# Survey of Simple Neural Networks in Semantic Textual Similarity Analysis

Derek S. Prijatelj, Jonathan Ventura, and Jugal Kalita

***Abstract*—Learning the semantic resemblance between natural language sentences for computers is a difficult task due to the inherent complexity of natural language. To combat this complexity in learning natural language semantics, a rise in the research of complex architectures for machine learning has become prevalent in the field. It is necessary to establish a lower bound of performance that must be met in order for new complex architectures to be not only novel, but also worth while in terms of implementation. This paper focuses on the specific natural language processing task of determining semantic textual similarity (STS). To construct the lower bound that all complex models must surpass to be deemed practical, this research investigated the performance of relatively simpler models in the STS matching task using SemEval's 2017 STS competition dataset.**

***Index Terms*—semantic matching, semantic textual similarity, compositional distributional semantics**

## I. Introduction

SEMANTICS in natural languages have eluded humans for centuries. Even today, the true meaning of a word can neither be quantified nor computed, but methods have arisen in defining the difference in the meaning between different words and phrases. Distributional semantics plays a key role in this definition of a meaning, where although the actual meaning is unknown, the words or phrases that share the same meaning may be approximated. This is known as the *distributional hypothesis*, where words that share the same context will tend to share similar meaning [1]. While this hypothesis holds true for words, applications of traditional distributional semantics ignore the context of word phrases and longer text fragments [2], [3]. With the introduction of compositional distributional semantics, different methods have been created to represent the meaning of word phrases, and perform better than traditional distributional semantics where context is necessary [3], [4]. There are different methods of representing the semantics of words and phrases, including word embedding and sentence or document embedding. This research concerned itself specifically with the semantic representation of sentences, and compared the different representations in the task of semantic textual similarity matching.

Semantic textual similarity matching is the task of determining the resemblance of the meanings between two sentences. The dataset used for this task is SemEvals' 2017 Semantic Textual Similarity corpus[1][2]. The task specifically is to output a continuous value on the scale from [0, 5] that represents the degree of semantic similarity between two given English sentences, where 0 is no similarity and 5 is complete similarity. In terms of machine learning, this is a regression problem. The 2017 STS corpus contains 1186 English sentence pairs with a corresponding rating and 249 pairs as the test set. The test set has been labeled with the average of multiple human expert ratings that SemEval calls the "golden standard". The distribution of ratings is stated to be as uniform throughout as they could make it, as well as have the same ratios as the training set's ratings.

The models that are examined in this research are simple neural network architectures compared to some of the more complicated models that are popular in recent natural language processing research [5]–[12]. Examining the simple neural network architectures better defines the perspective on creating new architectures for practical applications. If a simple architecture can perform equivalently or better than a newer model, then the new model is simply a new way to accomplish a task using a worse method. To properly establish the threshold that new models must surpass for practical performance in the STS task, simple models such as the perceptron to simple LSTMs and bidirectional LSTMs are evaluated on the STS task. The major components in these models are the pre-trained word vectors, the sentence embeddings, and the comparator of the two sentence embeddings that performs the regression.

## II. Related Work

Both semantic representation and related natural language processing tasks have become more popular due to the introduction of distributional semantics. In the recent past, there have been many improvements, enough to make a comparison of the simplest and latest methods necessary to comprehend the current standard.

### A. Semantic Representation

Mikolov invigorated the interest in distributional semantics with his team's creation of Word2Vec, a means of representing the co-occurrences of words in written text as vectors in a vector space [2]. This method is fairly successful, but by its very nature does not consider the context of larger phrases; this is where compositional distributional semantics was introduced. Stanford developed another method of computing

D. S. Prijatelj is with the Department of Mathematics and Computer Science, Duquesne University, Pittsburgh, PA, 15282 USA e-mail: prijateljd@duq.edu

This research was supported by the National Science Foundation under Grants No. 1359275 and No. 1659788 at the Department of Computer Science, University of Colorado at Colorado Springs, Colorado Springs, CO, 80918 USA

Manuscript received August 3, 2017

[1]http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools
[2]http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page

the distributional semantics of text and this method is known as GloVe. GloVe is similar to Word2Vec in that it computes the co-occurrence frequency of words and creates a vector of a specified dimension to represent a word, but the methods they use are somewhat different [13]. Either may be used for natural language processing tasks depending on preference or performance of the pre-trained word embeddings. In this research, 300 dimensional GloVe word vectors will be used as the initial state of the word vectors.

