Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks

Hua He,[1] **Kevin Gimpel,**[2] **and Jimmy Lin**[3]

[1] Department of Computer Science, University of Maryland, College Park
[2] Toyota Technological Institute at Chicago
[3] David R. Cheriton School of Computer Science, University of Waterloo

huah@cs.umd.edu, kgimpel@ttic.edu, jimmylin@uwaterloo.ca

Abstract

Modeling sentence similarity is complicated by the ambiguity and variability of linguistic expression. To cope with these challenges, we propose a model for comparing sentences that uses a multiplicity of perspectives. We first model each sentence using a convolutional neural network that extracts features at multiple levels of granularity and uses multiple types of pooling. We then compare our sentence representations at several granularities using multiple similarity metrics. We apply our model to three tasks, including the Microsoft Research paraphrase identification task and two SemEval semantic textual similarity tasks. We obtain strong performance on all tasks, rivaling or exceeding the state of the art without using external resources such as WordNet or parsers.

1 Introduction

Measuring the semantic relatedness of two pieces of text is a fundamental problem in language processing tasks like plagiarism detection, query ranking, and question answering. In this paper, we address the sentence similarity measurement problem: given a query sentence S_1 and a comparison sentence S_2, the task is to compute their similarity in terms of a score $sim(S_1, S_2)$. This similarity score can be used within a system that determines whether two sentences are paraphrases, e.g., by comparing it to a threshold.

Measuring sentence similarity is challenging because of the variability of linguistic expression and the limited amount of annotated training data. This makes it difficult to use sparse, hand-crafted features as in conventional approaches in NLP. Recent successes in sentence similarity have been obtained by using neural networks (Tai et al., 2015; Yin and Schütze, 2015). Our approach is also based on neural networks: we propose a modular functional architecture with two components, sentence modeling and similarity measurement.

For sentence modeling, we use a convolutional neural network featuring convolution filters with multiple granularities and window sizes, followed by multiple types of pooling. We experiment with two types of word embeddings as well as part-of-speech tag embeddings (Sec. 4). For similarity measurement, we compare pairs of local regions of the sentence representations, using multiple distance functions: cosine distance, Euclidean distance, and element-wise difference (Sec. 5).

We demonstrate state-of-the-art performance on two SemEval semantic relatedness tasks (Agirre et al., 2012; Marelli et al., 2014), and highly competitive performance on the Microsoft Research paraphrase (MSRP) identification task (Dolan et al., 2004). On the SemEval-2014 task, we match the state-of-the-art dependency tree Long Short-Term Memory (LSTM) neural networks of Tai et al. (2015) without using parsers or part-of-speech taggers. On the MSRP task, we outperform the recently-proposed convolutional neural network model of Yin and Schütze (2015) without any pretraining. In addition, we perform ablation experiments to show the contribution of our modeling decisions for all three datasets, demonstrating clear benefits from our use of multiple perspectives both in sentence modeling and structured similarity measurement.

2 Related Work

Most previous work on modeling sentence similarity has focused on feature engineering. Several types of sparse features have been found useful, including: (1) string-based, including n-gram overlap features on both the word and character levels (Wan et al., 2006) and features based on machine translation evaluation metrics (Madnani

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1576–1586,
Lisbon, Portugal, 17-21 September 2015.

et al., 2012); (2) knowledge-based, using external lexical resources such as WordNet (Fellbaum, 1998; Fern and Stevenson, 2008); (3) syntax-based, e.g., modeling divergence of dependency syntax between the two sentences (Das and Smith, 2009); (4) corpus-based, using distributional models such as latent semantic analysis to obtain features (Hassan, 2011; Guo and Diab, 2012).

Several strongly-performing approaches used system combination (Das and Smith, 2009; Madnani et al., 2012) or multi-task learning. Xu et al. (2014) developed a feature-rich multi-instance learning model that jointly learns paraphrase relations between word and sentence pairs.

Recent work has moved away from hand-crafted features and towards modeling with distributed representations and neural network architectures. Collobert and Weston (2008) used convolutional neural networks in a multitask setting, where their model is trained jointly for multiple NLP tasks with shared weights. Kalchbrenner et al. (2014) introduced a convolutional neural network for sentence modeling that uses dynamic k-max pooling to better model inputs of varying sizes. Kim (2014) proposed several modifications to the convolutional neural network architecture of Collobert and Weston (2008), including the use of both fixed and learned word vectors and varying window sizes of the convolution filters.

For the MSRP task, Socher et al. (2011) used a recursive neural network to model each sentence, recursively computing the representation for the sentence from the representations of its constituents in a binarized constituent parse. Ji and Eisenstein (2013) used matrix factorization techniques to obtain sentence representations, and combined them with fine-tuned sparse features using an SVM classifier for similarity prediction. Both Socher et al. and Ji and Eisenstein incorporated sparse features to improve performance, which we do not use in this work.

