Abstract

We describe an attention-based convolutional neural network for the English semantic textual similarity (STS) task in the SemEval-2016 competition (Agirre et al., 2016). We develop an attention-based input interaction layer and incorporate it into our multi-perspective convolutional neural network (He et al., 2015), using the PARAGRAM-PHRASE word embeddings (Wieting et al., 2016) trained on paraphrase pairs. Without using any sparse features, our final model outperforms the winning entry in STS2015 when evaluated on the STS2015 data.

1 Introduction

Measuring the semantic textual similarity (STS) of two pieces of text remains a fundamental problem in language research. It lies at the core of many language processing tasks, including paraphrase detection (Xu et al., 2014), question answering (Lin, 2007), and query ranking (Duh, 2009).

The STS problem can be formalized as: given a query sentence $S_1$ and a comparison sentence $S_2$, the task is to compute their semantic similarity in terms of a similarity score $sim(S_1, S_2)$. The SemEval Semantic Textual Similarity tasks (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016) are a popular evaluation venue for the STS problem. Over the years the competitions have made more than 15,000 human annotated sentence pairs publicly available, and have evaluated over 300 system runs.

Traditional approaches are based on hand-crafted feature engineering (Wan et al., 2006; Madnani et al., 2012; Fellbaum, 1998; Fern and Stevenson, 2008; Das and Smith, 2009; Guo and Diab, 2012; Sultan et al., 2014; Kashyap et al., 2014; Lynum et al., 2014). Competitive systems in recent years are mostly based on neural networks (He et al., 2015; Tai et al., 2015; Yin and Schütze, 2015; He and Lin, 2016), which can alleviate data sparseness with pre-training and distributed representations.

In this paper, we extend the multi-perspective convolutional neural network (MPCNN) of He et al. (2015). Most previous neural network models, including the MPCNN, treat input sentences separately, and largely ignore context-sensitive interactions between the input sentences. We address this problem by utilizing an attention mechanism (Bahdanau et al., 2014) to develop an attention-based input interaction layer (Sec. 3). It converts the two independent input sentences into an inter-related sentence pair, which can help the model identify important input words for improved similarity measurement. We also use the strongly-performing PARAGRAM-PHRASE word embeddings (Wieting et al., 2016) (Sec. 4) trained on phrase pairs from the Paraphrase Database (Ganitkevitch et al., 2013).

These components comprise our submission to the SemEval-2016 STS competition (shown in Figure 1): an attention-based multi-perspective convolutional neural network augmented with PARAGRAM-PHRASE word embeddings. We provide details of each component in the following sections. Unlike much previous work in the SemEval competitions (Šarić et al., 2012; Sultan et al., 2014), we do not use sparse features, syntactic parsers, or external resources like WordNet.
Figure 1: Model overview. Input sentences are processed by the attention-based input interaction layer and multi-perspective convolutional sentence model, then compared by the structured similarity measurement layer. The shaded components are our additions to the MPCNN model for the competition.

2 Base Model: Multi-Perspective Convolutional Neural Networks

We use the recently-proposed multi-perspective convolutional neural network model (MPCNN) of He et al. (2015) due to its competitive performance. It consists of two major components:

1. A multi-perspective sentence model for converting a sentence into a representation. A convolutional neural network captures different granularities of information in each sentence using multiple types of convolutional filters, types of pooling, and window sizes.

2. A structured similarity measurement layer with multiple similarity metrics for comparing local regions of sentence representations.

The MPCNN model has a Siamese structure (Bromley et al., 1993), with a multi-perspective sentence model for each of the two input sentences.

Multiple Convolutional Filters. The MPCNN model applies two types of convolutional filters: 1-d per-dimension filters and 2-d holistic filters. The holistic filters operate over sliding windows while considering the full dimensionality of the word embeddings, like typical temporal convolutional filters. The per-dimension filters focus on information at a finer granularity and operate over sliding windows of each dimension of the word embeddings. Per-dimension filters can find and extract information from individual dimensions, while holistic filters can discover broader patterns of contextual information. We use both kinds of filters for a richer representation of the input.

Multiple Window Sizes. The window size denotes how many words are matched by a filter. The MPCNN model uses filters with different window sizes $ws$ in order to capture information at different $n$-gram lengths. We use filters with $ws$ selected from \{1, 2, 3\}, so our filters can find unigrams, bigrams, and trigrams in the input sentences. In addition, to retain the raw information in the input, $ws$ is also set to $\infty$ where pooling layers are directly applied over the entire sentence embedding matrix without the use of convolution layers in-between.

Multiple Pooling Types. For each output vector of a convolutional filter, the MPCNN model converts it to a scalar via a pooling layer. Pooling helps a convolutional model retain the most prominent and prevalent features, which is helpful for robustness across examples. One widely adopted pooling layer is max pooling, which applies a max operation over the input vector and returns the maximum value. In addition to max pooling, The MPCNN model uses two other types of pooling, min and mean, to extract different aspects of the filter matches.

