Pairwise Word Interaction Modeling with Deep Neural Networks
for Semantic Similarity Measurement

Hua He\textsuperscript{1} and Jimmy Lin\textsuperscript{2}

\textsuperscript{1} Department of Computer Science, University of Maryland, College Park
\textsuperscript{2} David R. Cheriton School of Computer Science, University of Waterloo

huah@umd.edu, jimmylin@uwaterloo.ca

Abstract

Textual similarity measurement is a challenging problem, as it requires understanding the semantics of input sentences. Most previous neural network models use coarse-grained sentence modeling, which has difficulty capturing fine-grained word-level information for semantic comparisons. As an alternative, we propose to explicitly model pairwise word interactions and present a novel similarity focus mechanism to identify important correspondences for better similarity measurement. Our ideas are implemented in a novel neural network architecture that demonstrates state-of-the-art accuracy on three SemEval tasks and two answer selection tasks.

1 Introduction

Given two pieces of text, measuring their semantic textual similarity (STS) remains a fundamental problem in language research and lies at the core of many language processing tasks, including question answering (Lin, 2007), query ranking (Burges et al., 2005), and paraphrase generation (Xu, 2014).

Traditional NLP approaches, e.g., developing hand-crafted features, suffer from sparsity because of language ambiguity and the limited amount of annotated data available. Neural networks and distributed representations can alleviate such sparsity, thus neural network-based models are widely used by recent systems for the STS problem (He et al., 2015; Tai et al., 2015; Yin and Schütze, 2015).

However, most previous neural network approaches are based on sentence modeling, which first maps each input sentence into a fixed-length vector and then performs comparisons on these representations. Despite its conceptual simplicity, researchers have raised concerns about this approach (Mooney, 2014): Will fine-grained word-level information, which is crucial for similarity measurement, get lost in the coarse-grained sentence representations? Is it really effective to “cram” whole sentence meanings into fixed-length vectors?

In contrast, we focus on capturing fine-grained word-level information directly. Our contribution is twofold: First, instead of using sentence modeling, we propose pairwise word interaction modeling that encourages explicit word context interactions across sentences. This is inspired by our own intuitions of how people recognize textual similarity: given two sentences \textit{sent}_1 and \textit{sent}_2, a careful reader might look for corresponding semantic units, which we operationalize in our pairwise word interaction modeling technique (Sec. 5). Second, based on the pairwise word interactions, we describe a novel similarity focus layer which helps the model selectively identify important word interactions depending on their importance for similarity measurement. Since not all words are created equal, important words that can make more contributions deserve extra “focus” from the model (Sec. 6).

We conducted thorough evaluations on ten test sets from three SemEval STS competitions (Agirre et al., 2012; Marelli et al., 2014; Agirre et al., 2014) and two answer selection tasks (Yang et al., 2015; Wang et al., 2007). We outperform the recent multi-perspective convolutional neural networks of He et al. (2015) and demonstrate state-of-the-art accuracy on all five tasks. In addition, we conducted ablation
studies and visualized our models to show the clear benefits of modeling pairwise word interactions for similarity measurement.

2 Related Work

Feature engineering was the dominant approach in most previous work; different types of sparse features were explored and found useful. For example, \(n\)-gram overlap features at the word and character levels (Madnani et al., 2012; Wan et al., 2006), syntax features (Das and Smith, 2009; Xu et al., 2014), knowledge-based features using WordNet (Fellbaum, 1998; Fern and Stevenson, 2008) and word-alignment features (Sultan et al., 2014).

The recent shift from sparse feature engineering to neural network model engineering has significantly improved accuracy on STS datasets. Most previous work use sentence modeling with a “Siamese” structure (Bromley et al., 1993). For example, Hu et al. (2014) used convolutional neural networks that combine hierarchical structures with layer-by-layer composition and pooling. Tai et al. (2015) and Zhu et al. (2015) concurrently proposed tree-structured long short-term memory networks, which recursively construct sentence representations following their syntactic trees. There are many other examples of neural network-based sentence modeling approaches for the STS problem (Yin and Schütze, 2015; Huang et al., 2013; Andrew et al., 2013; Weston et al., 2011; Socher et al., 2011; Zarrella et al., 2015).

