Convolutional Neural Network for Paraphrase Identification

Wenpeng Yin and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
wenpeng@cis.uni-muenchen.de

Abstract

We present a new deep learning architecture Bi-CNN-MI for paraphrase identification (PI). Based on the insight that PI requires comparing two sentences *on multiple levels of granularity*, we learn multigranular sentence representations using convolutional neural network (CNN) and model interaction features at each level. These features are then the input to a logistic classifier for PI. All parameters of the model (for embeddings, convolution and classification) are directly optimized for PI. To address the lack of training data, we pretrain the network in a novel way using a language modeling task. Results on the MSRP corpus surpass that of previous NN competitors.

1 Introduction

In this paper, we address the problem of paraphrase identification. It is usually formalized as a binary classification task: for two sentences (S_1, S_2), determine whether they roughly have the same meaning.

Inspired by recent successes of deep neural networks (NNs) in fields like computer vision (Neverova et al., 2014), speech recognition (Deng et al., 2013) and natural language processing (Collobert and Weston, 2008), we adopt a deep learning approach to paraphrase identification in this paper.

The key observation that motivates our NN architecture is that the identification of a paraphrase relationship between S_1 and S_2 requires an analysis *at multiple levels of granularity.*

(A1) "Detroit manufacturers have raised vehicle prices by ten percent." – (A2) "GM, Ford and Chrysler have raised car prices by five percent."

Example A1/A2 shows that paraphrase identification requires comparison *at the word level.* A1 cannot be a paraphrase of A2 because the numbers "ten" and "five" are different.

(B1) "Mary gave birth to a son in 2000." – (B2) "He is 14 years old and his mother is Mary."

PI for B1/B2 can only succeed *at the sentence level* since B1/B2 express the same meaning using very different means.

Most work on paraphrase identification has focused on only one level of granularity: either on low-level features (e.g., Madnani et al. (2012)) or on the sentence level (e.g., ARC-I, Hu et al. (2014)).

An exception is the RAE model (Socher et al., 2011). It computes representations on all levels of a parse tree: each node – including nodes corresponding to words, phrases and the entire sentence – is represented as a vector. RAE then computes a $n_1 \times n_2$ comparison matrix of the two trees derived from S_1 and S_2 respectively, where n_1, n_2 are the number of nodes and each comparison is the Euclidean distance between two vectors. This is then the basis for paraphrase classification.

RAE (Socher et al., 2011) is one of three prior NN architectures that we draw on to design our system. It embodies the key insight that paraphrase identification involves analysis of information at multiple levels of granularity. However, relying on parsing has limitations for noisy text and for other applications in which highly accurate parsers are not available. We extend the basic idea of RAE by exploring stacked convolution layers which on one hand use sliding windows to split sentences into flexible phrases, furthermore, higher layers are able to extract more abstract features of longer-range phrases by combining phrases in lower layers.

A representative way of doing this in deep learning is the work by Kalchbrenner et al. (2014), the second prior NN architecture that we draw on. They use convolution to learn representations at multiple levels (Collobert and Weston, 2008). The motivation for convolution is that natural language consists of long sequences in which many short subsequences contribute in a stable way to the structure and meaning of the long sequence *regardless of the position of the subsequence within the long sequence*. Thus, it is advantageous to learn convolutional filters that detect a particular feature regardless of position. Kalchbrenner et al. (2014)'s architecture extends this idea in two important ways. First, k-max pooling extracts the k top values from a sequence of convolutional filter applications and guarantees a fixed length output. Second, they stack several levels of convolutional filters, thus achieving multigranularity. We incorporate this architecture as the part that analyzes an individual sentence.

The third prior NN architecture we draw on is ARC proposed by Hu et al. (2014) who also attempt to exploit convolution for paraphrase identification. Their key insight is that *we want to be able to directly optimize the entire system for the task we are addressing,* i.e., for paraphrase identification. Hu et al. (2014) do this by adopting a Siamese architecture: their NN consists of two shared-weight sentence analysis NNs that feed into a binary classifier that is directly trained on labeled sentence pairs. As we will show below, this is superior to separating the two steps: first learning sentence representations, then training binary classification for fixed, learned sentence representations as Bromley et al. (1993), Socher et al. (2011) and many others do.

We can now give an overview of our NN architecture (Figure 1). We call it Bi-CNN-MI: "Bi-CNN" stands for double CNNs used in Siamese framework, "MI" for *multigranular interaction* features. Bi-CNN-MI has three parts: (i) the sentence analysis network CNN-SM, (ii) the sentence interaction model CNN-IM and (iii) a logistic regression on top of the network that performs paraphrase identification. We now describe these three parts in detail.

(i) Following Kalchbrenner et al. (2014), we design CNN-SM, a convolutional sentence analysis NN that computes representations at four different levels: word, short ngram, long ngram and sentence. This multigranularity is important because paraphrase identification benefits from analyzing sentences at multiple levels.