There are various methods that exist to embed a sentence represented by a list of word vectors. Some of these methods involve the use of neural networks, including, but not limited to, LSTMs and their variations [5], [10], [14], [15]. There have also existed some algorithms to compute sentence representation as well, either similar to methods applied for word embeddings or as extensions of the word embedding methods [3], [4]. Arora et. al 2016 proposed a ”simple but tough-to-beat baseline for sentence embeddings” called the SIF embedding method [16]. SIF involves taking the average of all the word vectors in a sentence and removing the first principal component. Arora et. al have reported it to be a satisfactory baseline and it will be compared to simple neural networks for embedding to determine which is a better baseline for sentence embeddings.

### B. *Semantic Matching*

Different simple semantic matching processes will be examined as the comparator component of the models. These methods will be compared to modified versions of one of the recent developed neural net approaches to semantic matching, the Matrix Vector-LSTM (MV-LSTM) from [5]. The modified versions in this research will replace the similarity tensor with euclidean distance and cosine similarity to establish an understanding of the simplified models' performance. The models used will keep the use of bidirectional LSTMs for learning the sentence embeddings from the paired list of word vectors, and will keep the multi-layered perceptron at the end of the comparator component.

There exist other recent architectures for semantic matching of sentences. One of these newer architectures is the Deep-Fusion LSTM (DF-LSTM) by Penfang et. al 2016 [9]. The DF-LSTM builds upon two LSTM stacks of equal length for reading the two sentences by connecting the individual LSTM's together with a connection between the paired LSTM units' inputs. It's performance rivals that of the MV-LSTM, but is a more complicated version than the LSTM models examined in this paper. There are more complex architectures for semantic matching or similar tasks as well, which supports the need for an established lower bound of practical performance derived from the simpler models [6]–[8].

## III. EXAMINED MODELS

The simple models examined all share the same architecture. Pre-trained word embeddings, a sentence embedding component, and a comparator component. The sentence embedding component takes the list of word vectors that represents a

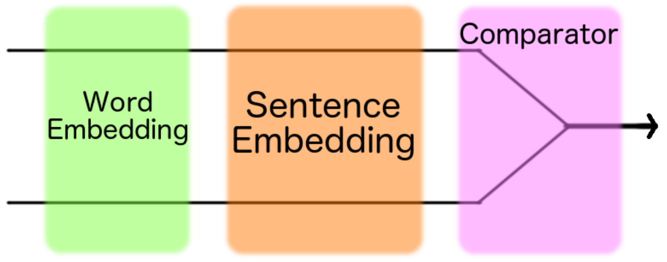

Fig. 1. The overall architecture of the simple models for the STS task. Two string sentences are are the inputs and one float in the range [0, 5] is the output.

sentence and combines them into a single vector that represents the meaning of the original sentence. The comparator component is the part of the model that evaluates the similarity between the two sentence vectors and performs regression to output the sentence pair's similarity score on the continuous inclusive scale from 0 to 5. For all components and individual neural network units of the model, ELU activations are used. The initial weights of each unit are randomly initialized using the He normal distribution [17]. For all models, RMSprop is used as the optimizer with a learning rate of 1e-4. Mean squared error is the loss function for all models as well. The metrics that are calculated are mean squared error, and the Pearson correlation coefficient (PCC), or Pearson R. The SemEval STS competition uses the PCC as the primary metric of a model's performance.

### A. *Pre-Trained Word Vectors*

The models start with the input of two sentences represented by strings. The sentences are embedded into word vectors based on the provided pre-trained word embeddings, which in this case is a set of GloVe word vectors. This specific set of word vectors have 300 dimensions and were pre-trained on 840 billion tokens taken from Common Crawl[3]. Different pre-trained word vectors may be used in-place of this specific pre-trained set. After being embedded into the pre-trained word vectors, the data is randomly shuffled and then sent to the sentence embedding component.

### B. *Sentence Embedding*

The model component responsible for taking a list of word vectors that represent a sentence and embedding them into a single vector to represent the entire sentence. The sentence vector should compress the size of the data that represents the sentence, yet still maintain the important information of the semantics of the sentence.

*1) Smooth Inverse Frequency (SIF):* Arora et. al 2017 proposed their method of sentence embedding called smooth inverse frequency (SIF) as a simple baseline for all sentence representations to surpass [16]. Their method involves taking the mean of all word vectors in a list and removing the first principal component. They found that this simple method of

[3]https://nlp.stanford.edu/projects/glove/

sentence embedding creates satisfactory results. SIF will serve as the simplest method of sentence representation tested in all models in this research.