Sentence modeling is coarse-grained by nature. Most recently, despite still using a sentence modeling approach, He et al. (2015) moved toward fine-grained representations by exploiting multiple perspectives of input sentences with different types of convolution filters and pooling, generating a “matrix” representation where rows and columns capture different aspects of the sentence; comparisons over local regions of the representation are then performed. He et al. (2015) achieves highly competitive accuracy, suggesting the usefulness of fine-grained information. However, these multiple perspectives are obtained at the cost of increased model complexity, resulting in slow model training. In this work, we take a different approach by focusing directly on pairwise word interaction modeling.

3 Model Overview

Figure 1 shows our end-to-end model with four major components:

1. Bidirectional Long Short-Term Memory Networks (Bi-LSTMs) (Graves et al., 2005; Graves et al., 2006) are used for context modeling of input sentences, which serves as the basis for all following components (Sec. 4).
2. A novel pairwise word interaction modeling technique encourages direct comparisons between word contexts across sentences (Sec. 5).
3. A novel similarity focus layer helps the model identify important pairwise word interactions across sentences (Sec. 6).
4. A 19-layer deep convolutional neural network (ConvNet) converts the similarity measurement problem into a pattern recognition problem for final classification (Sec. 7).

To our best knowledge this is the first neural network model, a novel hybrid architecture combining Bi-LSTMs and a deep ConvNet, that uses a similarity focus mechanism with selective attention to important pairwise word interactions for the STS problem. Our approach only uses pretrained word embeddings, and unlike several previous neural network models (Yin and Schütze, 2015; Tai et al., 2015), we do not use sparse features, unsupervised model pretraining, syntactic parsers, or external resources like WordNet. We describe details of each component in the following sections.

4 Context Modeling

Different words occurring in similar semantic contexts of respective sentences have a higher chance to contribute to the similarity measurement. We therefore need word context modeling, which serves as a basis for all following components of this work.

LSTM (Hochreiter and Schmidhuber, 1997) is a special variant of Recurrent Neural Networks (Williams and Zipser, 1989). It can capture long-range dependencies and nonlinear dynamics between words, and has been successfully applied to many NLP tasks (Sutskever et al., 2014; Filipova et al., 2015). LSTM has a memory cell that can store information over a long history, as well as three gates that control the flow of information into and out of the memory cell. At time step $t$, given an input $x_t$, previous output $h_{t-1}$, input gate $i_t$, output gate $o_t$ and forget gate $f_t$, $LSTM(x_t, h_{t-1})$ outputs the hidden state $h_t$ based on the equations below:

\[
\begin{align*}
    i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
    f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
    o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\
    u_t &= \tanh(W^u x_t + U^u h_{t-1} + b^u) \\
    c_t &= i_t \cdot u_t + f_t \cdot c_{t-1} \\
    h_t &= o_t \cdot \tanh(c_t) \\
    LSTM(x_t, h_{t-1}) &= h_t
\end{align*}
\]

where $\sigma$ is the logistic sigmoid activation, $W^*$, $U^*$ and $b^*$ are learned weight matrices and biases. LSTMs are better than RNNs for context modeling, in that their memory cells and gating mechanisms handle the vanishing gradients problem in training.

We use bidirectional LSTMs (Bi-LSTMs) for context modeling in this work. Bi-LSTMs consist of two LSTMs that run in parallel in opposite directions: one (forward $LSTM^f$) on the input sequence and the other (backward $LSTM^b$) on the reverse of the sequence. At time step $t$, the Bi-LSTMs hidden state $h_t^{bi}$ is a concatenation of the hidden state $h_t^{for}$ of $LSTM^f$ and the hidden state $h_t^{back}$ of $LSTM^b$, representing the neighbor contexts of input $x_t$ in the sequence. We define the unpack operation below:

\[
h_t^{for}, h_t^{back} = \text{unpack}(h_t^{bi})
\]

Context modeling with Bi-LSTMs allows all the following components to be built over word contexts, rather than over individual words.

5 Pairwise Word Interaction Modeling

From our own intuitions, given two sentences in a STS task, a careful human reader might compare words and phrases across the sentences to establish semantic correspondences and from these infer similarity. Our pairwise word interaction model is inspired by such behavior: whenever the next word of a sentence is read, the model would compare it and its context against all words and their contexts in the other sentence. Figure 2 illustrates this model.

