2863 |
| snli\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | 0,1424 | 1,4331 | 0,2866 |
| snli\_1n-150,100,50\_1d-1.0\_2n-150,100,50\_2d-1.0 | 0,1105 | 1,4948 | 0,2990 |
| snli\_1n-150,100,50\_1d-0.8\_2n-150,100,50\_2d-0.8 | 0,1032 | 1,5443 | 0,3089 |
| snli\_tied\_1n-500,250,100\_1d-1.0\_2n-500,250,100\_2d-1.0 | 0,3122 | 1,5480 | 0,3096 |
| snli\_tied\_1n-150,100,50\_1d-0.8\_2n-150,100,50\_2d-0.8 | 0,2845 | 1,5637 | 0,3127 |
| snli\_1n-500,250,100\_1d-1.0\_2n-500,250,100\_2d-1.0 | -0,0059 | 1,5843 | 0,3169 |
| sick2014\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | 0,2156 | 1,6536 | 0,3307 |

1. Average Dataset Performance

| ***Dataset*** | PCC | MSE5 | MSE1 |
| --- | --- | --- | --- |
| semeval2017-sts-track5 | 0,3146 | 1,3140 | 0,2628 |
| semeval2015-sts-answers-students | 0,2543 | 1,3147 | 0,2629 |
| semeval2015-sts-images | 0,4118 | 1,3381 | 0,2676 |
| sick2014 | 0,3154 | 1,4206 | 0,2841 |
| semeval2015-sts-belief | 0,1240 | 1,5023 | 0,3005 |
| semeval2015-sts-headlines | 0,1745 | 1,5267 | 0,3053 |
| semeval2015-sts-answers-forums | 0,0249 | 1,5461 | 0,3092 |
| semeval2016-sts-headlines | 0,1888 | 1,5666 | 0,3133 |
| semeval2016-sts-question-question | 0,1435 | 1,5892 | 0,3178 |
| semeval2016-sts-postediting | 0,2206 | 1,6091 | 0,3218 |
| semeval2016-sts-answer-answer | 0,1221 | 1,7057 | 0,3411 |
| semeval2016-sts-plagiarism | 0,2070 | 1,7723 | 0,3545 |

Top performing models together with test set applied are presented in the Table III.

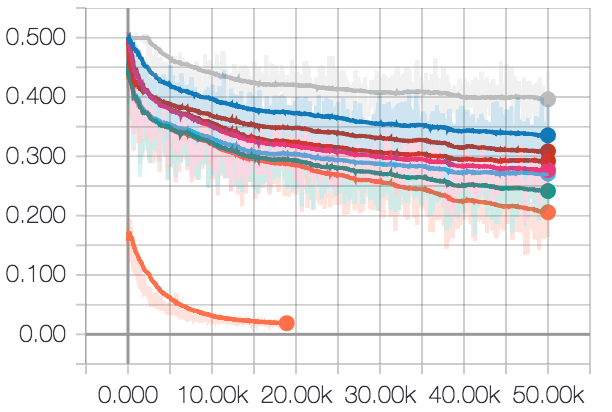
1. Top 10 Best Performing Model – Dataset Combinations

