Neural Networks for Semantic Textual Similarity

Derek S. Prijatelj
Duquesne University
Pittsburgh, PA 15282
prijateljd@duq.edu

Jonathan Ventura
University of Colorado
Colorado Springs, CO 80918
jventura@uccs.edu

Jugal Kalita
University of Colorado
Colorado Springs, CO 80918
jkalita@uccs.edu

Abstract

Complex neural network architectures are being increasingly used to learn to compute the semantic resemblances among natural language texts. 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 worthwhile in terms of implementation. This paper focuses on the specific task of determining semantic textual similarity (STS). We construct a number of models from simple to complex within a framework and report our results. Our findings show that a small number of LSTM stacks with an LSTM stack comparator produces the best results. We use SemEval 2017 STS Competition Dataset for evaluation.

1 Introduction

Scholars have attempted to capture the semantics in natural language texts in a formal manner for centuries. Even today, the true meaning of a word can neither be quantified nor computed, but methods have been developed to express meaning numerically in terms of co-occurrence and association. This is called distributional semantics, and it plays a key role in current approaches to representation of meaning, where although the actual meaning remains unknown, computations can be performed with words or phrases that share the same usage, and therefore, have similar meaning. The theoretical foundation lies in the so-called *distributional hypothesis*, which states that words that share the same context tend to share similar meaning (Harris, 1954). This hypothesis, which is claimed to hold true for words, has been used to obtain vector representations or embeddings for words (Mikolov et al., 2013; Le and Mikolov, 2014). Building on such word embeddings, various methods have been proposed to obtain the meaning of phrases, sentences, paragraphs and whole texts. These include complex linear-algebra based approaches, and more recently a variety of neural network architectures (Le and Mikolov, 2014; Kiros et al., 2015). This research concerns itself specifically with the semantic representation of sentences, and compares 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 "gold standard". The distribution of ratings is stated to be as uniform throughout as it could be, and the ratios of ratings for the test set are similar to the training set's ratings.

The models that are examined in this re-

[1] http://alt.qcri.org/semeval2017/task1/index.php?id=data-and-tools

[2] http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 456–465

search are simple neural network architectures compared to some of the more complicated models that are popular in recent natural language processing research (Wan et al., 2016; Wu et al., 2017b; Fu et al., 2016; Guo et al., 2016; Liu et al., 2016a; Liu et al., 2017b; Liu et al., 2017a; Liu et al., 2016b). Examining the simple neural network architectures better posits a perspective on creating new architectures for practical applications. If a simple architecture can perform equivalently or better than a more complex model being proposed, the new model is simply a new way to accomplish a task using a resource-hungry method. Our simple models use perceptrons to simple LSTMs and bidirectional LSTMs and 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.

2 Related Work

Pursuing better semantic representation for phrases and sentences, and the use of such representation to solve natural language processing tasks have become popular due to the influence of distributional semantics, in particular representations obtained using artificial neural networks.

2.1 Semantic Representation

Mikolov invigorated the interest in distributional semantics with his team's creation of Word2Vec, a means for representing the co-occurrences of words in written text as elements in a vector space (Mikolov et al., 2013). This method is fairly successful, but by its very nature, it creates embeddings for single words (and common short phrases) only. Stanford University developed another method of computing the distributional semantics of words, 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 more linear-algebra based (Pennington et al., 2014). 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 are used as the initial state of the word vectors.

Various methods have been developed to embed a sentence represented by a sequence of words, each with its own vector. Some of these methods involve the use of neural networks, including, but not limited to, LSTMs and their variations (Wan et al., 2016; Palangi et al., 2016; Chen et al., 2016; Liu et al., 2017b). Most of these methods compute sentence representations using methods similar to those applied for word embeddings or as direct extensions of such methods (Kiros et al., 2015; Le and Mikolov, 2014). (Arora et al., 2016) proposed a "simple but tough-to-beat baseline for sentence embeddings" called the SIF embedding method. 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 has been found to provide strong results for many tasks.