We first define a comparison unit for comparing two hidden states $\overrightarrow{h_1}, \overrightarrow{h_2}$ of Bi-LSTMs.

\[
\begin{align*}
    coU(\overrightarrow{h_1}, \overrightarrow{h_2}) &= \{\cos(\overrightarrow{h_1}, \overrightarrow{h_2}), L_2 \text{Euclid}(\overrightarrow{h_1}, \overrightarrow{h_2}), \\
                                          &\quad \text{DotProduct}(\overrightarrow{h_1}, \overrightarrow{h_2})\}
\end{align*}
\]

Cosine distance ($cos$) measures the distance of two vectors by the angle between them, while $L_2$ Euclidean distance ($L_2 \text{Euclid}$) and dot-product distance (DotProduct) measure magnitude differences. We use three similarity functions for richer measurement.

Algorithm 1 provides details of the modeling process. Given the input $x_t^a \in \text{sent}_a$ at time step $t$ where $a \in \{1, 2\}$, its Bi-LSTMs hidden state $h_t^{at}$ is the concatenation of the forward state $h_t^{for}$ and the
Algorithm 1 Pairwise Word Interaction Modeling

1: Initialize: simCube ∈ R^{13|sent_1|sent_2} to all 1
2: for each time step t = 1...|sent_1| do
3: for each time step s = 1...|sent_2| do
4: h^{for}_{1t}, h^{back}_{1t} = unpack(h^b_{1t})
5: h^{for}_{2t}, h^{back}_{2t} = unpack(h^b_{2t})
6: h^{add}_{1t} = h^{for}_{1t} + h^{back}_{1t}
7: h^{add}_{2t} = h^{for}_{2t} + h^{back}_{2t}
8: simCube[1:3][t][s] = \cos U(h^{b}_{1t}, h^{b}_{2t})
9: simCube[4:6][t][s] = \cos U(h^{for}_{2t}, h^{for}_{2t})
10: simCube[7:9][t][s] = \cos U(h^{back}_{1t}, h^{back}_{1t})
11: simCube[10:12][t][s] = \cos U(h^{add}_{1t}, h^{add}_{2t})
12: end for
13: end for
14: return simCube

backward state h^{back}_{1t}. Algorithm 1 proceeds as follows: it enumerates all word pairs (s,t) across both sentences, then perform comparisons using the \cos U unit four times over: 1) Bi-LSTMs hidden states h^{b}_{1t} and h^{b}_{2t}; 2) forward hidden states h^{for}_{1t} and h^{for}_{2t}; 3) backward hidden states h^{back}_{1t} and h^{back}_{2t}; and 4) the addition of forward and backward hidden states h^{add}_{1t} and h^{add}_{2t}. The output of Algorithm 1 is a similarity cube simCube with size R^{13|sent_1|sent_2}, where |sent_s| is the number of words in the sentence sent_s. The 13 values collected from each word pair (s,t) are: the 12 similarity distances, plus one extra dimension for the padding indicator. Note that all word interactions are modeled over word contexts in Algorithm 1, rather than individual words.

Our pairwise word interaction model shares similarities with recent popular neural attention models (Bahdanau et al., 2014; Rush et al., 2015). However, there are important differences: For example, we do not use attention weight vectors or weighted representations, which are the core of attention models. The other difference is that attention weights are typically interpreted as soft degrees with which the model attends to particular words; in contrast, our word interaction model directly utilizes multiple similarity metrics, and thus is more explicit.

6 Similarity Focus Layer

Since not all words are created equal, important pairwise word interactions between the sentences (Sec. 5) that can better contribute to the similarity measurement deserve more model focus. We therefore develop a similarity focus layer which can identify important word interactions and increase their model weights correspondingly. This similarity focus layer is directly incorporated into our end-to-end model and is placed on top of the pairwise word interaction model, as in Figure 1.

Figure 3 shows an example where each cell of the matrix represents a pairwise word interaction. The similarity focus layer introduces re-weightings to word interactions depending on their importance for similarity measurement. The ones tagged with black dots are considered important, and are given higher weights than those without.