(ii) Following Socher et al. (2011), CNN-IM, the interaction model, computes interaction features as $s_1 \times s_2$ matrices, where s_i is the number of items of a certain granularity in S_i. In contrast to Socher et al. (2011), CNN-IM computes these features at fixed levels and only for comparable units; e.g., we do not compare single words with entire sentences.

(iii) Following Hu et al. (2014), we integrate two copies of CNN-SM into a Siamese architecture that allows to optimize all parameters of the NN for paraphrase identification. In our case, these parameters include parameters for word embedding, for convolution filters, and for the classification of paraphrase candidate pairs. In contrast to Hu et al. (2014), the inputs to the final paraphrase candidate pair classification layer are *interaction feature matrices at multiple levels* – as opposed to *single-level features* that do not directly *compare an element of S_1 with a potentially corresponding element of S_2*.

There is one other problem we have to address to get good performance. Training sets for paraphrase identification are small in comparison with the high complexity of the task. Training a complex network like Bi-CNN-MI with a large number of parameters on a small training set is not promising due to sparseness and likely overfitting.

In order to make full use of the training data, we propose a new unsupervised training scheme CNN-LM (CNN Language Model) to pretrain the largest part of the model, the sentence analysis network CNN-SM. The key innovation is that we use a language modeling task in a setup similar to autoencoding for pretraining (see below for details). This means that embedding and convolutional parameters can be pretrained on very large corpora since no human labels are required for pretraining.

We will show below that this pretraining is critical for getting good performance in the paraphrase task. However, the general design principle of this type of unsupervised pretraining should be widely applicable given that next-word prediction training is possible in many NLP applications. Thus, this new way of unsupervised pretraining could be an important contribution of the paper independent of paraphrase identification.

Section 2 discusses related work. Sections 3 and 4 introduce the sentence model CNN-SM and the sentence interaction model CNN-IM. Section 5 describes the training regime. The experiments are presented in Section 6. Section 7 concludes.

2 Related work

Bi-CNN-MI is closely related to NN models for sentence representations and for text matching.

A pioneering work using CNN to model sentences is (Collobert and Weston, 2008). They conducted convolutions on sliding windows of a sentence and finally use max pooling to form a sentence representation. Kalchbrenner et al. (2014) introduce k-max pooling and stacking of several CNNs as discussed in Section 1.

Lu and Li (2013) developed a deep NN to match short texts, where interactions between components within the two objects were considered. These interactions were obtained via LDA (Blei et al., 2003). A two-dimensional interaction space is formed, then those local decisions will be sent to the corresponding neurons in upper layers to get rounds of fusion, finally logistic regression in the output layer produces the final matching score. Drawbacks of this approach are that LDA parameters are not optimized for the paraphrase task and that the interactions are formed on the level of single words only.

Gao et al. (2014) model *interestingness* between two documents with deep NNs. They map source-target document pairs to feature vectors in a latent space in such a way that the distance between the source document and its corresponding interesting target in that space is minimized. Interestingness is more like topic relevance, based mainly on the aggregate meaning of lots of keywords. Additionally, their model is a document-level model and is not multigranular.

Madnani et al. (2012) treated paraphrase relationship as a kind of mutual translation, hence combined eight kinds of machine translation metrics including BLEU (Papineni et al., 2002), NIST (Doddington, 2002), TER (Snover et al., 2006), TERp (Snover et al., 2009), METEOR (Denkowski and Lavie, 2010), SEPIA (Habash and Elkholy, 2008), BADGER (Parker, 2008) and MAXSIM (Chan and Ng, 2008). These features are not multigranular; rather they are low-level only; high-level features – e.g., a representation of the entire sentence – are not considered.

Bach et al. (2014) claimed that elementary discourse units obtained by segmenting sentences play an important role in paraphrasing. Their conclusion also endorses Socher et al. (2011)'s and our work, for both take similarities between component phrases into account.

We discussed Socher et al. (2011)'s RAE and Hu et al. (2014)'s ARC-I in Section 1. In addition to similarity matrices there are two other important aspects of (Socher et al., 2011). First, the similarity matrices are converted to a fixed size feature vector by *dynamic pooling*. We adopt this approach in Bi-CNN-MI; see Section 4.2 for details.

Second, (Socher et al., 2011) is partially based on parsing as is some other work on paraphrase identification (e.g., Wan et al. (2006), Ji and Eisenstein (2013)). Parsing is a potentially powerful tool for identifying the important meaning units of a sentence, which can then be the basis for determining meaning equivalence. However, reliance on parsing makes these approaches less flexible. For example, there are no high-quality parsers available for some domains and some languages. Our approach is in principle applicable for any domain and language. It is also unclear how we would identify comparable units in the parse trees of S_1 and S_2 if the parse trees have different height and the sentences different lengths. A key property of Bi-CNN-MI is that it is designed to produce units at fixed levels and only units at the same level are compared with each other.