Similarity Measurement Layer. After the sentence models produce representations for each sentence, we use a module that performs comparisons between the two sentence representations to output a final similarity score. One simple way to do this would be to flatten each sentence representation into a vector and then apply a similarity function such as cosine similarity. However, this discards important information because particular regions of the sentence representations come from different underlying sources. Therefore, the MPCNN model performs structured similarity measurements over particular local regions of the sentence representations.

The MPCNN model uses rules to identify local regions whose underlying components are related. These rules consider whether the local regions are: (1) from the same filter type; (2) from the convolutional filter with the same window size $ws$; (3) from the same pooling type; (4) from the same specific filter of the underlying convolution filter type.
Only feature vectors that share at least two of the above are compared. There are two algorithms using three similarity metrics to compare local regions: one works on the output of holistic filters only, while the other uses the outputs of both the holistic and per-dimension filters.

On top of the structured similarity measurement layer, we stack two linear layers with a tanh activation layer in between, followed by a log-softmax layer. More details are provided in He et al. (2015).

3 Attention-Based Input Interaction Layer

The MPCNN model treats input sentences separately with two neural networks in parallel, which ignores the input contextual interaction information. We instead utilize an attention mechanism (Bahdanau et al., 2014) and develop an attention-based interaction layer that converts the two independent input sentences into an inter-related sentence pair.

We incorporate this into the base MPCNN model as the first layer of our system. It is applied over raw word embeddings of input sentences to generate re-weighted word embeddings. The attention-based re-weightings can guide the focus of the MPCNN model onto important input words. That is, words in one sentence that are more relevant to the other sentence receive higher weights.

We first define input sentence representation $S^i \in \mathbb{R}^{\ell_i \times d}$ ($i \in \{0, 1\}$) to be a sequence of $\ell_i$ words, each with a $d$-dimensional word embedding vector. $S^i[a]$ denotes the embedding vector of the $a$-th word in $S^i$. We then define an attention matrix $D \in \mathbb{R}^{\ell_0 \times \ell_1}$. Entry $(a, b)$ in the matrix $D$ represents the pairwise word similarity score between the $a$-th word embedding of $S^0$ and the $b$-th word embedding of $S^1$. The similarity score uses cosine distance:

$$D[a][b] = \text{cosine}(S^0[a], S^1[b])$$

Given the attention matrix $D$, we generate the attention weight vector $A^i \in \mathbb{R}^{\ell_i}$ for input sentence $S^i$ ($i \in \{0, 1\}$). Each entry $A^i[a]$ of the attention weight vector can be viewed as an attention-based relevance score of one word embedding $S^i[a]$ with respect to all word embeddings of the other sentence $S^{1-i}[:i]$. Attention weights $A^i[:i]$ sum to one due to

the softmax normalization:

$$E^0[a] = \sum_b D[a][b]$$
$$E^1[b] = \sum_a D[a][b]$$
$$A^i = \text{softmax}(E^i)$$

We finally define updated embeddings $\text{attenEmb} \in \mathbb{R}^{2d}$ for each word as a concatenation of the original and attention-reweighted word embeddings:

$$\text{attenEmb}^i[a] = \text{concat}(S^i[a], A^i[a] \odot S^i[a])$$

where $\odot$ represents element-wise multiplication.

Our input interaction layer is inspired by recent work that incorporates attention mechanisms into neural networks (Bahdanau et al., 2014; Rush et al., 2015; Yin et al., 2015; Rocktäschel et al., 2016). Many of these add parameters and computational complexity to the model. However, our attention-based input layer is simpler and more efficient. Moreover, we do not introduce any additional parameters, as we simply use cosine distance to create the attention weights. Nevertheless, adding this attention layer improves performance, as we show in Section 5.

4 Word Embeddings

We compare several types of word embeddings to represent the initial sentence matrices ($S^i$). We use the PARAGRAM-SL999 embeddings from Wieting et al. (2015) and the PARAGRAM-PHRASE embeddings from Wieting et al. (2016). These were both constructed from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) by training on noisy paraphrase pairs using a hinge-based loss with negative sampling. However, they were trained on two different types of data.

The PARAGRAM-SL999 embeddings were trained on the lexical section of PPDB, which consists of word pairs only. The PARAGRAM-PHRASE embeddings were trained on the phrasal section of PPDB, which consists of phrase pairs. The representations for the phrases were created by simply averaging word embeddings, which was found to outperform more complicated compositional architectures like LSTMs (Hochreiter and Schmidhuber, 1997).
when evaluated on out-of-domain data.\footnote{For in-domain evaluation, LSTMs outperformed averaging.} The resulting word embeddings yield sentence embeddings (via simple averaging) that perform well across STS tasks without task-specific tuning. Their performance is thought to be due in part to how the vectors for less important words have smaller norms than those for information-bearing words.