Algorithm 2 shows the forward pass of the similarity focus layer. Its input is the similarity cube simCube (Section 5). Algorithm 2 is designed to incorporate two different aspects of similarity based on cosine (angular) and L2 (magnitude) similarity, thus it has two symmetric components: the first one is based on cosine similarity (Line 5 to Line 13); and the second one is based on L2 similarity (Line 15 to Line 23). We also aim for the goal that similarity values of all found important word in-
Algorithm 2 Forward Pass: Similarity Focus Layer

1: Input: \(\text{simCube} \in \mathbb{R}^{13 \times |\text{sent}_1| \times |\text{sent}_2|}\)
2: Initialize: \(\text{mask} \in \mathbb{R}^{13 \times |\text{sent}_1| \times |\text{sent}_2|}\) to all 0.1
3: Initialize: \(s1tag \in \mathbb{R}^{|\text{sent}_1|}\) to all zeros
4: Initialize: \(s2tag \in \mathbb{R}^{|\text{sent}_2|}\) to all zeros
5: \(\text{sortIndex}_1 = \text{sort}(\text{simCube}[10])\)
6: for each \(id = 1...|\text{sent}_1| + |\text{sent}_2|\) do
7: \(\text{pos}_1, \text{pos}_2 = \text{calcPos}(id, \text{sortIndex}_1)\)
8: if \(s1tag[\text{pos}_1] + s2tag[\text{pos}_2] == 0\) then
9: \(s1tag[\text{pos}_1] = 1\)
10: \(s2tag[\text{pos}_2] = 1\)
11: \(\text{mask}[:][\text{pos}_1][\text{pos}_2] = 1\)
12: end if
13: end for
14: Re-Initialize: \(s1tag, s2tag\) to all zeros
15: \(\text{sortIndex}_2 = \text{sort}(\text{simCube}[11])\)
16: for each \(id = 1...|\text{sent}_1| + |\text{sent}_2|\) do
17: \(\text{pos}_1, \text{pos}_2 = \text{calcPos}(id, \text{sortIndex}_2)\)
18: if \(s1tag[\text{pos}_1] + s2tag[\text{pos}_2] == 0\) then
19: \(s1tag[\text{pos}_1] = 1\)
20: \(s2tag[\text{pos}_2] = 1\)
21: \(\text{mask}[:][\text{pos}_1][\text{pos}_2] = 1\)
22: end if
23: end for
24: \(\text{mask}[13][:][:] = 1\)
25: \(\text{focusCube} = \text{mask} \cdot \text{simCube}\)
26: return \(\text{focusCube} \in \mathbb{R}^{13 \times |\text{sent}_1| \times |\text{sent}_2|}\)

The similarity focus layer is based on the follow-

```markdown
interactions should be maximized. To achieve this, we
sort the similarity values in descending order (Line 5
for cosine, Line 15 for L2). Note channels 10 and
11 of the simCube contain cosine and L2 values,
respectively; the padding indicator is in Line 24.

We start with the cosine part first, then L2. For
each, we check word interaction candidates moving
down the sorted list. Function calcPos is used to
calculate relative sentence positions pos in the
simCube given one interaction pair. We follow the
constraint that no word in both sentences should be
tagged to be important more than once. We in-
crease weights of important word interactions to 1
(in Line 11 based on cosine and Line 21 based on
L2), while unimportant word interactions receive
weights of 0.1 (in Line 2).

We use a mask matrix, mask, to hold the weights of
each. The final output of the similarity focus layer
is a focus-weighted similarity cube focusCube, which is the element-wise multiplication (Line 25)
of the matrix mask and the input simCube.

The similarity focus layer is based on the follow-
ing intuition: given each word in one sentence, we
look for its semantically similar twin in the other
sentence; if found then this word is considered im-
portant, otherwise it contributes to a semantic dif-
ference. Though technically different, this process
shares conceptual similarity with finding translation
equivalences in statistical machine translation (Al-
onaizan et al., 1999).
```

The backward pass of the similarity focus layer is
straightforward: we reuse the mask matrix as gener-
ated in the forward pass and apply the element-wise
multiplication of mask and inflow gradients, then
propagate the resulting gradients backward.

7 Similarity Classification with Deep
Convolutional Neural Networks

The focusCube contains focus-weighted fine-
grained similarity information. In the final model
component we use the focusCube to compute the
final similarity score. If we treat the focusCube as
an “image” with 13 channels, then semantic simi-
larity measurement can be converted into a pattern
recognition (image processing) problem, where we
are looking for patterns of strong pairwise word in-
teractions in the “image”. The stronger the overall
pairwise word interactions are, the higher similarity
the sentence pair will have.