3 Convolution sentence model CNN-SM

Our network Bi-CNN-MI for paraphrase detection (Figure 1) consists of four parts. On the left and on the right there are two multilayer NNs with seven layers, from "initialized word embeddings: sentence 1/2" to "k-max pooling". The weights of these two NNs are shared. This part of Bi-CNN-MI is based on (Kalchbrenner et al., 2014) and we refer to it as convolutional sentence model CNN-SM.

Between the two CNN-SMs there is the interaction model CNN-IM, consisting of four feature ma-

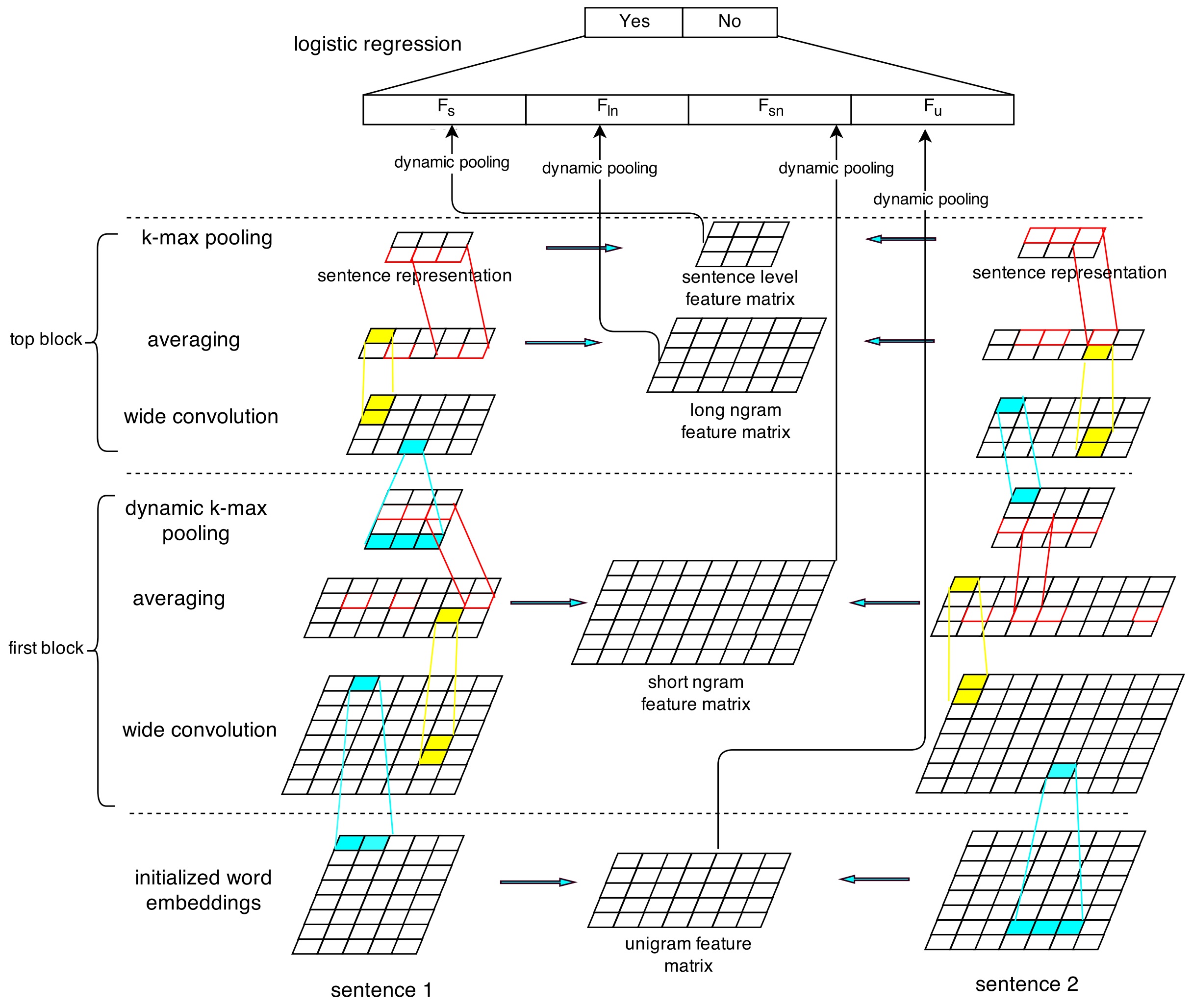

Figure 1: The paraphrase identification architecture Bi-CNN-MI

trices (unigram, short ngram, long ngram, sentence). CNN-IM feeds into a logistic classifier that performs paraphrase detection. See Sections 4 and 5 for these two parts of Bi-CNN-MI.