| ***Model*** | Dataset | PCC | MSE5 | MSE1 |
| --- | --- | --- | --- | --- |
| snli\_tied\_1n-150,100,50\_1d-1.0\_2n-150,100,50\_2d-1.0 | semeval2015-sts-images | 0,6672 | 1,1117 | 0,2223 |
| snli\_tied\_1n-150,100,50\_1d-1.0\_2n-150,100,50\_2d-1.0 | semeval2015-sts-answers-students | 0,4231 | 1,1548 | 0,2310 |
| snli\_tied\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | semeval2015-sts-answers-students | 0,4271 | 1,1588 | 0,2318 |
| snli\_tied\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | semeval2015-sts-images | 0,6659 | 1,1710 | 0,2342 |
| snli\_tied\_1n-500,250,100\_1d-1.0\_2n-500,250,100\_2d-1.0 | semeval2017-sts-track5 | 0,4681 | 1,1919 | 0,2384 |
| snli\_tied\_1n-500,250,100\_1d-1.0\_2n-500,250,100\_2d-1.0 | semeval2015-sts-images | 0,6712 | 1,1922 | 0,2384 |
| snli\_tied\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | sick2014 | 0,4686 | 1,2058 | 0,2412 |
| snli\_1n-150,100,50\_1d-1.0\_2n-150,100,50\_2d-1.0 | semeval2017-sts-track5 | 0,2744 | 1,2096 | 0,2419 |
| snli\_tied\_1n-50,50,50\_1d-1.0\_2n-50,50,50\_2d-1.0 | semeval2017-sts-track5 | 0,4720 | 1,2120 | 0,2424 |

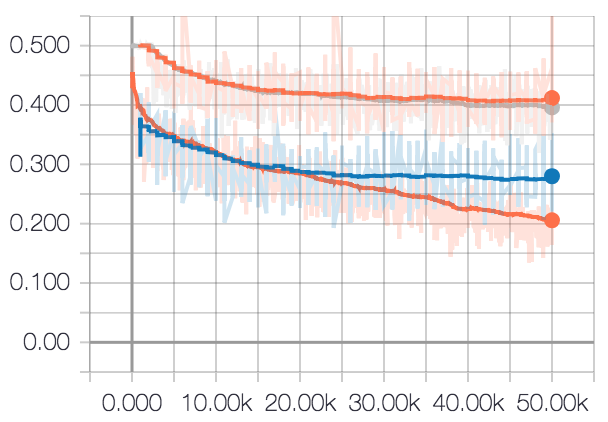
Interesting conclusions can be discovered from model learning performance logs analysis. MSE1 indicator plotted over time, expressed in terms of number of iterations passed, for different models is plotted in Figure 2. The dark-orange line at the bottom is the only model trained on SICK2014 set. It converges very quickly and reports outstanding results in comparison to the others. However, the model does not generalize at all – it is actually discovered as the worst one, according to the Table I.

What’s more, the worst model tends to be the gray one, and the second best – bright-orange one. This is the most complex 500 / 250 / 50 with untied weights and the most complex 500 / 250 / 50 with its weights tied! Generally, models with their weights untied have less step curves than their competitors. This leads us to the conclusion that untying weights do not necessarily improve the performance in STS exercises.

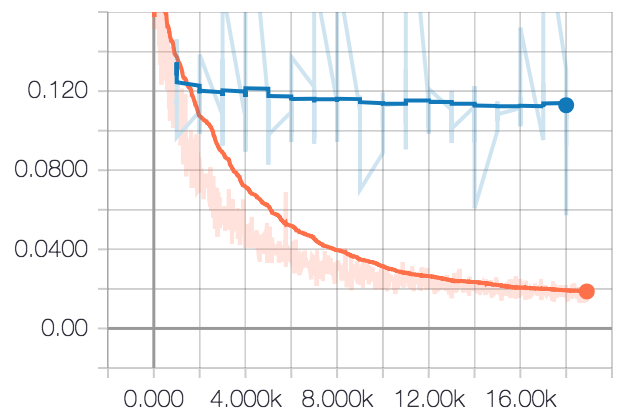
Looking at the degree of the training MSE1 curve for the best models could indicate that those models at 50 000 iteration point can still have an improvement potential. However, plotting extreme models together with their cross-validation sets – as we did in Figure 3 – shows that the MSE1 of the better one is significantly lower than of its cross-validation set and this model is not improving at all after ~25 000th iteration. Effective MSE1 of the best architecture tested is not lower than ~0.275. This is in fact lower than the best result pointed in the baseline result submission – 0.2286, but that model was trained and tested using SICK2014 dataset. It is also unclear whether MSE1 or MSE5 was used. MSE1 results of training tied / 50 / 50 / 50 model on SICK2014 are shown in the Figure 4.