Hu et al. (2014) used convolutional neural networks that combine hierarchical sentence modeling with layer-by-layer composition and pooling. While they performed comparisons directly over entire sentence representations, we instead develop a structured similarity measurement layer to compare local regions. A variety of other neural network models have been proposed for similarity tasks (Weston et al., 2011; Huang et al., 2013; Andrew et al., 2013; Bromley et al., 1993).

Most recently, Tai et al. (2015) and Zhu et al. (2015) concurrently proposed a tree-based LSTM neural network architecture for sentence modeling. Unlike them, we do not use syntactic parsers, yet our performance matches Tai et al. (2015) on the similarity task. This result is appealing because high-quality parsers are difficult to obtain for low-resource languages or specialized domains. Yin and Schütze (2015) concurrently developed a convolutional neural network architecture for paraphrase identification, which we compare to in our experiments. Their best results rely on an unsupervised pretraining step, which we do not need to match their performance.

Our model architecture differs from previous work in several ways. We exploit multiple perspectives of input sentences in order to maximize information utilization and perform structured comparisons over particular regions of the sentence representations. We now proceed to describe our model in detail, and we compare to the above related work in our experimental evaluation.

3 Model Overview

Modeling textual similarity is complicated by the ambiguity and variability of linguistic expression. We designed a model with these phenomena in mind, exploiting multiple types of input which are processed by multiple types of convolution and pooling. Our similarity architecture likewise uses multiple similarity functions.

To summarize, our model (shown in Figure 1) consists of two main components:

1. A **sentence model** for converting a sentence into a representation for similarity measurement; we use a convolutional neural network architecture with multiple types of convolution and pooling in order to capture different granularities of information in the inputs.
2. A **similarity measurement layer** using multiple similarity measurements, which compare local regions of the sentence representations from the sentence model.

Our model has a "Siamese" structure (Bromley et al., 1993) with two subnetworks each processing a sentence in parallel. The subnetworks share all of their weights, and are joined by the similarity measurement layer, then followed by a fully connected layer for similarity score output.

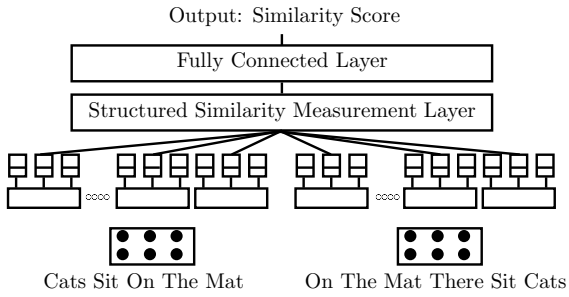

Figure 1: Model overview. Two input sentences (on the bottom) are processed in parallel by identical neural networks, outputting sentence representations. The sentence representations are compared by the structured similarity measurement layer. The similarity features are then passed to a fully-connected layer for computing the similarity score (top).

Importantly, we do not require resources like WordNet or syntactic parsers for the language of interest; we only use optional part-of-speech tags and pretrained word embeddings. The main difference from prior work lies in our use of multiple types of convolution, pooling, and structured similarity measurement over local regions. We show later in our experiments that the bulk of our performance comes from this use of multiple "perspectives" of the input sentences.

We describe our sentence model in Section 4 and our similarity measurement layer in Section 5.

4 Sentence Modeling

In this section we describe our convolutional neural network for modeling each sentence. We use two types of convolution filters defined on different perspectives of the input (Sec. 4.1), and also use multiple types of pooling (Sec. 4.2).

Our inputs are streams of tokens, which can be interpreted as a temporal sequence where nearby words are likely to be correlated. Let $sent \in \mathbf{R}^{len \times Dim}$ be a sequence of len input words represented by Dim-dimensional word embeddings, where $sent_i \in \mathbf{R}^{Dim}$ is the embedding of the i-th word in the sequence and $sent_{i:j}$ represents the concatenation of embeddings from word i up to and including word j. We denote the k-th dimension of the i-th word vector by $sent_i^{[k]}$ and we denote the vector containing the k-th dimension of words i to j by $sent_{i:j}^{[k]}$.

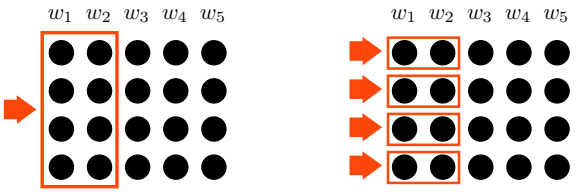

Figure 2: Left: a holistic filter matches entire word vectors (here, $ws = 2$). Right: per-dimension filters match against each dimension of the word embeddings independently.