3.1 Wide convolution

We use Kalchbrenner et al. (2014)'s **wide one-dimensional convolution**. Denoting the number of tokens of S_i as $|S_i|$, we convolve weight matrix $\mathbf{M} \in \mathbb{R}^{d \times m}$ over sentence representation matrix $\mathbf{S} \in \mathbb{R}^{d \times |S_i|}$ and generate a matrix $\mathbf{C} \in \mathbb{R}^{d \times (|S_i|+m-1)}$ each column of which is the representation of an m-gram. d is the dimension of word (and also ngram) embeddings. m is filter width.

Our motivation for using convolution is that after training, a convolutional filter corresponds to a feature detector that learns to recognize a class of m-grams that is useful for paraphrase detection.

3.2 Averaging

After convolution, to build simple relations across rows, each odd row and the row behind immediately are *averaged*, generating matrix $\mathbf{A} \in \mathbb{R}^{\frac{d}{2} \times (|S_i|+m-1)}$. Namely:

$$\mathbf{A} = (\mathbf{C}_{\text{odd}} + \mathbf{C}_{\text{even}})/2 \tag{1}$$

where $\mathbf{C}_{\text{odd}}$, $\mathbf{C}_{\text{even}}$ denote the odd and even rows of $\mathbf{C}$, respectively. Finally, this convolution layer will output matrix $\mathbf{B}$ whose jth column is defined thus:

$$\mathbf{B}_{:,j} = \tanh(\mathbf{A}_{:,j} + \mathbf{b}^T) \quad 0 \le j < (|S_i| + m - 1) \tag{2}$$

$\mathbf{b}$ is a bias vector with dimension $d/2$, same for each column.

3.3 Dynamic k-max pooling

We use Kalchbrenner et al. (2014)'s **dynamic k-max pooling** to extract features for variable-length sentences. It extracts k_{dy} top values from each dimension after the first layer of averaging and $k_{\text{top}} = 4$ top values after the top layer of averaging. We set

$$k_{\text{dy}} = \max(k_{\text{top}}, |S_i|/2 + 1) \tag{3}$$

Thus, k_{dy} depends on the length of S_i.

The sequence of layers in (Kalchbrenner et al., 2014) is convolution, folding, k-max pooling, $\tanh$. We experimented with this sequence and found that after k-max pooling many $\tanh$ units had an input close to 1, in the nondynamic range of the function (since the input is the addition of several values). This makes learning difficult. We therefore changed the sequence to convolution, averaging, $\tanh$, k-max pooling. This makes it less likely that $\tanh$ units will be saturated.

We have described convolution, averaging and k-max pooling. We can stack several blocks of these three layers to form deep architectures, as the two blocks (marked "first block" and "top block") in Figure 1.

4 Convolution interaction model CNN-IM

After the introduction in the previous section of the CNN-SM part of our architecture for processing an *individual sentence*, we now turn to the CNN-IM interaction model that computes the four feature matrices in Figure 1 to assess the *interactions between the two sentences*.

4.1 Feature matrices

One key innovation of our approach is multigranularity: we compute similarity between the two paraphrase candidates on multiple levels. Specifically, we compute similarity at four levels in this paper: unigram, short ngram, long ngram and sentence. We use notation $l \in \{\text{u}, \text{sn}, \text{ln}, \text{s}\}$ to refer to the four levels, and use $\mathbf{S}^{i,l}$ to denote the matrix representing sentence S_i at level l. For level l, we compute feature matrices $\hat{\mathbf{F}}_l$ as follows:

$$\hat{\mathbf{F}}_l^{ij} = \exp\left(\frac{-||\mathbf{S}^{1,l}_{:,i} - \mathbf{S}^{2,l}_{:,j}||^2}{2\beta}\right) \tag{4}$$

where $||\mathbf{S}^{1,l}_{:,i} - \mathbf{S}^{2,l}_{:,j}||^2$ is the Euclidean distance between the representations of the ith item of S_1 and the jth item of S_2 on level l. We set $\beta = 2$ (cf. Wu et al. (2013)).

We do not use cosine because the magnitude of the activations of hidden units is important, not just the overall direction; e.g., if $\mathbf{S}^{1,l}_{:,i}$ and $\mathbf{S}^{2,l}_{:,j}$ point in the same direction, but activations are much larger in $\mathbf{S}^{2,l}_{:,j}$, then the two vectors are very dissimilar.

The lowest level feature matrix ($l = \text{u}$) is the unigram similarity matrix $\hat{\mathbf{F}}_{\text{u}}$. It has size $|S_1| \times |S_2|$. The feature entry $\hat{\mathbf{F}}_{\text{u}}{}^{ij}$ is the similarity between the ith word of S_1 and the jth word of S_2 where each word is represented by a d-dimensional word embedding ($d = 100$ in our experiments).