2.2 Semantic Matching

A recently developed neural net approach to semantic matching is the Matrix Vector-LSTM (MV-LSTM) (Wan et al., 2016). MV-LSTM finds the paired sentence embeddings by processing the respective lists of word vectors individually through separate bidirectional LSTMs. The resulting sentence embeddings are then compared via a Matrix Vector, otherwise known as a similarity tensor. The important features of the similarity tensor are obtained through k-max pooling and the k-max pools are processed via a multilayered perceptron.

There exist other recent architectures for the semantic matching of sentences and they have been used in the recent SemEval STS 2017 competition. These architectures are listed in Table 3. ECNU accomplishes the STS task by combining traditional natural language processing methods with modern deep learning through the use of an ensemble of classifiers to average the two different parts' scores (Tian et al., 2017). This method attempts to use the best of both methods to overcome the limitations in each other. An ensemble of classifiers is a method of using various classifiers, in this case the traditional and

deep learning methods, and combining their votes via some mathematical method, which in ECNU's case is simple averaging. (Henderson et al., 2017) create the MITRE model, which also uses an ensemble, but instead has 5 systems in their ensemble. Their systems include a simple bag-of-words model, a recurrent convolutional neural network, an enhanced BiLSTM inference model, alignment measures of strings, and the open source TakeLab Semantic Text Similarity System (Šarić et al., 2012). Thus, both the MITRE and ECNU systems use ensembles in an attempt to harness the wisdom of the individual components in their final classification for the STS task.

The BIT model makes use of WordNet to create what is called a "semantic information space (SIS)" (Wu et al., 2017a). SIS attempts to obtain unique meaning of words by establishing the shared words of a certain meaning as a unique information space. WordNet (Miller, 1995) is a human created database that defines how English words share meaning, similar to an extended thesaurus. BIT attempts to leverage the knowledge in WordNet entered by actual humans to create a better system and achieve competitive results. HCTI uses a convolutional neural network in order to handle the STS task (Yang, 2017). The convolutional neural network is simple, yet yields competitive results.

FCICU is a model for solving the STS task through the use of a sense-based and surface-based alignment similarity method coupled with an existing semantic network (Hassan et al., 2017). FCICU uses BabelNet (Navigli and Ponzetto, 2012) and an alignment method to perform multilingual tasks in the SemEval 2017 competition. FCICU is an unsupervised model.

The DT_TEAM's model (Maharjan et al., 2017) uses Support Vector Regression (Smola and Schölkopf, 2004), Linear Regression, and Gradient Boosting Regressor with various features that are carefully selected. DT_TEAM's model uses many methods to generate the features on which their model depends. The DT_TEAM's performance in the English-English STS task was second overall with highly competitive scores. The ITNLPAiKE model (Liu et al., 2017c) also uses a Support Vector Regression model with feature engineering. The features ITNLPAiKE uses are ontology based, word embedding based, corpus based, alignment based and literal based features. The ITNLPAiKE model had relatively competitive results in the English-English STS task, and the authors found that the ontology, word embedding, and alignment based features were the most beneficial features that they tested.

3 Examined Models

We examine a number of models that all share the same architecture: Pre-trained word embeddings, a sentence embedding component, and a comparator component. Figure 1 depicts this shared architecture. The sentence embedding component takes the sequence of word vectors that represents a 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 a continuous inclusive scale from 0 to 5. For all components and individual neural network units of the model, ELU activations (Clevert et al., 2015) are used. The initial weights of each unit are randomly initialized using the He normal distribution (He et al., 2015). For all models, RMSprop (Tieleman and Hinton, 2012) 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.

We examine different sentence embeddings, as well as semantic matching processes as the comparator component of the models. These methods are compared to modified versions of the MV-LSTM. A modified version also replaces the similarity tensor with Euclidean distance to establish an understanding of the simplified models' performance; its results are not reported. One of the models (called the L2-LSTM) uses of bidirectional LSTMs for learning the sentence embeddings from the paired

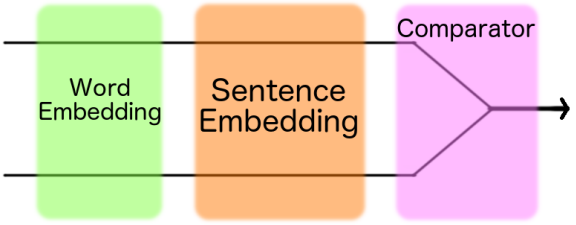

Figure 1: The overall architecture of the simple models for the STS task. Two sentence strings are are the inputs and one float in the range [0, 5] is the output.

list of word vectors. We keep the multi-layered perceptron at the end of the comparator component for most models.