4.1 Convolution on Multiple Perspectives

We define a convolution filter F as a tuple $\langle ws, w_F, b_F, h_F \rangle$, where ws is the sliding window width, $w_F \in \mathbf{R}^{ws \times Dim}$ is the weight vector for the filter, $b_F \in \mathbf{R}$ is the bias, and h_F is the activation function (a nonlinear function such as tanh). When filter F is applied to sequence $sent$, the inner product is computed between w_F and each possible window of word embeddings of length ws in $sent$, then the bias is added and the activation function is applied. This results in an output vector $out_F \in \mathbf{R}^{1+len-ws}$ where entry i equals

$$out_F[i] = h_F(w_F \cdot sent_{i:i+ws-1} + b_F) \quad (1)$$

where $i \in [1, 1 + len - ws]$. This filter can be viewed as performing "temporal" convolution, as it matches against regions of the word sequence. Since these filters consider the entirety of each word embedding at each position, we call them holistic filters; see the left half of Figure 2.

In addition, we target information at a finer granularity by constructing per-dimension filters $F^{[k]}$ for each dimension k of the word embeddings, where $w_{F^{[k]}} \in \mathbf{R}^{ws}$. See the right half of Figure 2. The per-dimension filters are similar to "spatial convolution" filters except that we limit each to a single, predefined dimension. We include separate per-dimension filters for each dimension of the input word embeddings.

Applying a per-dimension filter $F^{[k]} = \langle ws, w_{F^{[k]}}, b_{F^{[k]}}, h_{F^{[k]}} \rangle$ for dimension k results in an output vector $out_{F^{[k]}} \in \mathbf{R}^{1+len-ws}$ where entry i (for $i \in [1, 1 + len - ws]$) equals

$$out_{F^{[k]}}[i] = h_{F^{[k]}}(w_{F^{[k]}} \cdot sent_{i:i+ws-1}^{[k]} + b_{F^{[k]}})$$

Our use of word embeddings in both ways allows more information to be extracted for richer sentence modeling. While we typically do not expect individual dimensions of neural word embeddings

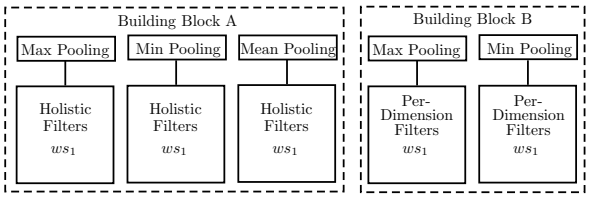

Figure 3: Each building block consists of multiple independent pooling layers and convolution layers with width ws_1. Left: $block_A$ operates on entire vectors of word embeddings. Right: $block_B$ operates on individual dimensions of word vectors to capture information of a finer granularity.

to be interpretable to humans, there may still be distinct information captured by the different dimensions that our model could exploit. Furthermore, if we update the word embeddings during learning, different dimensions could be encouraged further to capture distinct information.

We define a convolution layer as a set of convolution filters that share the same type (holistic or per-dimension), activation function, and width ws. The type, width, activation function, and number of filters $numFilter$ in the layer are chosen by the modeler and the weights of each filter (w_F and b_F) are learned.

4.2 Multiple Pooling Types

The output vector out_F of a convolution filter F is typically converted to a scalar for subsequent use by the model using some method of pooling. For example, "max-pooling" applies a max operation across the entries of out_F and returns the maximum value. In this paper, we experiment with two additional types of pooling: "min-pooling" and "mean-pooling".

A group, denoted $group(ws, pooling, sent)$, is an object that contains a convolution layer with width ws, uses pooling function $pooling$, and operates on sentence $sent$. We define a building block to be a set of groups. We use two types of building blocks, $block_A$ and $block_B$, as shown in Figure 3. We define $block_A$ as

$$\{group_A(ws_a, p, sent) : p \in \{\max, \min, \text{mean}\}\}.$$

That is, an instance of $block_A$ has three convolution layers, one corresponding to each of the three pooling functions; all have the same window size ws_a. An alternative choice would be to use the multiple types of pooling on the *same* filters (Rennie et al., 2014); we instead use independent sets of filters for the different pooling types.[1] We use blocks of type A for all holistic convolution layers.

We define $block_B$ as

$$\{group_B(ws_b, p, sent) : p \in \{\max, \min\}\}.$$

That is, $block_B$ contains two groups of convolution layers of width ws_b, one with max-pooling and one with min-pooling. Each $group_B(*)$ contains a convolution layer with Dim per-dimension convolution filters. That is, we use blocks of type B for convolution layers that operate on individual dimensions of word vectors.

We use these multiple types of pooling to extract different types of information from each type of filter. The design of each $group(*)$ allows a pooling function to interact with its own underlying convolution layers independently, so each convolution layer can learn to recognize distinct phenomena of the input for richer sentence modeling.