The next level feature matrix is the short ngram similarity matrix $\hat{\mathbf{F}}_{\text{sn}}$. It has size $(|S_1| + m_{\text{sn}} - 1) \times (|S_2| + m_{\text{sn}} - 1)$ where $m_{\text{sn}} = 3$ is the filter width in this convolution layer and $|S_i| + m_{\text{sn}} - 1$ is the number of short ngrams in S_i. The feature entry $\hat{\mathbf{F}}_{\text{sn}}{}^{ij}$ is the similarity between two $d/2$-dimensional vectors representing two short ngrams from S_1 and S_2.

We use multiple feature maps to improve the system performance. Different feature maps are expected to extract different kinds of sentence features, and can be implemented in the same convolution layer in parallel. Specifically, we use $f_{\text{sn}} = 6$ feature maps on this level following Kalchbrenner et al. (2014). Thus, we actually compute six feature matrices $\hat{\mathbf{F}}_{\text{sn},i}$ $(i = 1, \cdots, f_{\text{sn}})$, one for each pair of feature maps that share convolution weights while derived from S_1 and S_2 respectively. (Figure 1 only shows one of those six matrices.)

The next level feature matrix is the long ngram similarity matrix $\hat{\mathbf{F}}_{\text{ln}}$. It has size $(k_{\text{dy},1} + m_{\text{ln}} - 1) \times (k_{\text{dy},2} + m_{\text{ln}} - 1)$ where $k_{\text{dy},i}$ (Equation 3) is the k value in dynamic k-max pooling for sentence i, $k_{\text{dy},i} + m_{\text{ln}} - 1$ is the number of long ngrams in S_i and $m_{\text{ln}} = 5$ is the filter width in this convolution layer. The feature entry $\hat{\mathbf{F}}_{\text{ln}}{}^{ij}$ is the similarity between two $d/4$-dimensional vectors representing two long ngrams from S_1 and S_2.

We use $f_{\text{ln}} = 14$ feature maps on this level following Kalchbrenner et al. (2014). Thus, we compute 14 feature matrices $\hat{\mathbf{F}}_{\text{ln},i}$ $(i = 1, \cdots, f_{\text{ln}})$, in a way analogous to the $f_{\text{sn}} = 6$ feature maps $\hat{\mathbf{F}}_{\text{sn},i}$.

The last feature matrix is the sentence similarity matrix $\hat{\mathbf{F}}_{\text{s}}$. It has size $k_{\text{top}} \times k_{\text{top}}$ where $k_{\text{top}} = 4$ is the parameter in k-max pooling at the last max pooling layer. The feature entry $\hat{\mathbf{F}}_{\text{s}}{}^{ij}$ is the similarity between two $d/4$-dimensional vectors computed by max pooling from S_1 and S_2.

For $l = \text{s}$, there are also $f_{\text{ln}} = 14$ feature matrices $\hat{\mathbf{F}}_{\text{s},i}$ $(i = 1, \cdots, f_{\text{ln}})$, analogous to the $f_{\text{ln}} = 14$ feature matrices $\hat{\mathbf{F}}_{\text{ln},i}$.

A general design principle of the architecture is that we compute each interaction feature matrix between two feature maps that share the same convolution weights. Two feature maps learned with the same filter will contain the same kinds of features derived from the input.

4.2 Dynamic pooling of feature matrices

As sentence lengths vary, feature matrices $\hat{\mathbf{F}}_l$ have different sizes, which makes it impossible to use them directly as input of the last layer.

That means we need to map $\hat{\mathbf{F}}_l \in \mathbb{R}^{r \times c}$ into a matrix $\mathbf{F}_l$ of fixed size $r' \times c'$ ($l \in \{\text{u}, \text{sn}, \text{ln}, \text{s}\}$; r', c' are parameters and are the same for all sentence pairs while r, c depend on $|S_1|$ and $|S_2|$). Dynamic pooling divides $\hat{\mathbf{F}}_l$ into $r' \times c'$ nonoverlapping *(dynamic) pools* and copies the maximum value in each dynamic pool to $\mathbf{F}_l$. Our method is similar to (Socher et al., 2011), but preserves locality better.

$\hat{\mathbf{F}}_l$ can be split into equal regions only if r (resp. c) is divisible by r' (resp. c'). Otherwise, for $r > r'$ and if $r \bmod r' = b$, the dynamic pools in the first $r' - b$ splits each have $\lfloor \frac{r}{r'} \rfloor$ rows while the remaining b splits each have $\lfloor \frac{r}{r'} \rfloor + 1$ rows. In Figure 2, a $r \times c = 4 \times 5$ matrix (left) is split into $r' \times c' = 3 \times 3$ dynamic pools (middle): each row is split into [1, 1, 2] and each column is split into [1, 2, 2].