Recent advances from successful systems at
ImageNet competitions (Simonyan and Zisserman,
2014; Szegedy et al., 2015) show that the depth of a
neural network is a critical component for achieving
competitive performance. We therefore use a deep
homogeneous architecture which has repetitive con-
volution and pooling layers.

Our network architecture (Table 1) is composed
of spatial max pooling layers, spatial convolutional
layers (Conv) with a small filter size of \(3 \times 3\) plus
stride 1 and padding 1. We adopt this filter size
because it is the smallest one to capture the space
of left/right, up/down, and center; the padding and
stride is used to preserve the spatial input resolution.
We then use fully-connected layers followed by the
final softmax layer for the output. After each spatial
convolutional layer, a rectified linear units (ReLU)
non-linearity layer (Krizhevsky et al., 2012) is used.

The input to this deep ConvNet is the focusCube,
which does not always have the same size because
Table 1: Deep ConvNet architecture given two padding size configurations for final classification.

<table>
<thead>
<tr>
<th>Deep ConvNet Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Size:</strong> 32 by 32</td>
</tr>
<tr>
<td>Spatial Conv 128: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>Spatial Conv 164: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>Spatial Conv 192: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>Spatial Conv 128: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>Spatial Conv 192: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>Spatial Conv 192: size $3 \times 3$, stride 1, pad 1</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Max Pooling: size $2 \times 2$, stride 2</td>
</tr>
</tbody>
</table>

Table 2: Description of STS2014 test sets.

<table>
<thead>
<tr>
<th>STS2014</th>
<th>Domain</th>
<th>Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>deft-forum</td>
<td>discussion forums</td>
<td>450</td>
</tr>
<tr>
<td>deft-news</td>
<td>news articles</td>
<td>300</td>
</tr>
<tr>
<td>headlines</td>
<td>news headlines</td>
<td>750</td>
</tr>
<tr>
<td>images</td>
<td>image descriptions</td>
<td>750</td>
</tr>
<tr>
<td>OnWN</td>
<td>word sense definitions</td>
<td>750</td>
</tr>
<tr>
<td>tweet-news</td>
<td>social media</td>
<td>750</td>
</tr>
</tbody>
</table>

Total: 3,750

8 Experimental Setup

Datasets. We conducted five separate experiments on ten different datasets: three recent SemEval competitions and two answer selection tasks. Note that the answer selection task, which is to rank candidate answer sentences based on their similarity to the questions, is essentially the similarity measurement problem. The five experiments are as follows:

1. Sentences Involving Compositional Knowledge (SICK) is from Task 1 of the 2014 SemEval competition (Marelli et al., 2014) and consists of 9,927 annotated sentence pairs, with 4,500 for training, 500 as a development set, and 4,927 for testing. Each pair has a relatedness score $\in [1, 5]$ which increases with similarity.

2. Microsoft Video Paraphrase Corpus (MSRVID) is from Task 6 of the 2012 SemEval competition (Agirre et al., 2012) and consists of 1,500 annotated pairs of video descriptions, with half for training and the other half for testing. Each sentence pair has a relatedness score $\in [0, 5]$ which increases with similarity. Following the competition rules, our training data is only drawn from previous STS competitions in 2012 and 2013. We excluded training sentences with lengths longer than the 48 word padding limit, resulting in 7,382 training pairs out of a total of 7,592. Table 2 provides a brief description of the test sets.

3. Task 10 of the 2014 SemEval competition on Semantic Textual Similarity (STS2014) (Agirre et al., 2014) provided six different test sets from different domains. Each pair has a similarity score $\in [0, 5]$ which increases with similarity. Following the competition rules, our training data is only drawn from previous STS competitions in 2012 and 2013. We excluded training sentences with lengths longer than the 48 word padding limit, resulting in 7,382 training pairs out of a total of 7,592. Table 2 provides a brief description of the test sets.

4. The open domain question-answering WikiQA data is from Bing query logs by Yang et al. (2015). We followed the same pre-processing steps as Yang et al. (2015), where questions with no correct candidate answer sentences are excluded and answer sentences are truncated to 40 tokens. The resulting dataset consists of 873 questions with 8,672 question-answer pairs in the training set, 126 questions with 1,130 pairs in the development set, and 243 questions with 2,351 pairs in the test set.