For a $group_A(ws_a, pooling_a, sent)$ with a convolution layer with $numFilter_A$ filters, we define the output oG_A as a vector of length $numFilter_A$ where entry j is

$$oG_A[j] = pooling_a(out_{F_j}) \quad (2)$$

where filters are indexed as F_j. That is, the output of $group_A(*)$ is a $numFilter_A$-length vector containing the output of applying the pooling function on each filter's vector of filter match outputs.[2]

A component $group_B(*)$ of $block_B$ contains Dim filters, each operating on a particular dimension of the word embeddings. We define the output oG_B of $group_B(ws_b, pooling_b, sent)$ as a $Dim \times numFilter_B$ matrix where entry $[k][j]$ is

$$oG_B[k][j] = pooling_b(out_{F_j^{[k]}})$$

where filter $F_j^{[k]}$ is filter j for dimension k.

4.3 Multiple Window Sizes

Similar to traditional n-gram-based models, we use multiple window sizes ws in our building blocks in order to learn features of different lengths. For example, in Figure 4 we use four building blocks, each with one window size $ws =$

[1] We note that max and min are not both strictly necessary when using certain activation functions, but they still may help us find a more felicitous local optimum.

[2] We note that there is no pooling across multiple filters in a layer/group, or across groups. Each pooling operation is performed independently on the matches of a single filter.

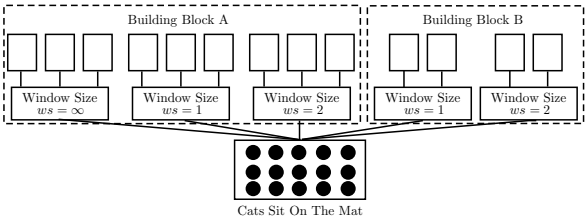

Figure 4: Example neural network architecture for a single sentence, containing 3 instances of $block_A$ (with 3 types of pooling) and 2 instances of $block_B$ (with 2 types) on varying window sizes $ws = 1, 2$ and $ws = \infty$; $block_A$ operates on entire word vectors while $block_B$ contains filters that operate on individual dimensions independently.

1 or 2 for its own convolution layers. In order to retain the original information in the sentences, we also include the entire matrix of word embeddings in the sentence, which essentially corresponds to $ws = \infty$.

The width ws represents how many words are matched by a filter, so using larger values of ws corresponds to matching longer n-grams in the input sentences. The ranges of ws values and the numbers of filters $numFilter$ of $block_A$ and $block_B$ are empirical choices tuned based on validation data.

5 Similarity Measurement Layer

In this section we describe the second part of our model, the similarity measurement layer.

Given two input sentences, the first part of our model computes sentence representations for each of them in parallel. One straightforward way to compare them is to flatten the sentence representations into two vectors, then use standard metrics like cosine similarity. However, this may not be optimal because different regions of the flattened sentence representations are from different underlying sources (e.g., groups of different widths, types of pooling, dimensions of word vectors, etc.). Flattening might discard useful compositional information for computing similarity. We therefore perform structured comparisons over particular regions of the sentence representations.

One important consideration is how to identify suitable local regions for comparison so that we can best utilize the compositional information in the sentence representations. There are many possible ways to group local comparison regions. In doing so, we consider the following four aspects: 1) whether from the same building block; 2) whether from convolutional layers with the same window size; 3) whether from the same pooling layer; 4) whether from the same filter of the underlying convolution layers.[3] We focus on comparing regions that share at least two of these conditions.

To concretize this, we provide two algorithms below to identify meaningful local regions. While there exist other sets of comparable regions that share the above conditions, we do not explore them all due to concerns about learning efficiency; we find that the subset we consider performs strongly in practice.

5.1 Similarity Comparison Units

We define two comparison units for comparing two local regions in the sentence representations:

$$comU_1(\overrightarrow{x}, \overrightarrow{y}) = \{\cos(\overrightarrow{x}, \overrightarrow{y}), L_2Euclid(\overrightarrow{x}, \overrightarrow{y}), |\overrightarrow{x} - \overrightarrow{y}|\} \quad (3)$$

$$comU_2(\overrightarrow{x}, \overrightarrow{y}) = \{\cos(\overrightarrow{x}, \overrightarrow{y}), L_2Euclid(\overrightarrow{x}, \overrightarrow{y})\} \quad (4)$$

Cosine distance (cos) measures the distance of two vectors according to the angle between them, while L_2 Euclidean distance ($L_2Euclid$) and element-wise absolute difference measure magnitude differences.

5.2 Comparison over Local Regions

Algorithms 1 and 2 show how the two sentence representations are compared in our model. Algorithm 1 works on the output of $block_A$ only, while Algorithm 2 deals with both $block_A$ and $block_B$, focusing on regions from the output of the same pooling type and same block type, but with different filters and window sizes of convolution layers.