3.1 Pre-Trained Word Vectors

The models start with the input of two sentences represented by strings. The words in the sentences are replaced by word vectors using 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.

3.2 Sentence Embedding

The model component is 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 compresses the size of the data that represents the sentence, but synthesizes important semantic information contained in the sentence.

3.2.1 Smooth Inverse Frequency (SIF)

(Arora et al., 2016) propose a method of sentence embedding called smooth inverse frequency (SIF) as a simple baseline for all sentence representations to surpass. The method involves taking the mean of all word vectors in a list and removing the first principal component. They found that this simple method for sentence embedding creates satisfactory results. The hypothesis is that SIF removes the effect of most common words that occur in documents, and is akin to removing stop words from consideration. SIF serves as the method of sentence representation tested in all models in this research.

The formal mathematical representation of SIF is as follows: $v_s = \frac{1}{s}\sum_{w \in s} \frac{a}{a+p(w)} v_w$ where $a = \frac{1-\alpha}{\alpha Z}$, $v_s = v_s - uu^\top v_s$, where s is the current sentence, w represents a word in the sentence, v_w is the vector representation of the word, α is a hyperparameter, Z is the normalizing constant (the partition function) that is roughly the same in all directions, p represents the estimated probabilities of the words, and u is the calculated first principal component.

3.2.2 LSTM

Sentences are sequences of words where order matters and each word may modify 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 (Hochreiter and Schmidhuber, 1997). 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.

The traditional LSTM used is defined as follows:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_c(c_t)
\end{aligned}
$$

where x_t is the input vector, h_t is the output vector, c_t is the cell state vector, W, U and b are the parameter matrices, and vectors f_t, i_t, and o_t are the forget, input, and output gate vectors respectively. σ represents the activation functions where σ_g is an ELU function and σ_c is the hard sigmoid. Figure 2 depicts this LSTM block architecture.

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

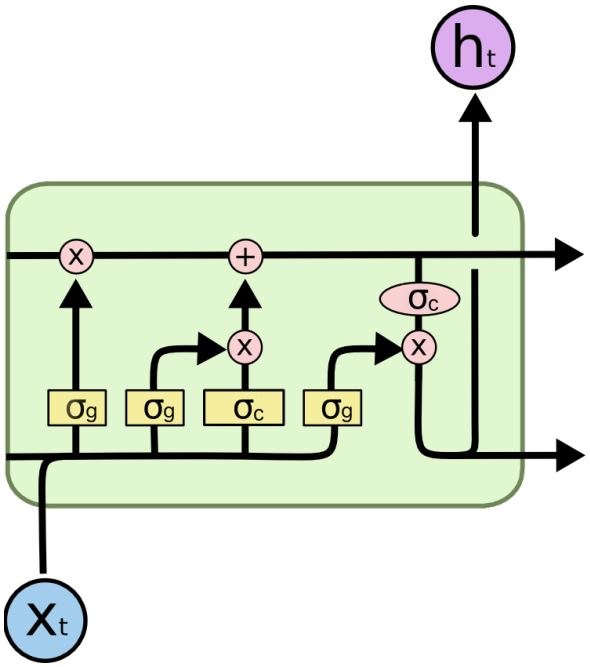

Figure 2: Example of the traditional LSTM architecture with the modified activation functions. The two arrows exiting from the side are the LSTM's recurrent connections. This image was modified from Christopher Olah's original image[5].

3.2.3 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.