5. The TrecQA dataset (Wang et al., 2007) from the Text Retrieval Conferences has been widely used for the answer selection task during the past decade. To enable direct comparison with previous work, we used the same training, development, and test sets as released by Yao et al. (2013).
The TrecQA data consists of 1,229 questions with 53,417 question-answer pairs in the TRAIN-ALL training set, 82 questions with 1,148 pairs in the development set, and 100 questions with 1,517 pairs in the test set.

**Training.** For experiments on SICK, MSRVID, and STS2014, the training objective is to minimize the KL-divergence loss:

\[
    \text{loss}(\theta) = \frac{1}{n} \sum_{k=1}^{n} KL(f_{\theta}^{k} \| \hat{f}_{\theta}^{k})
\]

(11)

where \( f \) is the ground truth, \( \hat{f}_{\theta} \) is the predicted distribution with model weights \( \theta \), and \( n \) is the number of training examples.

We used a hinge loss for the answer selection task on WikiQA and TrecQA data. The training objective is to minimize the following loss, summed over examples \( \langle x, y_{\text{gold}} \rangle \):

\[
    \text{loss}(\theta, x, y_{\text{gold}}) = \sum_{y' \neq y_{\text{gold}}} \max(0, 1 + f_{\theta}(x, y') - f_{\theta}(x, y_{\text{gold}}))
\]

(12)

where \( y_{\text{gold}} \) is the ground truth label, input \( x \) is the pair of sentences \( x = \{S_1, S_2\} \), \( \theta \) is the model weight vector, and the function \( f_{\theta}(x, y') \) is the output of our model.

In all cases, we performed optimization using RMSProp (Tieleman and Hinton, 2012) with backpropagation (Bottou, 1998), with a learning rate fixed to \( 10^{-4} \).

**Settings.** For the SICK and MSRVID experiments, we used 300-dimension GloVe word embeddings (Pennington et al., 2014). For the STS2014, WikiQA, and TrecQA experiments, we used 300-dimension PARAGRAM-SL999 embeddings from Wieting et al. (2015) and the PARAGRAM-PHRASE embeddings from Wieting et al. (2016), trained on word pairs from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013). We did not update word embeddings in all experiments.

We used the SICK development set for tuning and then applied exactly the same hyperparameters to all ten test sets. For the answer selection task (WikiQA and TrecQA), we used the official tree_eval scorer to compute the metrics Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) and selected the best development model based on MRR for final testing. Our timing experiments were conducted on an Intel Xeon E5-2680 CPU.

Due to sentence length variations, for the SICK and MSRVID data we padded the sentences to 32 words; for the STS2014, WikiQA, and TrecQA data, we padded the sentences to 48 words.

### 9 Results

**SICK Results** (Table 3). Our model outperforms previous neural network models, most of which are based on sentence modeling. The ConvNet work (He et al., 2015) and TreeLSTM work (Tai et al., 2015) achieve comparable accuracy; for example, their difference in Pearson’s \( r \) is only 0.1%. In comparison, our model outperforms both by 1% in Pearson’s \( r \), over 1.1% in Spearman’s \( \rho \), and 2-3% in MSE. Note that we used the same word embeddings, sparse distribution targets, and loss function as in He et al. (2015) and Tai et al. (2015), thereby representing comparable experimental conditions.

**MSRVID Results** (Table 4). Our model outperforms the work of He et al. (2015), which already reports a Pearson’s \( r \) score of over 0.9.