Given two sentences S_1 and S_2, we set the maximum window size ws of $block_A$ and $block_B$ to be n, let $regM_*$ represent a $numFilter_A$ by $n+1$ matrix, and assume that each $group_*$ outputs its corresponding oG_*. The output features are accumulated in a final vector fea.

5.3 One Simplified Example

We provide a simplified working example to show how the two algorithms compare outputs of $block_A$ only. If we arrange the sentence representations into the shape of sentence matrices as in Figure 5,

[3] We note that since we apply the same network to both sentences, the same filters are used to match both sentences, so we can directly compare filter matches of individual filters across the two sentences.

Algorithm 1 Horizontal Comparison

1: **for** each pooling $p = \max, \min, \text{mean}$ **do**
2: **for** each width $ws_1 = 1...n, \infty$ **do**
3: $regM_1[*][ws_1] = group_A(ws_1, p, S_1)$
4: $regM_2[*][ws_1] = group_A(ws_1, p, S_2)$
5: **end for**
6: **for** each $i = 1...numFilter_A$ **do**
7: $fea_h = comU_2(regM_1[i], regM_2[i])$
8: accumulate fea_h for final layer
9: **end for**
10: **end for**

Algorithm 2 Vertical Comparison

1: **for** each pooling $p = \max, \min, \text{mean}$ **do**
2: **for** each width $ws_1 = 1...n, \infty$ **do**
3: $oG_{1A} = group_A(ws_1, p, S_1)$
4: **for** each width $ws_2 = 1...n, \infty$ **do**
5: $oG_{2A} = group_A(ws_2, p, S_2)$
6: $fea_a = comU_1(oG_{1A}, oG_{2A})$
7: accumulate fea_a for final layer
8: **end for**
9: **end for**
10: **for** each width $ws_1 = 1...n$ **do**
11: $oG_{1B} = group_B(ws_1, p, S_1)$
12: $oG_{2B} = group_B(ws_1, p, S_2)$
13: **for** each $i = 1...numFilter_B$ **do**
14: $fea_b = comU_1(oG_{1B}[*][i], oG_{2B}[*][i])$
15: accumulate fea_b for final layer
16: **end for**
17: **end for**
18: **end for**

then in Algorithms 1 and 2 we are essentially comparing local regions of the two matrices in two directions: along rows and columns.

In Figure 5, each column of the max/min/mean groups is compared with all columns of the same pooling group for the other sentence. This is shown in red dotted lines in the Figure and listed in lines 2 to 9 in Algorithm 2. Note that both ws_1 and ws_2 columns within each pooling group should be compared using red dotted lines, but we omit this from the figure for clarity.

In the horizontal direction, each equal-sized max/min/mean group is extracted as a vector and is compared to the corresponding one for the other sentence. This process is repeated for all rows and comparisons are shown in green solid lines, as performed by Algorithm 1.

5.4 Other Model Details

Output Fully-Connected Layer. On top of the similarity measurement layer (which outputs a vector containing all fea_*), we stack two linear layers with an activation layer in between, followed by a log-softmax layer as the final output layer, which outputs the similarity score.

Activation Layers. We used element-wise tanh as the activation function for all convolution filters and for the activation layer placed between the final two layers.

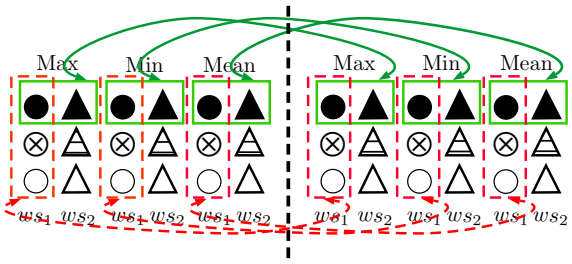

Figure 5: Simplified example of local region comparisons over two sentence representations that use $block_A$ only. The "horizontal comparison" (Algorithm 1) is shown with green solid lines and "vertical comparison" (Algorithm 2) with red dotted lines. Each sentence representation uses window sizes ws_1 and ws_2 with max/min/mean pooling and $numFilter_A = 3$ filters.

6 Experiments and Results

Everything necessary to replicate our experimental results can be found in our open-source code repository.[4]

6.1 Tasks and Datasets

We consider three sentence pair similarity tasks:

1. Microsoft Research Paraphrase Corpus (MSRP). This data was collected from news sources (Dolan et al., 2004) and contains 5,801 pairs of sentences, with 4,076 for training and the remaining 1,725 for testing. Each sentence pair is annotated with a binary label indicating whether the two sentences are paraphrases, so the task here is binary classification.

2. Sentences Involving Compositional Knowledge (SICK) dataset. This data was collected for the 2014 SemEval competition (Marelli et al., 2014) and consists of 9,927 sentence pairs, with 4,500 for training, 500 as a development set, and the remaining 4,927 in the test set. The sentences are drawn from image and video descriptions. Each sentence pair is annotated with a relatedness score $\in [1, 5]$, with higher scores indicating the two sentences are more closely-related.