*2) LSTM:* Sentences are sequences of words where order matters and each word may affect any other's meaning despite their location in the sentence. Given that sentences are sequences, it is only natural to use the version of the recurrent neural network known as the LSTM. The version of the LSTM used throughout model is based on the original from Horchreiter et. al 1997 [18]. This sentence embedding component consists of a single LSTM per sentence with a number of hidden units in parallel equal to that of the word embedding's number of dimensions.

*3) Stacked LSTMs:* The stacked LSTMs' construction is the same as the the single LSTM, except that instead of one LSTM per sentence there are two stacks of LSTMs of equal length. All hyper-parameters are the same otherwise. Various sized stacks of LSTMs are experimented with, including 2, 3, 4, 5, and 10. Multiple LSTMs should be able to capture the kernels of meaning in a sentence. As stated by Palangi et. al 2016, the higher the number of LSTMs in the stack, the better the predicted performance of the sentence embedding [14].

## *C. Comparator*

The comparator examines the two sentence embeddings and performs regression on them to find a continuous value on the inclusive range from 0 to 5. This continuous value indicates the level of similarity between the two sentences, where 0 is no semantic similarity and 5 is complete semantic similarity.

*1) Perceptron:* The simplest of all the comparators, the perceptron with ELU as its activation is used as the regression operation. The weights are initialized at random using the He normal distribution. The outputs from the sentence embeddings are concatenated and sent to a fully connected dense layer, which then connects to a single output node.

*2) LSTM:* In order to learn the relationship between the words in the two sentences, a LSTM takes the concatenated sequence output from the two LSTM sentence embedding components. This single LSTM performs the regression on the two embeddings and learns how the two embeddings relate to one another.

*3) Stacked LSTMs:* Applying the reasoning behind deep LSTM stacks as proposed by Palangi et. al 2016, a stack of LSTMs is used as the comparator of LSTM sentence embeddings. The process is the same as the single LSTM comparator, but instead with a stack of LSTMs. Varying sizes of stacks are used, but match the size of the LSTM stacks in the sentence embedding component.

## *D. Simplified MV-LSTM: L2-LSTM*

Unlike the other simple models that come in parts, this model comes together as a whole. A simplified version of the MV-LSTM from Wan et. al 2016 will also be tested among the simple models [5]. This model matches the MV-LSTM exactly except for the similarity tensor which is replaced with a euclidean distance calculation to compare the similarity to the two sentence embeddings. Bidirectional LSTMs are used for the sentence embeddings and the euclidean distance is followed by a multilayered perceptron with 3 layers that cuts their density in half from the previous layer. The first layer has 5 nodes. This simplified version of the MV-LSTM will be referred to as the L2-LSTM.

## IV. Current Implementation

To analyze the task of semantic textual similarity,

The training data is then given to the neural networks with mean squared error as their loss, due to this being a regression problem. After training, the model is evaluated using the remaining fold as the validation set.

## V. Evaluation Process

The SemEval STS dataset is already provided with training and testing data. Each dataset either has or can be made to have proper training, testing, and validation proportions. Also, Each dataset can easily be evaluated to determine the statistics of the models' performances. Most, if not all, already have their own rules for determining the performance of the model. Each model will be evaluated on each task and will be compared to other models tested in this research as well as those tested on the same datasets in the same standardized fashion. These external models have their information listed on the websites hosting the datasets. The overall best model in each task will be indicated, as well as those with in the subcategories of supervised, semi-supervised, and unsupervised.

## VI. Results

The results indicate that models with a better capacity for memory storage are better suited for solving the STS task optimally. The simplified MV-LSTMs also perform approximately the same as a perceptron, and thus should be discarded from use in practical application for the STS task. However, these are only simplified versions of the MV-LSTM. The actual MV-LSTM could perform better than these less complicated versions.

## *A. LSTM*

The Single LSTM embeddings for both the perceptron comparator and the single LSTM comparator performed worse than any of the models that included a stack of LSTMs. This indicates that the memory of a single LSTM compared to that of a stack of LSTM is unable to learn the semantic kernels of a sentence. This encourages the use of models with increased memory due to their ability to learn important semantic features of a sentence.