**STS2014 Results** (Table 5). Systems in the competition are ranked by the weighted mean (the of-

<table>
<thead>
<tr>
<th>Model</th>
<th>( r )</th>
<th>( \rho )</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socher et al. (2014)</td>
<td>0.7863</td>
<td>0.7305</td>
<td>0.3983</td>
</tr>
<tr>
<td>Socher et al. (2014)</td>
<td>0.7886</td>
<td>0.7280</td>
<td>0.3859</td>
</tr>
<tr>
<td>Lai and Hockenmaier</td>
<td>0.7993</td>
<td>0.7538</td>
<td>0.3692</td>
</tr>
<tr>
<td>Jimenez et al. (2014)</td>
<td>0.8070</td>
<td>0.7489</td>
<td>0.3550</td>
</tr>
<tr>
<td>Bjerva et al. (2014)</td>
<td>0.8268</td>
<td>0.7721</td>
<td>0.3224</td>
</tr>
<tr>
<td>Zhao et al. (2014)</td>
<td>0.8414</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.8477</td>
<td>0.7921</td>
<td>0.2949</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>0.8522</td>
<td>0.7952</td>
<td>0.2850</td>
</tr>
<tr>
<td>2-layer LSTM</td>
<td>0.8411</td>
<td>0.7849</td>
<td>0.2980</td>
</tr>
<tr>
<td>2-layer Bi-LSTM</td>
<td>0.8488</td>
<td>0.7926</td>
<td>0.2893</td>
</tr>
<tr>
<td>Tai et al. (2015)</td>
<td>0.8491</td>
<td>0.7873</td>
<td>0.2852</td>
</tr>
<tr>
<td>Tai et al. (2015)</td>
<td>0.8676</td>
<td>0.8083</td>
<td>0.2532</td>
</tr>
<tr>
<td>He et al. (2015)</td>
<td>0.8686</td>
<td>0.8047</td>
<td>0.2606</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td><strong>0.8784</strong></td>
<td><strong>0.8199</strong></td>
<td><strong>0.2329</strong></td>
</tr>
</tbody>
</table>

Table 3: Test results on SICK grouped as: (1) RNN variants; (2) SemEval 2014 systems; (3) Sequential LSTM variants; (4) Dependency and constituency tree LSTMs. Evaluation metrics are Pearson’s \( r \), Spearman’s \( \rho \), and mean squared error (MSE). Rows in grey are neural network models.
Table 4: Test results on MSRVID data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pearson’s r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beltagy et al. (2014)</td>
<td>0.8300</td>
</tr>
<tr>
<td>B¨ar et al. (2012)</td>
<td>0.8730</td>
</tr>
<tr>
<td>Šari´c et al. (2012)</td>
<td>0.8803</td>
</tr>
<tr>
<td>He et al. (2015)</td>
<td>0.9090</td>
</tr>
<tr>
<td>This work</td>
<td>0.9112</td>
</tr>
</tbody>
</table>

Table 5: Test results on all six test sets in STS2014. We show results of the top three participating systems at the competition in Pearson’s r scores.

<table>
<thead>
<tr>
<th>STS2014</th>
<th>3rd</th>
<th>2nd</th>
<th>1st</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>deft-forum</td>
<td>0.5305</td>
<td>0.4711</td>
<td>0.4828</td>
<td>0.5684</td>
</tr>
<tr>
<td>deft-news</td>
<td>0.7813</td>
<td>0.7628</td>
<td>0.7657</td>
<td>0.7079</td>
</tr>
<tr>
<td>headlines</td>
<td>0.7837</td>
<td>0.7597</td>
<td>0.7646</td>
<td>0.7551</td>
</tr>
<tr>
<td>image</td>
<td>0.8343</td>
<td>0.8013</td>
<td>0.8214</td>
<td>0.8221</td>
</tr>
<tr>
<td>OnWN</td>
<td>0.8502</td>
<td>0.8745</td>
<td>0.8589</td>
<td>0.8847</td>
</tr>
<tr>
<td>tweetnews</td>
<td>0.6755</td>
<td>0.7793</td>
<td>0.7639</td>
<td>0.7469</td>
</tr>
<tr>
<td>Wt. Mean</td>
<td>0.7549</td>
<td>0.7605</td>
<td>0.7610</td>
<td>0.7666</td>
</tr>
</tbody>
</table>

Table 6: Test results on WikiQA data.

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cui et al. (2005)</td>
<td>0.4271</td>
<td>0.5259</td>
</tr>
<tr>
<td>Wang et al. (2007)</td>
<td>0.6029</td>
<td>0.6852</td>
</tr>
<tr>
<td>Heilman and Smith (2010)</td>
<td>0.6091</td>
<td>0.6917</td>
</tr>
<tr>
<td>Wang and Manning (2010)</td>
<td>0.5951</td>
<td>0.6951</td>
</tr>
<tr>
<td>Yao et al. (2013)</td>
<td>0.6307</td>
<td>0.7477</td>
</tr>
<tr>
<td>Severny and Moschitti (2013)</td>
<td>0.6781</td>
<td>0.7358</td>
</tr>
<tr>
<td>Yih et al. (2013)</td>
<td>0.7092</td>
<td>0.7700</td>
</tr>
<tr>
<td>Wang and Nyberg (2015)</td>
<td>0.7134</td>
<td>0.7913</td>
</tr>
<tr>
<td>Severny and Moschitti (2015)</td>
<td>0.7459</td>
<td>0.8078</td>
</tr>
<tr>
<td>This work</td>
<td>0.7588</td>
<td>0.8219</td>
</tr>
</tbody>
</table>

Table 7: Test results on TrecQA data.