1. Mean Squared Error 1.0 on training set for different architectures over time (iterations)



1. Mean Squared Error 1.0 for the best and the worst performing architectures on train and validation sets over time (iterations)



1. Mean Squared Error 1.0 for tied / 50 / 50 / 50 model trained on SICK2014 over time (iterations)

# Acknowledgements

The solution developed as a background for this paper is publicly available as GitHub code repository at https://github.com/zacateras/deep-siamese-text-similarity. Contrary to the baseline solution [15], its interface was redesigned in order to improve configurability for different network architectures (e.g. unbalanced Siamese networks) and non-standard data set formats (other that native for SemEval exercises).

##### References

1. J. Mueller, A. Thyagarajan, “Siamese Recurrent Architectures for Learning Sentence Similarity”, Proceedings of the Thirteenth AAAI Conference on Artificial Inteligence (AAAI-16), 2016.
2. P. Neculoiu, M. Versteegh, M. Rotaru, “Learning Text Similarity with Siamese Recurrent Networks”, Proceedings of the 1st Workshop on Representation Learning for NLP, 2016 (http://aclweb.org/anthology/W16-1617).
3. H. He, J. Lin, “Pairwise Work Interaction Modelling with Deep Neural Networks for Semantic Similarity Measurement”, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, (http://www.aclweb.org/anthology/N16-1108).
4. A. Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks”, Andrej Karpathy’s Blog, May 2015, (http://karpathy.github.io/2015/05/21/rnn-effectiveness/).
5. J. Colah, “Understanding LSTM Networks”, Colah’s Blog, August 2015 (http://colah.github.io/posts/2015-08-Understanding-LSTMs/).
6. T. Mikolov, K. Chen, G.Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, Google Inc., Mountain View, CA, September 2013 (https://arxiv.org/pdf/1301.3781.pdf).
7. P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, “Enriching Word Vectors with Subwork Information”, Facebook AI Reaserch, June 2017 (https://arxiv.org/pdf/1607.04606.pdf).
8. M. Wang, Z. Lu, J. Zhou, Q. Liu, “Deep Neural Machine Translation with Linear Associative Unit”, Annual Meeting of the Association for Computational Linguistics 2017, May 2017 (https://arxiv.org/pdf/1705.00861.pdf).
9. T. Bui, J. Collomosse, L. Ribeiro, T. Nazare, M. Ponti, “Regression in Deep Learning: Siamese and Triplet Networks”, SibGrapi’17 30th Conference on Graphics, Patterns and Images, Niteroi, Brazil, October 2017 (http://conteudo.icmc.usp.br/pessoas/moacir/p17sibgrapi-tutorial/2017-SIBGRAPI\_Tutorial-DLCV-Part2-Regression-Deep-Learning-Siamese\_and\_Triplet\_Nets.pdf).
10. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”, Facebook AI Research, Menlo Park, CA, USA, 2014 (https://www.cs.toronto.edu/~ranzato/publications/taigman\_cvpr14.pdf).
11. S. Singla, “Experiments with a New Loss Term Added to the Standard Cross entropy”, Medium, September 2017 (https://medium.com/mlreview/experiments-with-a-new-loss-term-added-to-the-standard-cross-entropy-85b080c42446).
12. Y. Wen, K. Zhang, Z. Li, Y. Qiao, “A Discriminative Feature Learning Approach for Deep Face Recognition”, European Conference on Computer Vision 2016, October 2016 (http://ydwen.github.io/papers/WenECCV16.pdf).
13. J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, R. Shah, “Signature Verification usingg a “Siamese” Time Delay Neural Network, AT&T Bell Laboratories, 1994 (https://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf).
14. The Stanford Natural Language Interface (SNLI) Corpus (https://nlp.stanford.edu/projects/snli/).
15. Deep LSTM siamese network for text similarity (https://github.com/dhwajraj/deep-siamese-text-similarity/).