A stack of LSTMs is simply multiple LSTMs whose output is the input to the next LSTM in the stack. The version of LSTM stacks used in this research is sequential, meaning that the processed sequence data from the LSTM is saved and outputted to the next LSTM. The final LSTM of the stack outputs a matrix with the dimensions of batch size, and time steps. Figure 3 indicates the input and output dimensions previously described.

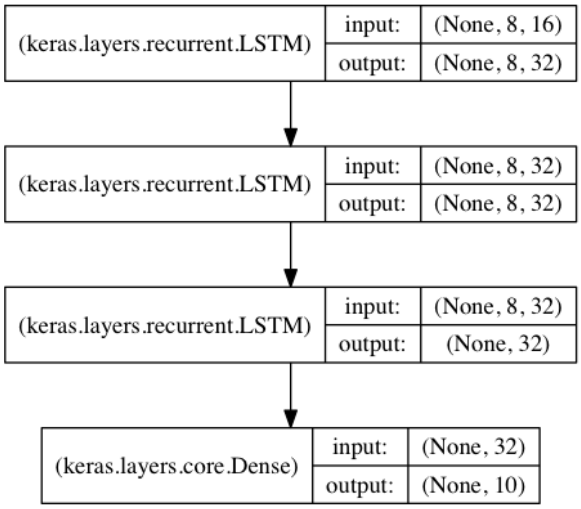

Figure 3: Example of the input and output dimensions of the layers in a small stack of LSTMs as depicted by the Keras API at `https://keras.io/getting-started/sequential-model-guide/`.

3.3 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.

3.3.1 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.

3.3.2 LSTM

In order to learn the relationship between the words in the two sentences, an 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.

3.3.3 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.

3.4 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) is also tested among the simple models. This model matches the MV-LSTM exactly except for the similarity tensor which is replaced with an 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 is referred to as the L2-LSTM.

4 Implementation

The implementation was done using Keras[6] on an Ubuntu system with GPU support. The training data is given to the neural networks with mean squared error as their loss, due to this being a regression problem. The model is evaluated using cross-validation.

5 Evaluation Process

The STS Benchmark comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017. The selection of datasets includes text from image captions, news headlines and user forums. There are a total of 1186 ranked sentence pairs from various domains such as image captions, Twitter news, questions, answers, headlines, plagiarism, and post-editing. Table 1 shows a few pairs of sentences and their gold standard STS rankings from this dataset.

Each model is evaluated on the sentence similarity task. The results of various models are compared in terms of Pearson Correlation Coefficient, as mentioned earlier. The Pearson Correlation Coefficient is as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where n is the number of samples, x_i and y_i are the single samples indexed with i, $\bar{x}$ and $\bar{y}$ are the two samples' respective means.

6 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.

6.1 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 essence of a sentence. This encourages the use of models with increased memory due to their ability to learn important semantic features of a sentence.

6.2 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 of the LSTM stacks indicates that these models are 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.

6.3 Simplified MV-LSTM: L2-LSTM

The L2-LSTM performed worse than any of the other models, except for the perceptron

[6] https://keras.io/

Rank	Sentence 1	Sentence 2
3	In the US, it will depend on the school.	It really depends on the school and the program.
2	I did this one time as well.	I have this habit as well.
5	You do not need to worry.	You don't have to worry.
3	I remained under the banyan tree, exhausted by my daily ritual of dragooning the men every two hours.	I remained under the banyan tree, exhausted by my daily ritual of herding the cats every two hours.
0	You need to read a lot to know what you like and what you don't.	You should tell a good story and sometimes you have to tweak reality/history to do so.
1	If you are not sure how to do it, don't do it at all.	If not, don't do that and spend that time with something you like to do.

Table 1: Example sentence pairs from the SemEval STS 2017 dataset

Simple Models' Mean Pearson R across 10 K-Fold Cross Validation		
Model Name	Pearson R	In-Sample Pearson R
2 LSTM Stack and 2 LSTM Stack	**0.8608**	0.9963
10 LSTM Stack and Perceptron	0.7824	0.9082
2 LSTM Stack and Perceptron	0.7595	0.8757
3 LSTM Stack and Perceptron	0.7235	0.8275
4 LSTM Stack and Perceptron	0.7150	0.8269
5 LSTM Stack and Perceptron	0.4538	0.6465
1 LSTM and Perceptron	0.4301	0.5445
1 LSTM and 1 LSTM	0.4163	0.9902
L2-LSTM 50 epochs	0.2740	0.3537
SIF and Perceptron	0.2214	0.8795
L2-LSTM 100 epochs	0.2183	0.3066

Table 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. In-Sample Pearson R is the Pearson R of the model evaluated on the data used to train the model.