Ablation Studies. Table 8 shows the results of ablation studies on SICK and WikiQA data. We removed or replaced one component at a time from the full system and performed re-training and re-testing. We found large drops when removing the context modeling component, indicating that the context information provided by the Bi-LSTMs is crucial for the following components (e.g., interaction modeling). The use of our similarity focus layer is also beneficial, especially on the WikiQA data. When we replaced the entire similarity focus layer with a random dropout layer ($p = 0.3$), the dropout layer hurts accuracy; this shows the importance of directing the model to focus on important pairwise word interactions, to better capture similarity.

Model Efficiency and Storage. He et al. (2015)’s
Ablation on SICK Data

<table>
<thead>
<tr>
<th></th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Model</td>
<td>0.8784</td>
</tr>
<tr>
<td>Remove context modeling (Sec. 4)</td>
<td>-0.1225</td>
</tr>
<tr>
<td>Remove entire focus layer (Sec. 6)</td>
<td>-0.0083</td>
</tr>
<tr>
<td>Replace entire focus layer with dropout</td>
<td>-0.0314</td>
</tr>
</tbody>
</table>

Ablation on WikiQA Data

<table>
<thead>
<tr>
<th></th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Model</td>
<td>0.7234</td>
</tr>
<tr>
<td>Remove context modeling (Sec. 4)</td>
<td>-0.0990</td>
</tr>
<tr>
<td>Remove entire focus layer (Sec. 6)</td>
<td>-0.0327</td>
</tr>
<tr>
<td>Replace entire focus layer with dropout</td>
<td>-0.0403</td>
</tr>
</tbody>
</table>

Table 8: Ablation studies on SICK and WikiQA data, removing each component separately.

Table 9: Comparison of training efficiency and number of tunable model parameters on SICK data. Timing is the average epoch time in seconds for training on a single CPU thread.

<table>
<thead>
<tr>
<th>Model</th>
<th># of Parameters</th>
<th>Timing (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>He et al. (2015)</td>
<td>10.0 million</td>
<td>2265</td>
</tr>
<tr>
<td>This work</td>
<td>1.7 million</td>
<td>664</td>
</tr>
</tbody>
</table>

ConvNet model uses multiple types of convolution and pooling for sentence modeling. This results in a wide architecture with around 10 million tunable parameters. Our approach only models pairwise word interactions and does not require such a complicated architecture. Compared to that previous work, Table 9 shows that our model is 3.4× faster in training and has 83% fewer tunable parameters.

Visualization. Table 10 visualizes the cosine value channel of the focusCube for pairwise word interactions given two sentence pairs in the SICK test set. Note for easier visualization, the values are multiplied by 10. Darker red areas indicate stronger pairwise word interactions. From these visualizations, we see that our model is able to identify important word pairs (in dark red) and tag them with proper similarity values, which are significantly higher than the ones of their neighboring unimportant pairs. This shows that our model is able to recognize important fine-grained word-level information for better similarity measurement, suggesting the reason why our model performs well.

11 Conclusion

In summary, we developed a novel neural network model based on a hybrid of ConvNet and Bi-LSTMs for the semantic textual similarity measurement problem. Our pairwise word interaction model and the similarity focus layer can better capture fine-grained semantic information, compared to previous sentence modeling approaches that attempt to “cram” all sentence information into a fixed-length vector. We demonstrated the state-of-the-art accuracy of our approach on data from three SemEval competitions and two answer selection tasks.

Acknowledgments

This work was supported by the U.S. National Science Foundation under awards IIS-1218043 and CNS-1405688. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsor. We would like to thank Kevin Gimpel, John Wieting and TS for all the support.

References


Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe.


Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics, 2:207–218.


Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich.