## 5 Experiments and Results

### Datasets.
The test data of the SemEval-2016 English STS competition consists of five datasets from different domains. We tokenize all data using Stanford CoreNLP (Manning et al., 2014). Each pair has a similarity score $s \in [0, 5]$ which increases with similarity. We use training data from previous STS competitions (2012 to 2015). Table 1 provides a brief description.

### Experimental Settings.
We largely follow the same experimental settings as He et al. (2015), e.g., we perform optimization with stochastic gradient descent using a fixed learning rate of 0.01. We use the 300-dimensional PARAGRAM-PHRASE XXL word embeddings ($d = 300$).

### Results on STS2016.
We provide results of three runs in Table 2. The three runs are from the same system, but with models of different training epochs.

### Ablation Study on STS2015.
Table 3 shows an ablation study on the STS2015 test sets which consist of 3,000 sentence pairs from five domains. Our training data for the ablation study is from previous test sets in STS2012-2014 following the rules of the STS2015 competition (Agirre et al., 2015). We remove or replace one component at a time from the full system and perform re-training and re-testing.

---

### Acknowledgments
This work was supported by NSF awards IIS-1218043 and CNS-1405688. The views expressed here are those of the authors and do not necessarily reflect those of the sponsor. We thank Mohit Bansal and Karen Livescu for helpful comments and NVIDIA for additional hardware support.

---

### Table 1: Data statistics for STS2016.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>answer-answer</td>
<td>254</td>
</tr>
<tr>
<td>headlines</td>
<td>249</td>
</tr>
<tr>
<td>plagiarism</td>
<td>230</td>
</tr>
<tr>
<td>postediting</td>
<td>244</td>
</tr>
<tr>
<td>question-question</td>
<td>209</td>
</tr>
<tr>
<td>Test Total</td>
<td>1,186</td>
</tr>
<tr>
<td>Train Total</td>
<td>STS2012-2015</td>
</tr>
</tbody>
</table>

### Table 2: Pearson’s $r$ on all five test sets. We show our three submission runs.

<table>
<thead>
<tr>
<th>Domain</th>
<th>1st run</th>
<th>2nd run</th>
<th>3rd run</th>
</tr>
</thead>
<tbody>
<tr>
<td>answer-answer</td>
<td>0.6607</td>
<td>0.6443</td>
<td>0.6432</td>
</tr>
<tr>
<td>headlines</td>
<td>0.7946</td>
<td>0.7871</td>
<td>0.7780</td>
</tr>
<tr>
<td>plagiarism</td>
<td>0.8154</td>
<td>0.7989</td>
<td>0.7816</td>
</tr>
<tr>
<td>postediting</td>
<td>0.8094</td>
<td>0.7934</td>
<td>0.7779</td>
</tr>
<tr>
<td>question-question</td>
<td>0.6187</td>
<td>0.5947</td>
<td>0.5586</td>
</tr>
<tr>
<td>Wt. Mean</td>
<td>0.7420</td>
<td>0.7262</td>
<td>0.7111</td>
</tr>
</tbody>
</table>

### Table 3: Ablation study on STS2015 test data.

<table>
<thead>
<tr>
<th></th>
<th>Pearson’s $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full System</td>
<td>0.8040</td>
</tr>
<tr>
<td>- Remove the attention layer (Sec. 3)</td>
<td>0.7948</td>
</tr>
<tr>
<td>- Replace PARAGRAM-PHRASE with GloVe (Sec. 4)</td>
<td>0.7622</td>
</tr>
<tr>
<td>- Replace PARAGRAM-PHRASE with PARAGRAM-SL999</td>
<td>0.7721</td>
</tr>
<tr>
<td>Winning System of STS2015</td>
<td>0.8015</td>
</tr>
</tbody>
</table>

We observe a significant drop when the attention-based input interaction layer (Sec. 3) is removed. We also find that the PARAGRAM-PHRASE word embeddings are highly beneficial, outperforming both GloVe word embeddings (Pennington et al., 2014) and the PARAGRAM-SL999 embeddings of Wieting et al. (2015). Our full system performs favorably compared to the winning system (Sultan et al., 2015) at the STS2015 SemEval competition.

### 6 Conclusion

Our submission to the SemEval-2016 STS competition uses our multi-perspective convolutional neural network model as the base model. We develop an attention-based input interaction layer to guide the convolutional neural network to focus on the most important input words. We further improve performance by using the PARAGRAM-PHRASE word embeddings, yielding a result on the 2015 test data that surpasses that of the top system from STS2015.
References