## *B. Stacked LSTMs*

The stacked LSTMs performed the best overall with the paired stack of 10 LSTMs for embedding and a perceptron comparator as the best of all LSTM stack embedding and perceptron comparator models. The stack of 2 LSTMs with a stack of 2 LSTMs as the comparator performed the best out of all of the models with a .05 lead over the second place model, the stack of 10 LSTMs and perceptron model. The success

| Simple Models' Mean Performances across 10 K-Fold Cross Validation | | |
|---|---|---|
| Model Name | Pearson R | In-Sample Pearson R |
| 2 LSTM Stack and 2 LSTM Stack Comparator | 0.86075441 | 0.99629578 |
| 10 LSTM Stack and Perceptron | 0.78242820 | 0.90816820 |
| 2 LSTM Stack and Perceptron | 0.75952832 | 0.87565377 |
| 3 LSTM Stack and Perceptron | 0.72353597 | 0.82746079 |
| 4 LSTM Stack and Perceptron | 0.71500020 | 0.82689888 |
| 5 LSTM Stack and Perceptron | 0.45382610 | 0.64646486 |
| 1 LSTM and Perceptron | 0.43011681 | 0.54451700 |
| 1 LSTM and 1 LSTM | 0.41634211 | 0.99024633 |
| L2-LSTM 50 epochs | 0.2740426 | 0.3536559 |
| L2-LSTM 100 epochs | 0.2183386 | 0.3065541 |
| SIF and Perceptron | 0.2211686836 | 0.879533587 |

Fig. 2. The mean Pearson R out-of-sample and in-sample from k-fold cross validation where k = 10. The LSTM and LSTM Stack embeddings were all computed with 50 epochs. The SIF embedding and perceptron comparator were calculated with 100 epochs. The Model Names are ordered by embedding component and comparator, except for the L2-LSTM model which is combined embedding and comparator.