If $r < r'$, we first repeat all rows until no fewer than r' rows remain. Then first r' rows are kept and split into r' dynamic pools. The same principle applies to the partitioning of columns. In Figure 2 (right), the areas with dashed lines and dotted lines are repeated parts for rows and columns, respectively; each cell is its own dynamic pool.

5 Training

5.1 Supervised training

Dynamic pooling gives us fixed size interaction feature matrices for sentence, ngram and unigram levels. As shown in Figure 1, the concatenation of these features ($\mathbf{F}_{\text{s}}$, $\mathbf{F}_{\text{ln}}$, $\mathbf{F}_{\text{sn}}$ and $\mathbf{F}_{\text{u}}$) is the input to a logistic regression layer for paraphrase classification. We have now described all three parts of Bi-CNN-MI: CNN-SM, CNN-IM and logistic regression.

Bi-CNN-MI with all its parameters – including word embeddings and convolution weights – is trained on MSRP. We initialize embeddings with those provided by Turian et al. (2010)[1] (based on Collobert and Weston (2008)). For layer sn, we have $f_{\text{sn}} = 6$ feature maps and set filter width $m_{\text{sn}} = 3$. For layer ln, we have $f_{\text{ln}} = 14$ feature maps and set filter width $m_{\text{ln}} = 5$ and $k_{\text{top}} = 4$. Dynamic pooling

[1]`metaoptimize.com/projects/wordreprs/`

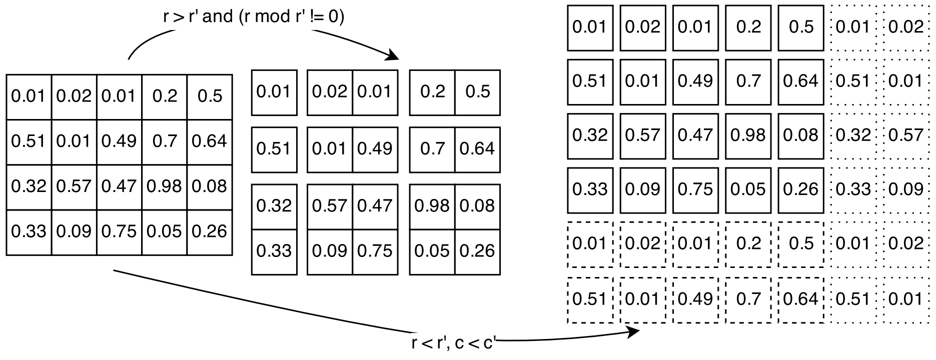

Figure 2: Partition methods in dynamic pooling. Original matrix with size 4×5 is mapped into matrix with size 3×3 and matrix with size 6×7, respectively. Each dynamic pool is distinguished by a border of empty white space around it.

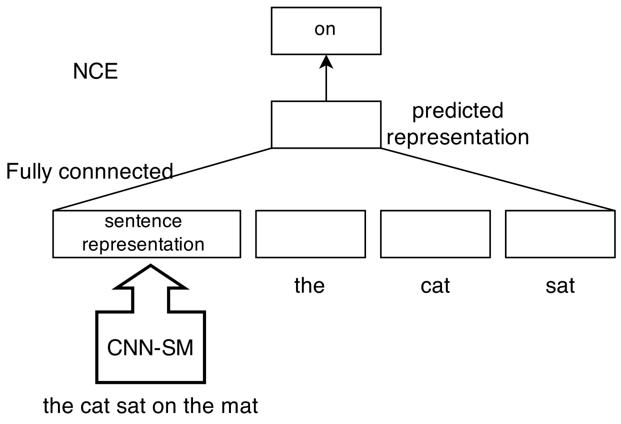

Figure 3: Unsupervised architecture: CNN-LM

sizes are 10×10, 10×10, 6×6, 2×2 for unigram, short ngram, long ngram and sentence, respectively. For training, we employ mini-batch of size 70, L_2 regularization with weight 5×10^{-4} and Adagrad (Duchi et al., 2011).

5.2 Unsupervised pretraining

One of the key contributions of this paper is the architecture CNN-LM shown in Figure 3. CNN-LM is used to pretrain the convolutional filters on unlabeled data. This addresses sparseness and limited training data for paraphrase identification.

The convolution sentence model CNN-SM (Section 3) is part of CNN-LM ("CNN-SM" in Figure 3). The input to CNN-SM is the entire sentence ("the cat sat on the mat"); its output ("sentence representation" in the leftmost rectangle in Figure 3 and the two grids labeled "sentence representation" in the top layer of the top block in Figure 1) is concatenated with a history consisting of the embeddings of the $h = 3$ preceding words ("the", "cat", "sat") as the input of a fully connected layer to generate a predicted representation for the next word ("on"). We employ noise-contrastive estimation (NCE) (Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013) to compute the cost: the model learns to discriminate between true next words and noise words. NCE allows us to fit unnormalized models making the training time effectively independent of the vocabulary size.