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.

6.4 Comparison with SemEval STS 2017 Results

We compare our results with papers and systems from SemEval STS 2017 (Cer et al., 2017). These papers and systems use Pearson Correlation Coefficient also for evaluation. Table 3 presents the results of the top-performing systems from the SemEval 2017 contest.

We must note that the papers published in SemEval 2017 used the Training, Development, Evaluation datasets whereas we performed 10-fold cross-validation. Thus, the results are not quite comparable, but we feel that cross-validation may be a better way to

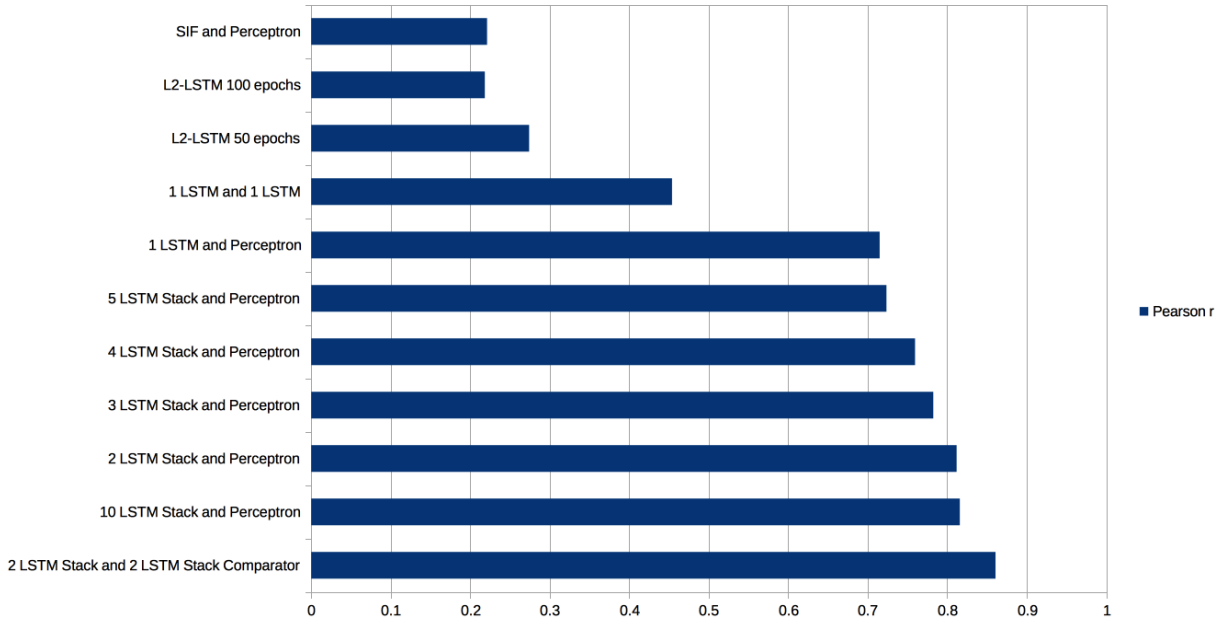

Figure 4: The mean Pearson R across all test and validation sets in k-fold cross validation where k = 10.