[4] http://hohocode.github.io/textSimilarityConvNet/

3. Microsoft Video Paraphrase Corpus (MSRVID). This dataset was collected for the 2012 SemEval competition and consists of 1,500 pairs of short video descriptions which were then annotated (Agirre et al., 2012). Half of it is for training and the other half is for testing. Each sentence pair has a relatedness score $\in [0, 5]$, with higher scores indicating the two sentences are more closely-related.

6.2 Training

We use a hinge loss for the MSRP paraphrase identification task. This is simpler than log loss since it only penalizes misclassified cases. The training objective is to minimize the following loss (summed over examples $\langle x, y_{gold} \rangle$):

$$loss(\theta, x, y_{gold}) = \sum_{y' \neq y_{gold}} \max(0, 1 + f_\theta(x, y') - f_\theta(x, y_{gold})) \quad (5)$$

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

We use regularized KL-divergence loss for the semantic relatedness tasks (SICK and MSRVID), since the goal is to predict the similarity of the two sentences. The training objective is to minimize the KL-divergence loss plus an L_2 regularizer:

$$loss(\theta) = \frac{1}{m} \sum_{k=1}^{m} KL\left(f^k \,||\, \widehat{f}_\theta^k\right) + \frac{\lambda}{2}||\theta||_2^2 \quad (6)$$

where $\widehat{f}_\theta$ is the predicted distribution with model weight vector θ, f is the ground truth, m is the number of training examples, and λ is the regularization parameter. Note that we use the same KL-loss function and same sparse target distribution technique as Tai et al. (2015).

6.3 Experiment Settings

We conduct experiments with ws values in the range $[1, 3]$ as well as $ws = \infty$ (no convolution).

We use multiple kinds of embeddings to represent each sentence, both on words and part-of-speech (POS) tags. We use the $Dim_g = 300$-dimensional GloVe word embeddings (Pennington et al., 2014) trained on 840 billion tokens. We use $Dim_k = 25$-dimensional PARAGRAM vectors (Wieting et al., 2015) only on the MSRP task since they were developed for paraphrase tasks, having been trained on word pairs from the Paraphrase Database (Ganitkevitch et al., 2013). For POS embeddings, we run the Stanford POS tagger (Manning et al., 2014) on the English side of the Xinhua machine translation parallel corpus, which consists of Xinhua news articles with approximately 25 million words. We then train $Dim_p = 200$-dimensional POS embeddings using the `word2vec` toolkit (Mikolov et al., 2013). Adding POS embeddings is expected to retain syntactic information which is reported to be effective for paraphrase identification (Das and Smith, 2009). We use POS embeddings only for the MSRP task.

Therefore for MSRP, we concatenate all word and POS embeddings and obtain $Dim = Dim_g + Dim_p + Dim_k = 525$-dimension vectors for each input word; for SICK and MSRVID we only use $Dim = 300$-dimension GloVe embeddings.

We use 5-fold cross validation on the MSRP *training* data for tuning, then largely re-use the same hyperparameters for the other two datasets. However, there are two changes: 1) for the MSRP task we update word embeddings during training but not so on SICK and MSRVID tasks; 2) we set the fully connected layer to contain 250 hidden units for MSRP, and 150 for SICK and MSRVID. These changes were done to speed up our experimental cycle on SICK and MSRVID; on SICK data they are the same experimental settings as used by Tai et al. (2015), which makes for a cleaner empirical comparison.

We set the number of holistic filters in $block_A$ to be the same as the input word embeddings, therefore $numFilter_A = 525$ for MSRP and $numFilter_A = 300$ for SICK and MSRVID. We set the number of per-dimension filters in $block_B$ to be $numFilter_B = 20$ per dimension for all three datasets, which corresponds to $20 * Dim$ filters in total.

We perform optimization using stochastic gradient descent (Bottou, 1998). The backpropagation algorithm is used to compute gradients for all parameters during training (Goller and Kuchler, 1996). We fix the learning rate to 0.01 and regularization parameter $\lambda = 10^{-4}$.