In experiments, CNN-LM is trained on unlabeled MSRP data and an additional 100,000 sentences from English Gigaword (Graff et al., 2003). In principle, sentences from any source, not just English Gigaword, can be used to train this model. In NCE, 20 noise words are sampled for each true example.

So training has two parts: unsupervised, CNN-LM (Figure 3) and supervised, Bi-CNN-MI (Figure 1). In the first phase, the unsupervised training phase, we adopt a language modeling approach because it does not require human labels and can use large corpora to pretrain word embeddings and convolution weights. The goal is to learn sentence features that are unbiased and reflect useful attributes of the input sentence. More importantly, pretraining is useful to relieve overfitting, which is a severe problem when building deep NNs on small corpora like MSRP (cf. Hu et al. (2014)).

In the second phase, the supervised training phase, pretrained word embeddings and convolution weights are tuned for optimal performance on PI.

In CNN-LM, we have combined several architectural elements to pretrain a high-quality sentence analysis NN despite the lack of training data. (i) Similar to PV-DM (Le and Mikolov, 2014), we integrate global context (CNN-SM) and local context (the history of size h) into one model – although our global context consists only of a sentence, not of a paragraph or document. (ii) Similar to work on autoencoding (Vincent et al., 2010), the output that is to be predicted is part of the input. Autoencoding is a successful approach to learning representations and we adapt it here to pretrain good sentence representations. (iii) A second successful approach to learning embeddings is neural network language modeling (Bengio et al., 2003; Mikolov, 2012). Again, we adopt this by including in CNN-LM an ngram language modeling part to predict the next word. The great advantage of this type of embedding learning is that no labels are needed. (iv) CNN-LM only adds one hidden layer over CNN-SM. It keeps simple architecture like PV-DM (Le and Mikolov, 2014), CBOW (Mikolov et al., 2013) and LBL (Mnih and Teh, 2012), enabling the CNN-SM as main training target.

In summary, the key contribution of CNN-LM is that we pretrain convolutional filters. Architectural elements from the literature are combined to support effective pretraining of convolutional filters.

6 Experiments

6.1 Data set and evaluation metrics

We use the Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2004; Das and Smith, 2009). The training set contains 2753 true and 1323 false paraphrase pairs; the test set contains 1147 and 578 pairs, respectively. For each triple (label, S_1, S_2) in the training set we also add (label, S_2, S_1) to make best use of the training data; these additions are nonredundant because the interaction feature matrices (Section 4.1) are asymmetric. Systems are evaluated by accuracy and F_1.

6.2 Paraphrase detection systems

Since we want to show that Bi-CNN-MI performs better than previous NN work, we compare with three NN approaches: NLM, ARC and RAE (Table 1).[2] We also include the majority baseline ("baseline") and MT (Madnani et al., 2012). **RAE** (Socher et al., 2011) and **MT** were discussed in Sections 1 and 2. We now briefly describe the other prior work.

Blacoe and Lapata (2012) compute the vector representation of a sentence from the neural language model (NLM) embeddings (computed based on (Collobert and Weston, 2008)) of the words of the sentence as the sum of the word embeddings (**NLM+**), as the element-wise multiplication of the word embeddings (**NLM**$\odot$), or by means of the recursive autoencoder (**NLM_RAE**, Socher et al. (2011)). The representations of the two paraphrase candidates are then concatenated as input to an SVM classifier. See Blacoe and Lapata (2012) for details.

The ARC model (Hu et al., 2014) is a convolutional architecture similar to (Collobert and Weston, 2008). **ARC-I** is a Siamese architecture in which two shared-weight convolutional sentence models are trained on the binary paraphrase detection task. Hu et al. (2014) find that ARC-I is suboptimal in that it defers the interaction between S_1 and S_2 to the very end of processing: only after the vectors representing S_1 and S_2 have been computed does an interaction occur. To remedy this problem, they propose **ARC-II** in which the Siamese architecture is replaced by a multilayer NN that processes a single representation produced by interleaving S_1 and S_2.

We also evaluate **Bi-CNN-MI–**, an NN identical to Bi-CNN-MI, except that it is not pretrained in unsupervised training.

6.3 Results

Table 1 shows that Bi-CNN-MI outperforms all other systems. The comparison with Bi-CNN-MI– indicates that this is partly due to one major innovation we introduced: unsupervised pretraining. Bi-CNN-MI–, the model without unsupervised pretraining, performs badly. Thus, unsupervised training is helpful to pretrain parameters in paraphrase

[2]A reviewer suggests an additional experiment to directly evaluate the importance of multigranularity: a "system that puts all unigrams, short ngrams, long ngrams, and sentence representations into one interaction matrix." This would indeed be an interesting baseline, but there is no obvious way to conduct this experiment since vectors from different levels are not comparable; e.g., they have different dimensionality.

method	acc	F_1
baseline	66.5	79.9
NLM+	69.0	80.1
NLM$\odot$	67.8	79.3
NLM_RAE	70.3	81.3
ARC-I	69.6	80.3
ARC-II	69.9	80.9
RAE	76.7	83.6
MT	77.4	84.1
Bi-CNN-MI–	72.5	81.4
Bi-CNN-MI	**78.1**	**84.4**

Table 1: Performance of different systems on MSRP

	features used	acc	F_1
1	no features	66.5	79.9
2	+ u: unigram	68.4	79.7
3	+ sn: short ngram	75.3	82.8
4	+ ln: long ngram	76.2	83.1
5	+ s: sentence	73.4	82.3
6	– u: unigram	77.8	84.3
7	– sn: short ngram	76.3	83.5
8	– ln: long ngram	75.6	83.2
9	– s: sentence	77.6	84.2
10	all features	78.1	84.4

Table 2: Analysis of impact of the four feature classes. Line 1: majority baseline. Line 10: Bi-CNN-MI result from Table 1. Lines 2–5: Bi-CNN-MI when only one feature class is used. Line 6–9: ablation experiment: on each line one feature class is removed.

detection, especially when the training set is small. RAE also uses pretraining, but not as effectively as Bi-CNN-MI as Table 1 indicates. Hu et al. (2014) also suggest that training complex NNs only with supervised training runs the risk of overfitting on the small MSRP corpus.

Table 2 looks at the relative importance of the four feature matrices shown in Figure 1. (The unsupervised part of the training regime is not changed for this experiment.) The results indicate that levels sn and ln are most informative: F_1 scores are highest if only these two levels are used (lines 3&4: 82.8, 83.1) and performance drops most when they are removed (lines 7&8: 83.5, 83.2).

Unigrams contribute little to overall performance (lines 2&6), probably because the paraphrases in the corpus typically do not involve individual words (replacing one word by its synonym); rather, the paraphrase relationship involves larger context, which can only be judged by the higher-level features.

Just using the sentence matrix by itself performs well (line 5), but less well than using only levels sn or ln (lines 3&4). Most prior NN work on PI has taken the sentence-level approach. Our results indicate that combining this with the more fine-grained comparison on the ngram-level is superior.

Removing the sentence matrix results in a small drop in performance (line 9). The reason is that sentence representations are computed by k-max pooling from level ln. Thus, we can roughly view the sentence-level feature matrix $\mathbf{F}_\mathrm{s}$ as a subset of $\mathbf{F}_\mathrm{ln}$.

Adding (Madnani et al., 2012)'s MT metrics as input to the Bi-CNN-MI logistic regression further improves performance: accuracy of 78.4 and F_1 of 84.6.

7 Conclusion and future work

We presented the deep learning architecture Bi-CNN-MI for paraphrase identification (PI). Based on the insight that PI requires comparing two sentences *on multiple levels of granularity*, we learn multigranular sentence representations using convolution and compute interaction feature matrices at each level. These matrices are then the input to a logistic classifier for PI. All parameters of the model (for embeddings, convolution and classification) are directly optimized for PI. To address the lack of training data, we pretrain the network in a novel way for a language modeling task. Results on MSRP are state of the art.

In the future, we plan to apply Bi-CNN-MI to sentence matching, question answering and other tasks.

Acknowledgments

We are grateful to Thang Vu, Irina Sergienya and Sebastian Ebert for comments on earlier versions of this paper. This work was supported by Baidu (through a Baidu scholarship awarded to Wenpeng Yin) and by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

References

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2014. Exploiting discourse information to identify paraphrases. *Expert Systems with Applications*, 41(6):2832–2841.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

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.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

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(04):669–688.

Yee Seng Chan and Hwee Tou Ng. 2008. Maxsim: A maximum similarity metric for machine translation evaluation. In *ACL*, pages 55–62.

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 ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 250–253.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

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*, page 350.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Nizar Habash and Ahmed Elkholy. 2008. Sepia: surface span extension to syntactic dependency precision-based mt evaluation. In *Proceedings of the NIST metrics for machine translation workshop at the association for machine translation in the Americas conference, AMTA-2008. Waikiki, HI*.

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*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th international conference on Machine learning*.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 NAACL-HLT*, pages 182–190.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

N Neverova, C Wolf, GW Taylor, and F Nebout. 2014. Multi-scale deep learning for gesture detection and localization. In *European Conference on Computer Vision (ECCV) 2014 ChaLearn Workshop. Zurich*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318.

Steven Parker. 2008. Badger: A new machine translation metric. *Metrics for Machine Translation Challenge*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 153–162. ACM.