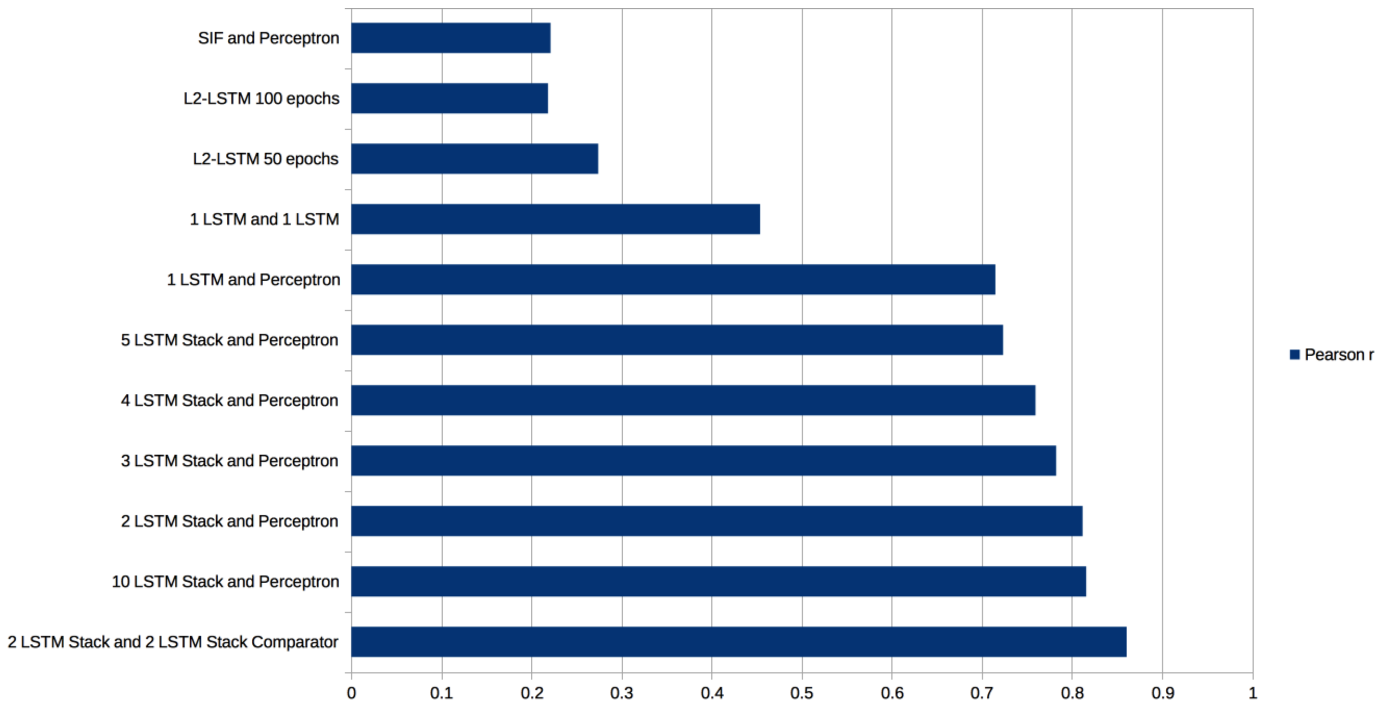

Fig. 3. The mean Pearson R across all test and validation sets in k-fold cross validation where k = 10.

of the LSTM stacks indicates that these models were able to learn kernels of meaning in the sentences and compare them correctly to one another. The quality performance from these models raise the standards for newer, more complex models for the STS task.

### C. *Simplified MV-LSTM: L2-LSTM*

The L2-LSTM performed worse than any of the other models, except for the perceptron when compared to the MV-LSTM with 50 epochs. This indicates that either the bidirectional LSTMs are not suitable for learning the semantics between the two sentences, or the similarity comparison with the euclidean distance is not as effective as the power of the learning the sequences with LSTMs. Given its performance roughly matches that of a perceptron, the L2-LSTM is a model not to be used given its similar performance to, but greater complexity than the perceptron.

## VII. Further Research

The performance of the simplified MV-LSTMs bring into question the adequacy of the original MV-LSTM for the STS task. The next step is to evaluate the performance of the MV-LSTM in the STS task and compare it to that of the LSTM stacks. The results indicated that models with a higher capacity for memory were better suited to learn the semantic representation of the sentence and appropriately compare them. These results encourage further research in memory augmented neural networks for use in learning the semantics of natural languages. Exploring the implementation of more complicated memory augmented neural networks, such as the DNC model created by Graves et. al 2016, is the next step in pursuing better performance in sentence embedding and semantic textual similarity matching [19].

## VIII. Conclusion

The performances of various simple neural network models have been examined on the task of semantic textual similarity matching using SemEval's provided dataset. The model to perform the best with a Pearson correlation of 0.8608, based on the mean k-fold cross validation, is the model where a stack of 2 LSTMs embedded the sentences and were then compared with another stack of 2 LSTMs after concatenating the two sentence embedding stacks' sequences output. This supports the findings that natural language tasks are sequence problems where the elements in the sequence have interconnected relatedness, in which neural networks with memory are better at learning. The large number of LSTMs in the stack also suggests that there exist major groups of meaning in a sentence that can be learned to know the unique meaning of that sentence. This supports the findings with the MV-LSTM. The evaluation of these simple models for semantic textual similarity serves as the lower bound to compare all other models that possess increased complexity in their design. All future researchers should ensure that their new model architectures surpass these lower bounds.

## References

[1] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ArXiv preprint arXiv:1301.3781*, 2013.

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[4] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[5] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[6] Z. Wu, H. Zhu, G. Li, Z. Cui, H. Huang, J. Li, E. Chen, and G. Xu, "An efficient wikipedia semantic matching approach to text document classification," *Information Sciences*, vol. 393, pp. 15–28, 2017.

[7] J. Fu, X. Qiu, and X. Huang, "Convolutional deep neural networks for document-based question answering," in *International Conference on Computer Processing of Oriental Languages*, Springer, 2016, pp. 790–797.

[8] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "Semantic matching by non-linear word transportation for information retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM, 2016, pp. 701–710.

[9] P. Liu, X. Qiu, J. Chen, and X. Huang, "Deep fusion lstms for text semantic matching," in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2016.

[10] P. Liu, X. Qiu, and X. Huang, "Dynamic compositional neural networks over tree structure," *ArXiv preprint arXiv:1705.04153*, 2017.

[11] ——, "Adversarial multi-task learning for text classification," *ArXiv preprint arXiv:1704.05742*, 2017.

[12] ——, "Recurrent neural network for text classification with multi-task learning," *ArXiv preprint arXiv:1605.05101*, 2016.

[13] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[14] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 4, pp. 694–707, 2016.

[15] J. Chen, K. Chen, X. Qiu, Q. Zhang, X. Huang, and Z. Zhang, "Learning word embeddings from intrinsic and extrinsic views," *ArXiv preprint arXiv:1608.05852*, 2016.

[16] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: http://dx.doi.org/10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[19] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.