6.4 Results on Three Datasets

Results on MSRP Data. We report F1 scores and accuracies from prior work in Table 1. Ap-

Model	Acc.	F1
Hu et al. (2014) ARC-I	69.6%	80.3%
Hu et al. (2014) ARC-II	69.9%	80.9%
Blacoe and Lapata (2012)	73.0%	82.3%
Fern and Stevenson (2008)	74.1%	82.4%
Finch (2005)	75.0%	82.7%
Das and Smith (2009)	76.1%	82.7%
Wan et al. (2006)	75.6%	83.0%
Socher et al. (2011)	76.8%	83.6%
Madnani et al. (2012)	77.4%	84.1%
Ji and Eisenstein (2013)	**80.41**%	**85.96**%
Yin and Schütze (2015) (without pretraining)	72.5%	81.4%
Yin and Schütze (2015) (with pretraining)	78.1%	84.4%
Yin and Schütze (2015) (pretraining+sparse features)	78.4%	84.6%
This work	**78.60**%	**84.73**%

Table 1: Test set results on MSRP for paraphrase identification. Rows in grey are neural network-based approaches.

proaches shown in gray rows of the table are based on neural networks. The recent approach by Yin and Schütze (2015) includes a pretraining technique which significantly improves results, as shown in the table. We do not use any pretraining but still slightly outperform their best results which use both pretraining and additional sparse features from Madnani et al. (2012).

When comparing to their model without pretraining, we outperform them by 6% absolute in accuracy and 3% in F1. Our model is also superior to other recent neural network models (Hu et al., 2014; Socher et al., 2011) without requiring sparse features or unlabeled data as in (Yin and Schütze, 2015; Socher et al., 2011). The best result on MSRP is from Ji and Eisenstein (2013) which uses unsupervised learning on the MSRP test set and rich sparse features.

Results on SICK Data. Our results on the SICK task are summarized in Table 2, showing Pearson's r, Spearman's ρ, and mean squared error (MSE). We include results from the literature as reported by Tai et al. (2015), including prior work using recurrent neural networks (RNNs), the best submissions in the SemEval-2014 competition, and variants of LSTMs. When measured by Pearson's r, the previous state-of-the-art approach uses a tree-structured LSTM (Tai et al., 2015); note that their best results require a dependency parser.

On the contrary, our approach does not rely on parse trees, nor do we use POS/PARAGRAM embeddings for this task. The word embeddings,

Model	r	ρ	MSE
Socher et al. (2014) DT-RNN	0.7863	0.7305	0.3983
Socher et al. (2014) SDT-RNN	0.7886	0.7280	0.3859
Lai and Hockenmaier (2014)	0.7993	0.7538	0.3692
Jimenez et al. (2014)	0.8070	0.7489	0.3550
Bjerva et al. (2014)	0.8268	0.7721	0.3224
Zhao et al. (2014)	0.8414	-	-
LSTM	0.8477	0.7921	0.2949
Bi-LSTM	0.8522	0.7952	0.2850
2-layer LSTM	0.8411	0.7849	0.2980
2-layer Bidirectional LSTM	0.8488	0.7926	0.2893
Tai et al. (2015) Const. LSTM	0.8491	0.7873	0.2852
Tai et al. (2015) Dep. LSTM	0.8676	**0.8083**	**0.2532**
This work	**0.8686**	0.8047	0.2606

Table 2: Test set results on SICK, as reported by Tai et al. (2015), grouped as: (1) RNN variants; (2) SemEval 2014 systems; (3) sequential LSTM variants; (4) dependency and constituency tree LSTMs (Tai et al., 2015). Evaluation metrics are Pearson's r, Spearman's ρ, and mean squared error (MSE).

Model	Pearson's r
Rios et al. (2012)	0.7060
Wang and Cer (2012)	0.8037
Beltagy et al. (2014)	0.8300
Bär et al. (2012)	0.8730
Šarić et al. (2012)	0.8803
This work	**0.9090**

Table 3: Test set results on MSRVID data. The Bär et al. (2012) and Šarić et al. (2012) results were the top two submissions in the Semantic Textual Similarity task at the SemEval-2012 competition.

sparse distribution targets, and KL loss function are exactly the same as used by Tai et al. (2015), therefore representing comparable conditions.

Results on MSRVID Data. Our results on the MSRVID data are summarized in Table 3, which includes the top 2 submissions in the Semantic Textual Similarity (STS) task from SemEval-2012. We find that we outperform the top system from the task by nearly 3 points in Pearson's r.

6.5 Model Ablation Study

We report the results of an ablation study in Table 4. We identify nine major components of our approach, remove one at a time (if applicable), and perform re-training and re-testing for all three tasks. We use the same experimental settings in Sec. 6.3 and report differences (in accuracy for MSRP, Pearson's r for SICK/MSRVID) compared to our results in Tables 1–3.

Gp	ID	Ablation Component	MSRP Accuracy Diff.	MSRVID Pearson Diff.	SICK Pearson Diff.
1	1	Remove POS embeddings (Sec. 6.3)	-0.81	NA	NA
	2	Remove PARAGRAM embeddings (Sec. 6.3)	-1.33	NA	NA
2	3	Remove per-dimension embeddings, building block A only (Sec. 4.1)	-0.75	-0.0067	-0.0014
	4	Remove $\min$ and mean pooling, use $\max$ pooling only (Sec. 4.2)	-0.58	-0.0112	+0.0001
	5	Remove multiple widths, $ws = 1$ and $ws = \infty$ only (Sec. 4.3)	-2.14	-0.0048	-0.0012
3	6	Remove cosine and $L_2 Euclid$ distance in $comU_*$ (Sec. 5.1)	-2.31	-0.0188	-0.0309
4	7	Remove Horizontal Algorithm (Sec. 5.2)	-0.92	-0.0097	-0.0117
	8	Remove Vertical Algorithm (Sec. 5.2)	-2.15	-0.0063	-0.0027
	9	Remove similarity layer (completely flatten) (Sec. 5)	-1.90	-0.0121	-0.0288

Table 4: Ablation study over test sets of all three datasets. Nine components are divided into four groups. We remove components one at a time and show differences.

The nine components can be divided into four groups: (1) input embeddings (components 1–2); (2) sentence modeling (components 3–5); (3) similarity measurement metrics (component 6); (4) similarity measurement layer (components 7–9). For MSRP, we use all nine components. For SICK and MSRVID, we use components 3–9 (as described in Sec. 6.3).

From Table 4 we find drops in performance for all components, with the largest differences appearing when removing components of the similarity measurement layer. For example, conducting comparisons over flattened sentence representations (removing component 9) leads to large drops across tasks, because this ignores structured information within sentence representations. Groups (1) and (2) are also useful, particularly for the MSRP task, demonstrating the extra benefit obtained from our multi-perspective approach in sentence modeling.

We see consistent drops when ablating the Vertical/Horizontal algorithms that target particular regions for comparison. Also, removing group (3) hinders both the Horizontal and Vertical algorithms (as described in Section 5.1), so its removal similarly causes large drops in performance. Though convolutional neural networks already perform strongly when followed by flattened vector comparison, we are able to leverage the full richness of the sentence models by performing structured similarity modeling on their outputs.

7 Discussion and Conclusion

On the SICK dataset, the dependency tree LSTM (Tai et al., 2015) and our model achieve comparable performance despite taking very different approaches. Tai et al. use syntactic parse trees and gating mechanisms to convert each sentence into a vector, while we use large sets of flexible feature extractors in the form of convolution filters, then compare particular subsets of features in our similarity measurement layer.

Our model architecture, with its many paths of information flow, is admittedly complex. Though we have removed hand engineering of features, we have added a substantial amount of functional architecture engineering. This may be necessary when using the small training sets provided for the tasks we consider here. We conjecture that a simpler, deeper neural network architecture may outperform our model when given large amounts of training data, but we leave an investigation of this direction to future work.

In summary, we developed a novel model for sentence similarity based on convolutional neural networks. We improved both sentence modeling and similarity measurement. Our model achieves highly competitive performance on three datasets. Ablation experiments show that the performance improvement comes from our use of multiple perspectives in both sentence modeling and structured similarity measurement over local regions of sentence representations. Future work could extend this model to related tasks including question answering and information retrieval.

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 the anonymous reviewers for their feedback and CLIP labmates for their support.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: a pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 385–393.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1247–1255.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 435–440.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1210–1219.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: formal semantics for recognizing textual entailment and determining semantic similarity. *International Workshop on Semantic Evaluation*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556.

Léon Bottou. 1998. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 160–167.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468–476.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Samuel Fern and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Computational Linguistics UK 11th Annual Research Colloquium*.

Andrew Finch. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the International Workshop on Paraphrasing*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks*, pages 347–352.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 864–872.

Samer Hassan. 2011. *Measuring Semantic Relatedness Using Salient Encyclopedic Concepts*. Ph.D. thesis, University of North Texas, Denton, Texas, USA.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods for Natural Language Processing*, pages 891–896.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: combining soft cardinality features for semantic textual similarity, relatedness and entailment. *International Workshop on Semantic Evaluation*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: a denotational and distributional approach to semantics. *International Workshop on Semantic Evaluation*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *International Workshop on Semantic Evaluation*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Steven Rennie, Vaibhava Goel, and Samuel Thomas. 2014. Deep order statistic networks. In *Proceedings of the IEEE Workshop on Spoken Language Technology*.

Miguel Rios, Wilker Aziz, and Lucia Specia. 2012. UOW: semantically informed text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 673–678.

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*, pages 441–448.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

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.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using Dependency-based Features to Take the "Para-farce" out of Paraphrase. In *Australasian Language Technology Workshop*, pages 131–138.

Mengqiu Wang and Daniel Cer. 2012. Probabilistic edit distance metrics for STS. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 648–654.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: scaling up to large vocabulary image annotation. In *International Joint Conference on Artificial Intelligence*, pages 2764–2770.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. ECNU: one stone two birds: ensemble of heterogenous measures for semantic relatedness and textual entailment. *International Workshop on Semantic Evaluation*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612.