Team	Paper	PCC Results (3 or fewer runs)
ECNU	(Tian et al., 2017)	.8515, .8181, .8387
BIT	(Wu et al., 2017a)	.8400, .8161, .8222
HCTI	(Yang, 2017)	.8113, .8156
MITRE	(Henderson et al., 2017)	.8053, .8048
FCICU	(Hassan et al., 2017)	.8272, .8280, .8217
RTV	none	.8541, .8541, **.8547**
DT_TEAM	(Maharjan et al., 2017)	.8536, .8360, .8329
ITNLPAiKE	(Liu et al., 2017c)	.8231, .8231, .8159

Table 3: Results of SemEval STS 2017 Competition in terms of Pearson Correlation Coefficient, as published in (Cer et al., 2017)

evaluate than using a hand-selected Evaluation dataset, but this point remains debatable. In spite of this discrepancy, we claim that one of our methods, namely "2 LSTM Stack and 2 LSTM Stack" shows better performance than the approaches in Table 3.

7 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 indicate that models with a higher capacity for memory are better suited to learn the semantic representation of the sentences 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.

8 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. Our findings also suggest that there exist coherent subgroups of words in a sentence whose meanings can learned and composed to obtain the unique meaning of a sentence. This supports the findings that MV-LSTM also obtains. The evaluation of these simple models for semantic textual similarity serves as the lower bound to compare all other models that have increased complexity in their design. All future researchers should ensure that their new model architectures surpass these lower bounds.

References

[Arora et al.2016] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*.

[Cer et al.2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

[Chen et al.2016] Jifan Chen, Kan Chen, Xipeng Qiu, Qi Zhang, Xuanjing Huang, and Zheng Zhang. 2016. Learning word embeddings from intrinsic and extrinsic views. *arXiv preprint arXiv:1608.05852*.

[Clevert et al.2015] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[Fu et al.2016] Jian Fu, Xipeng Qiu, and Xuanjing Huang. 2016. Convolutional deep neural networks for document-based question answering. In *International Conference on Computer Processing of Oriental Languages*, pages 790–797. Springer.

[Graves et al.2016] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

[Guo et al.2016] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 701–710. ACM.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

[Hassan et al.2017] Basma Hassan, Samir Abdel-Rahman, Reem Bahgat, and Ibrahim Farag. 2017. Fcicu at semeval-2017 task 1: Sense-based language independent semantic textual similarity approach. *Proceedings of SemEval-2017*.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

[Henderson et al.2017] John Henderson, Elizabeth Merkhofer, Laura Strickhart, and Guido Zarrella. 2017. Mitre at semeval-2017 task 1: Simple semantic similarity. *Proceedings of SemEval-2017*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

[Le and Mikolov2014] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

[Liu et al.2016a] Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion lstms for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

[Liu et al.2016b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

[Liu et al.2017a] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

[Liu et al.2017b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017b. Dynamic compositional neural networks over tree structure. *arXiv preprint arXiv:1705.04153*.

[Liu et al.2017c] Wenjie Liu, Chengjie Sun, Lei Lin, and Bingquan Liu. 2017c. Itnlp-aikf at semeval-2017 task 1: Rich features based svr for semantic textual similarity computing. *Proceedings of SemEval-2017*.

[Maharjan et al.2017] Nabin Maharjan, Rajendra Banjade, Dipesh Gautam, Lasang J Tamang, and Vasile Rus. 2017. Dt team at semeval-2017 task 1: Semantic similarity using alignments, sentence-level embeddings and gaussian mixture model output. *Proceedings of SemEval-2017*.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Miller1995] George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Palangi et al.2016] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. 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)*, 24(4):694–707.

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

[Šarić et al.2012] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 441–448. Association for Computational Linguistics.

[Smola and Schölkopf2004] Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[Tian et al.2017] Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. 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.

[Tieleman and Hinton2012] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Wan et al.2016] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Wu et al.2017a] Hao Wu, Heyan Huang, Ping Jian, Yuhang Guo, and Chao Su. 2017a. Bit at semeval-2017 task 1: Using semantic information space to evaluate semantic textual similarity. *Proceedings of SemEval-2017*.

[Wu et al.2017b] Zongda Wu, Hui Zhu, Guiling Li, Zongmin Cui, Hui Huang, Jun Li, Enhong Chen, and Guandong Xu. 2017b. An efficient wikipedia semantic matching approach to text document classification. *Information Sciences*, 393:15–28.

[Yang2017] Shao Yang. 2017. HCTI at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity.