Bilateral Multi-Perspective Matching for Natural Language Sentences

Zhiguo Wang, Wael Hamza, Radu Florian
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{zhigwang,whamza,raduf}@us.ibm.com

Abstract

Natural language sentence matching is a fundamental technology for a variety of tasks. Previous approaches either match sentences from a single direction or only apply single granular (word-by-word or sentence-by-sentence) matching. In this work, we propose a bilateral multi-perspective matching (BiMPM) model. Given two sentences P and Q, our model first encodes them with a BiLSTM encoder. Next, we match the two encoded sentences in two directions $P \rightarrow Q$ and $P \leftarrow Q$. In each matching direction, each time step of one sentence is matched against all time-steps of the other sentence from multiple perspectives. Then, another BiLSTM layer is utilized to aggregate the matching results into a fix-length matching vector. Finally, based on the matching vector, the decision is made through a fully connected layer. We evaluate our model on three tasks: paraphrase identification, natural language inference and answer sentence selection. Experimental results on standard benchmark datasets show that our model achieves the state-of-the-art performance on all tasks.

1 Introduction

Natural language sentence matching (NLSM) is the task of comparing two sentences and identifying the relationship between the two sentences. It is a fundamental technology for a variety of tasks. For example, in paraphrase identification task, NLSM is used to determine whether two sentences are paraphrase or not [Yin *et al.*, 2015]. For natural language inference task, NLSM is utilized to judge whether a hypothesis sentence can be inferred from a premise sentence [Bowman *et al.*, 2015]. For question answering and information retrieval tasks, NLSM is employed to assess the relevance between query-answer pairs and rank all the candidate answers [Wang *et al.*, 2016d]. For machine comprehension task, NLSM is used for matching passage with the question and pointing out the correct answer span [Wang *et al.*, 2016b].

With the renaissance of neural network models, two types of deep learning frameworks were proposed for NLSM. The **first** framework is based on the "Siamese" architecture [Bromley *et al.*, 1993]. In this framework, the same neural network encoder (such as CNN or RNN) is applied to two input sentences individually, so that both of the two sentences are encoded into sentence vectors in the same embedding space. Then, a matching decision is made solely based on the two sentence vectors [Bowman *et al.*, 2015; Tan *et al.*, 2015]. The advantage of this framework is that sharing parameters makes the model smaller and easier to train, and the sentence vectors can be used for visualization, sentence clustering and many other purposes [Wang *et al.*, 2016c]. But the disadvantage is that there is no interaction between the two sentences during the encoding procedure, which may lose some important information. To deal with this problem, the **second** framework "matching-aggregation" is proposed [Wang and Jiang, 2016; Wang *et al.*, 2016d]. Under this framework, smaller units (such as words or contextual vectors) of the two sentences are firstly matched, then the matching results are aggregated (by CNN or LSTM) into a vector to make the final decision. The new framework captures more interactive features between the two sentences, therefore it acquires significant improvements. However, the previous "matching-aggregation" approaches still have some limitations. First, some of the approaches only explored the word-by-word matching [Rocktäschel *et al.*, 2015], but ignored other granular matchings (e.g., phrase-by-sentence); Second, the matching is only performed in a single direction (e.g., matching P against Q) [Wang and Jiang, 2015], but neglected the reverse direction (e.g., matching Q against P).

We address this by attentio n and a second lstm

In this paper, to tackle these limitations, we propose a bilateral multi-perspective matching (BiMPM) model for NLSM tasks. Our model essentially belongs to the "matching-aggregation" framework. Given two sentences P and Q, our model first encodes them with a bidirectional Long Short-Term Memory Network (BiLSTM). Next, we match the two encoded sentences in two directions $P \rightarrow Q$ and $P \leftarrow Q$. In each matching direction, let's say $P \rightarrow Q$, each time step of Q is matched against all time-steps of P from multiple perspectives. Then, another BiLSTM layer is utilized to aggregate the matching results into a fix-length matching vector. Finally, based on the matching vector, the decision is made through a fully connected layer. We evaluate our model on three NLSM tasks: paraphrase identification, natural language inference and answer sentence selection. Experimental results on some standard benchmark datasets show that our model achieves the state-of-the-art performance on all tasks.

In following parts, we start with a brief definition of the NLSM task (Section 2), followed by the details of our model (Section 3). Then we evaluate our model on some standard benchmark datasets (Section 4).

2 Task Definition

Formally, we can represent each example of the NLSM task as a triple (P, Q, y), where $P = (p_1, ..., p_j, ..., p_M)$ is a sentence with a length M, $Q = (q_1, ..., q_i, ..., q_N)$ is the second sentence with a length N, $y \in \mathcal{Y}$ is the label representing the relationship between P and Q, and $\mathcal{Y}$ is a set of task-specific labels. The NLSM task can be represented as estimating the conditional probability $\Pr(y|P, Q)$ based on the training set, and predicting the relationship for testing examples by $y^* = \arg\max_{y \in \mathcal{Y}} \Pr(y|P, Q)$. Concretely, for paraphrase identification task, P and Q are two sentences, $\mathcal{Y} = \{0, 1\}$, where $y = 1$ means P and Q are paraphrase of each other, and $y = 0$ otherwise. For natural language inference task, P is a premise sentence, Q is a hypothesis sentence, and $\mathcal{Y} = \{entailment, contradiction, neutral\}$ where $entailment$ indicates Q can be inferred from P, $contradiction$ indicates Q cannot be true condition on P, and $neutral$ means P and Q are irrelevant to each other. For answer sentence selection task, P is a question, Q is a candidate answer, and $\mathcal{Y} = \{0, 1\}$ where $y = 1$ means Q is a correct answer for P, and $y = 0$ otherwise.

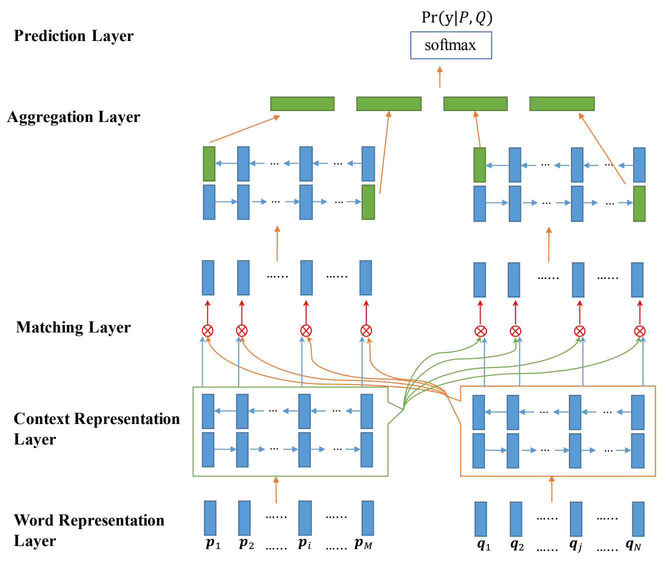
3 Method

3.1 Model Overview

In this section, we propose a bilateral multi-perspective matching (BiMPM) model to estimate the probability distribution $\Pr(y|P, Q)$. Our model belongs to the "matching-aggregation" framework [Wang and Jiang, 2016]. But different from the previous "matching-aggregation" approaches, our model matches P and Q in two directions ($P \rightarrow Q$ and $P \leftarrow Q$). And, in each individual direction, our model matches the two sentences from multiple perspectives. Figure 1 shows the architecture of our model. Given a pair of sentences P and Q, the BiMPM model estimates the probability distribution $\Pr(y|P, Q)$ through the following five layers.

Word Representation Layer. The goal of this layer is to represent each word in P and Q with a d-dimensional vector. We construct the d-dimensional vector with two components: word embeddings and character-composed embeddings. The word embedding is a fixed vector for each individual word, which is pre-trained with GloVe [Pennington *et al.*, 2014] or word2vec [Mikolov *et al.*,]. The character-composed embedding is calculated by feeding each character (also represented as a vector) within a word into a Long Short-Term Memory Network (LSTM) [Hochreiter and Schmidhuber, 1997]. The output of this layer are two sequence of word vectors $P : [\boldsymbol{p}_1, ..., \boldsymbol{p}_M]$ and $Q : [\boldsymbol{q}_1, ..., \boldsymbol{q}_N]$.

Context Representation Layer. The purpose of this layer is to incorporate contextual information into the representation of each time step of P and Q. We utilize a bi-directional LSTM (BiLSTM) to encode contextual embeddings for each



Pr(y|P,Q)
softmax
Prediction Layer
Aggregation Layer
Matching Layer
Context Representation Layer
Word Representation Layer
p_1 p_2 ...... p_i ...... p_M
q_1 q_2 ...... q_j ...... q_N

Figure 1: Architecture for Bilateral Multi-Perspective Matching (BiMPM) Model, where $\otimes$ is the multi-perspective matching operation described in sub-section 3.2.

time-step of P.

$$\overrightarrow{\boldsymbol{h}}_i^p = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}^p, \boldsymbol{p}_i) \qquad i = 1, ..., M$$
$$\overleftarrow{\boldsymbol{h}}_i^p = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}^p, \boldsymbol{p}_i) \qquad i = M, ..., 1 \qquad (1)$$

Meanwhile, we apply the same BiLSTM to encode Q:

$$\overrightarrow{\boldsymbol{h}}_j^q = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{j-1}^q, \boldsymbol{q}_j) \qquad j = 1, ..., N$$
$$\overleftarrow{\boldsymbol{h}}_j^q = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{j+1}^q, \boldsymbol{q}_j) \qquad j = N, ..., 1 \qquad (2)$$

Matching Layer. This is the core layer within our model. The goal of this layer is to compare each contextual embedding (time-step) of one sentence against all contextual embeddings (time-steps) of the other sentence. As shown in Figure 1, we will match the two sentences P and Q in two directions: match each time-step of P against all time-steps of Q, and match each time-step of Q against all time-steps of P. To match one time-step of a sentence against all time-steps of the other sentence, we design a multi-perspective matching operation $\otimes$. We will give more details about this operation in sub-section 3.2. The output of this layer are two sequence of matching vectors (right above the operation $\otimes$ in Figure 1), where each matching vector corresponds to the matching result of one time-step against all time-steps of the other sentence.

Aggregation Layer. This layer is employed to aggregate the two sequence of matching vectors into a fix-length matching vector. We utilize another BiLSTM model, and apply it to the two sequence of matching vectors individually. Then, we construct the fix-length matching vector by concatenating (the four green) vectors from the last time-step of the BiLSTM models.

Prediction Layer. The purpose of this layer is to evaluate the probability distribution $\Pr(y|P, Q)$. To this end, we employ a two layer feed-forward neural network to consume the

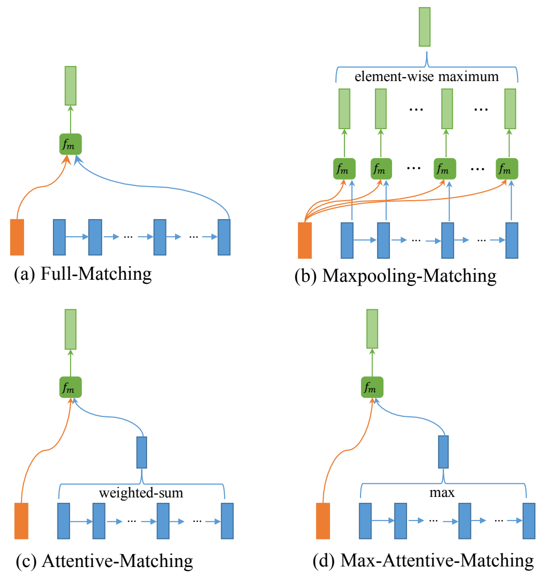


Figure 2: Diagrams for different matching strategies, where f_m is the multi-perspective cosine matching function in Eq.(3), the input includes one time step of one sentence (left orange block) and all the time-steps of the other sentence (right blue blocks), and the output is a vector of matching values (top green block) calculated by Eq.(3).

fix-length matching vector, and apply the $softmax$ function in the output layer. The number of nodes in the output layer is set based on each specific task described in Section 2.

3.2 Multi-perspective Matching Operation

We define the multi-perspective matching operation $\otimes$ in following two steps:

First, we define a multi-perspective cosine matching function f_m to compare two vectors

$$\boldsymbol{m} = f_m(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{W}) \qquad (3)$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two d-dimensional vectors, $\boldsymbol{W} \in \Re^{l \times d}$ is a trainable parameter, l is the number of perspectives, and the returned value $\boldsymbol{m}$ is a l-dimensional vector $\boldsymbol{m} = [m_1, ..., m_k, ..., m_l]$. Each element $m_k \in \boldsymbol{m}$ is a matching value from the k-th perspective, and it is calculated by the cosine similarity between two weighted vectors

$$m_k = cosine(W_k \circ \boldsymbol{v}_1, W_k \circ \boldsymbol{v}_2) \qquad (4)$$

where $\circ$ is the element-wise multiplication, and W_k is the k-th row of $\boldsymbol{W}$, which controls the k-th perspective and assigns different weights to different dimensions of the d-dimensional space.

Second, based on f_m, we define four matching strategies to compare each time-step of one sentence against all time-steps of the other sentence. To avoid repetition, we only define these matching strategies for one matching direction $P \rightarrow Q$. The readers can infer equations for the reverse direction easily.

(1) Full-Matching. Figure 2 (a) shows the diagram of this matching strategy. In this strategy, each forward (or backward) contextual embedding $\overrightarrow{\boldsymbol{h}}_i^p$ (or $\overleftarrow{\boldsymbol{h}}_i^p$) is compared with the last time step of the forward (or backward) representation of the other sentence $\overrightarrow{\boldsymbol{h}}_N^q$ (or $\overleftarrow{\boldsymbol{h}}_1^q$).

$$\begin{aligned} \overrightarrow{\boldsymbol{m}}_i^{full} &= f_m(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_N^q; \boldsymbol{W}^1) \\ \overleftarrow{\boldsymbol{m}}_i^{full} &= f_m(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_1^q; \boldsymbol{W}^2) \end{aligned} \qquad (5)$$

(2) Maxpooling-Matching. Figure 2 (b) gives the diagram of this matching strategy. In this strategy, each forward (or backward) contextual embedding $\overrightarrow{\boldsymbol{h}}_i^p$ (or $\overleftarrow{\boldsymbol{h}}_i^p$) is compared with every forward (or backward) contextual embeddings of the other sentence $\overrightarrow{\boldsymbol{h}}_j^q$ (or $\overleftarrow{\boldsymbol{h}}_j^q$) for $j \in (1...N)$, and only the maximum value of each dimension is retained.

$$\begin{aligned} \overrightarrow{\boldsymbol{m}}_i^{max} &= \max_{j \in (1...N)} f_m(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_j^q; \boldsymbol{W}^3) \\ \overleftarrow{\boldsymbol{m}}_i^{max} &= \max_{j \in (1...N)} f_m(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_j^q; \boldsymbol{W}^4) \end{aligned} \qquad (6)$$

(3) Attentive-Matching. Figure 2 (c) shows the diagram of this matching strategy. We first calculate the cosine similarities between each forward (or backward) contextual embedding $\overrightarrow{\boldsymbol{h}}_i^p$ (or $\overleftarrow{\boldsymbol{h}}_i^p$) and every forward (or backward) contextual embeddings of the other sentence $\overrightarrow{\boldsymbol{h}}_j^q$ (or $\overleftarrow{\boldsymbol{h}}_j^q$):

Cosine vs dot product

$$\begin{aligned} \overrightarrow{\alpha}_{i,j} &= cosine(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_j^q) \qquad j = 1, ..., N \\ \overleftarrow{\alpha}_{i,j} &= cosine(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_j^q) \qquad j = 1, ..., N \end{aligned} \qquad (7)$$

Then, we take $\overrightarrow{\alpha}_{i,j}$ (or $\overleftarrow{\alpha}_{i,j}$) as the weight of $\overrightarrow{\boldsymbol{h}}_j^q$ (or $\overleftarrow{\boldsymbol{h}}_j^q$), and calculate an attentive vector for the entire sentence Q by weighted summing all the contextual embeddings of Q:

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_i^{mean} &= \frac{\sum_{j=1}^N \overrightarrow{\alpha}_{i,j} \cdot \overrightarrow{\boldsymbol{h}}_j^q}{\sum_{j=1}^N \overrightarrow{\alpha}_{i,j}} \\ \overleftarrow{\boldsymbol{h}}_i^{mean} &= \frac{\sum_{j=1}^N \overleftarrow{\alpha}_{i,j} \cdot \overleftarrow{\boldsymbol{h}}_j^q}{\sum_{j=1}^N \overleftarrow{\alpha}_{i,j}} \end{aligned} \qquad (8)$$

Finally, we match each forward (or backward) contextual embedding of $\overrightarrow{\boldsymbol{h}}_i^p$ (or $\overleftarrow{\boldsymbol{h}}_i^p$) with its corresponding attentive vector:

$$\begin{aligned} \overrightarrow{\boldsymbol{m}}_i^{att} &= f_m(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_i^{mean}; \boldsymbol{W}^5) \\ \overleftarrow{\boldsymbol{m}}_i^{att} &= f_m(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_i^{mean}; \boldsymbol{W}^6) \end{aligned} \qquad (9)$$

(4) Max-Attentive-Matching. Figure 2 (d) shows the diagram of this matching strategy. This strategy is similar to the

Attentive-Matching strategy. However, instead of taking the weighed sum of all the contextual embeddings as the attentive vector, we pick the contextual embedding with the highest cosine similarity as the attentive vector. Then, we match each contextual embedding of the sentence P with its new attentive vector.

We apply all these four matching strategies to each time-step of the sentence P, and concatenate the generated eight vectors as the matching vector for each time-step of P. We also perform the same process for the reverse matching direction.

4 Experiments

In this section, we evaluate our model on three tasks: paraphrase identification, natural language inference and answer sentence selection. We will first introduce the general setting of our BiMPM models in sub-section 4.1. Then, we demonstrate the properties of our model through some ablation studies in sub-section 4.2. Finally, we compare our model with state-of-the-art models on some standard benchmark datasets in sub-section 4.3, 4.4 and 4.5.

4.1 Experiment Settings

We initialize the word embeddings in the word representation layer with the 300-dimensional GloVe word vectors pretrained from the 840B Common Crawl corpus [Pennington *et al.*, 2014]. For the out-of-vocabulary (OOV) words, we initialize the word embeddings randomly. For the character-composed embeddings, we initialize each character as a 20-dimensional vector, and compose each word into a 50-dimensional vector with a LSTM layer. We set the hidden size as 100 for all BiLSTM layers, and set the number of perspectives l of our multi-perspective cosine matching function (Equation (3)) as 20. We apply dropout to every layers in Figure 1, and set the dropout ratio as 0.1. To train the model, we minimize the cross entropy of training set, and use the ADAM optimizer [Kingma and Ba, 2014] to update parameters. We set the learning rate as 0.001. During training, we do not update the pre-trained word embeddings. For all the experiments, we pick the model which works the best on the dev set, and then evaluate it on the test set.

4.2 Model Properties

To demonstrate the properties of our model, we choose the paraphrase identification task, and experiment on the "Quora Question Pairs" dataset [1]. This dataset consists of over 400,000 question pairs, and each question pair is annotated with a binary value indicating whether the two questions are paraphrase of each other. We randomly select 5,000 paraphrases and 5,000 non-paraphrases as the dev set, and sample another 5,000 paraphrases and 5,000 non-paraphrases as the test set. We keep the remaining instances as the training set [2].

First, we study the influence of our multi-perspective cosine matching function in Eq.(3). We varied the number of perspectives l among $\{1, 5, 10, 15, 20\}$[3], and kept the other options unchanged. We also build a baseline model by replacing Eq.(3) with the vanilla cosine similarity function. Figure 3 shows the performance curve on the dev set, where $l = 0$ corresponds to the performance of our baseline model. We can see that, even if we only utilize one perspective ($l = 1$), our model gets a significant improvement. When increasing the number of perspectives, the performance improves significantly. Therefore, our multi-perspective cosine matching function is really effective for matching vectors.

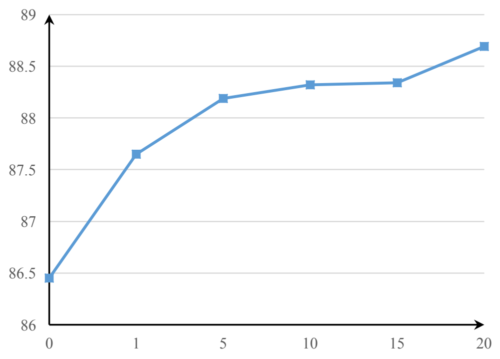


Figure 3: Influence of the multi-perspective cosine matching function in Eq.(3) .

Second, we check if the bilateral matching is really helpful. To this end, we build two ablation models to matching sentences in only a single direction: 1) "Only $P \rightarrow Q$" which only matches P against Q; 2) "Only $P \leftarrow Q$" which only matches Q against P. Table 1 shows the performance on the dev set. Comparing the two ablation models with the "Full Model", we can observe that single direction matching hurts the performance for about 1 percent. Therefore, matching sentences in two directions is really necessary for acquiring better performance.

Third, we evaluate the effectiveness of different matching strategies. To this end, we construct four ablation models (w/o Full-Matching, w/o Maxpooling-Matching, w/o Attentive-Matching, w/o Max-Attentive-Matching) by eliminating a matching strategy at each time. Table 1 shows the performance on the dev set. We can see that eliminating any of the matching strategies would hurt the performance significantly.

4.3 Experiments on Paraphrase Identification

In this sub-section, we compare our model with state-of-the-art models on the paraphrase identification task. We still experiment on the "Quora Question Pairs" dataset, and use the same dataset partition as sub-section 4.2. This dataset is a brand-new dataset, and no previous results have been published yet. Therefore, we implemented three types of baseline models.

First, under the Siamese framework, we implemented two baseline models: "Siamese-CNN" and "Siamese-LSTM".

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2]We will release our dataset partition at https://zhiguowang.github.io/ .

[3]Due to practical limitations, we did not experiment with more perspectives.

Models	Accuracy
Only $P \rightarrow Q$	87.74
Only $P \leftarrow Q$	87.47
w/o Full-Matching	87.86
w/o Maxpooling-Matching	87.64
w/o Attentive-Matching	87.87
w/o MaxAttentive-Matching	87.98
Full Model	**88.69**

Table 1: Ablation studies on the dev set.

Models	Accuracy
Siamese-CNN	79.60
Multi-Perspective-CNN	81.38
Siamese-LSTM	82.58
Multi-Perspective-LSTM	83.21
L.D.C.	85.55
BiMPM	**88.17**

Table 2: Performance for paraphrase identification on Quora dataset.

Both of the two models will encode two input sentences into two sentence vectors with a neural network encoder, and make a decision based on the cosine similarity between the two sentence vectors. But they implement the sentence encoder with CNN or LSTM models respectively. We design the CNN and LSTM models according to the architectures in [Wang *et al.*, 2016c].

Second, based on the two baseline models, we implemented two more baseline models "Multi-Perspective-CNN" and "Multi-Perspective-LSTM". In these two models, we change the cosine similarity calculation layer with our multi-perspective cosine matching function in Eq.(3), and apply a fully-connected layer (with $sigmoid$ function on the top) to make the prediction.

Third, we re-implemented the "L.D.C." model proposed by [Wang *et al.*, 2016d], which is a model under the "matching-aggregation" framework and acquires the state-of-the-art performance on several tasks.

Table 2 shows the performances of all baseline models and our "BiMPM" model. We can see that "Multi-Perspective-CNN" (or "Multi-Perspective-LSTM") works much better than "Siamese-CNN" (or "Siamese-LSTM"), which further indicates that our multi-perspective cosine matching function (Eq.(3)) is very effective for matching vectors. Our "BiMPM" model outperforms "L.D.C." model by more than two percent. Therefore, our model is very effective for paraphrase identification task.

4.4 Experiments on Natural Language Inference

In this sub-section, we evaluate our model for natural language inference task on the SNLI dataset [Bowman *et al.*, 2015]. We tested four variations of our model on this dataset, where "Only $P \rightarrow Q$" and "Only $P \leftarrow Q$" are the single direction matching models described in sub-section 4.2, "BiMPM" is our full model, and "BiMPM (Ensemble)" is an ensemble version of our "BiMPM" model. We design the ensemble model by simply averaging the probability distributions of four "BiMPM" models, and each of the "BiMPM" model has the same architecture but initialize with different seed.

Models	Accuracy
[Bowman *et al.*, 2015]	77.6
[Vendrov *et al.*, 2015]	81.4
[Mou *et al.*, 2015]	82.1
[Rocktäschel *et al.*, 2015]	83.5
[Liu *et al.*, 2016b]	85.0
[Liu *et al.*, 2016a]	85.1
[Wang and Jiang, 2015]	86.1
[Cheng *et al.*, 2016]	86.3
[Parikh *et al.*, 2016]	86.8
[Munkhdalai and Yu, 2016]	87.3
[Sha *et al.*, 2016]	87.5
[Chen *et al.*, 2016] (Single)	87.7
[Chen *et al.*, 2016] (Ensemble)	88.3
Only $P \rightarrow Q$	85.6
Only $P \leftarrow Q$	86.3
BiMPM	86.9
BiMPM (Ensemble)	**88.8**

Table 3: Performance for natural language inference on SNLI dataset.

Table 3 shows the performances of the state-of-the-art models and our models. First, we can see that "Only $P \leftarrow Q$" works significantly better than "Only $P \rightarrow Q$", which tells us that, for natural language inference, matching hypothesis against the premise is more effective than the other way around. Second, our "BiMPM" model works much better than "Only $P \leftarrow Q$", which reveals that matching premise against the hypothesis can also bring some benefits. Finally, comparing our models with all the state-of-the-art models, we can observe that our single model "BiMPM" is on par with the state-of-the-art single models, and our 'BiMPM (Ensemble)" works much better than "[Chen *et al.*, 2016] (Ensemble)". Therefore, our models achieve the state-of-the-art performance in both single and ensemble scenarios for natural language inference task.

4.5 Experiments on Answer Sentence Selection

In this sub-section, we study the effectiveness of our model for answer sentence selection tasks. The answer sentence selection task is to rank a list of candidate answer sentences based on their similarities to the question, and the performance is measured by mean average precision (MAP) and mean reciprocal rank (MRR). We experiment on two datasets: TREC-QA [Wang *et al.*, 2007] and WikiQA [Yang *et al.*, 2015]. Experimental results of the state-of-the-art models [4] and our "BiMPM" model are listed in Table 4, where the

[4][Rao *et al.*, 2016] pointed out that there are two versions of TREC-QA dataset: raw-version and clean-version. In this work,

Models	TREC-QA		WikiQA	
	MAP	MRR	MAP	MRR
[Yang *et al.*, 2015]	0.695	0.763	0.652	0.665
[Tan *et al.*, 2015]	0.728	0.832	–	–
Wang and Itty. [2015]	0.746	0.820	–	–
[Santos *et al.*, 2016]	0.753	0.851	0.689	0.696
[Yin *et al.*, 2015]	–	–	0.692	0.711
[Miao *et al.*, 2016]	–	–	0.689	0.707
[Wang *et al.*, 2016d]	0.771	0.845	0.706	0.723
[He and Lin, 2016]	0.777	0.836	0.709	0.723
[Rao *et al.*, 2016]	0.801	**0.877**	0.701	0.718
[Wang *et al.*, 2016a]	–	–	0.734	0.742
[Wang and Jiang, 2016]	–	–	**0.743**	**0.755**
BiMPM	**0.802**	0.875	0.718	0.731

Table 4: Performance for answer sentence selection on TREC-QA and WikiQA datasets.

performances were evaluated with the standard trec_eval-8.0 script [5]. We can see that the performance from our model is on par with the state-of-the-art models. Therefore, our model is also effective for answer sentence selection tasks.

5 Related Work

Natural language sentence matching (NLSM) has been studied for many years. The early approaches interested in designing hand-craft features to capture n-gram overlapping, word reordering and syntactic alignments phenomena [Heilman and Smith, 2010; Wang and Ittycheriah, 2015]. This kind of method can work well on a specific task or dataset, but it's hard to generalize well to other tasks.

With the availability of large-scale annotated datasets [Bowman *et al.*, 2015], many deep learning models were proposed for NLSM. The first kind of framework is based the Siamese architecture [Bromley *et al.*, 1993], where sentences are encoded into sentence vectors based on some neural network encoder, and then the relationship between two sentences was decided solely based on the two sentence vectors [Bowman *et al.*, 2015; Yang *et al.*, 2015; Tan *et al.*, 2015]. However, this kind of framework ignores the fact that the lower level interactive features between two sentences are indispensable. Therefore, many neural network models were proposed to match sentences from multiple level of granularity [Yin *et al.*, 2015; Wang and Jiang, 2016; Wang *et al.*, 2016d]. Experimental results on many tasks have proofed that the new framework works significantly better than the previous methods. Our model also belongs to this framework, and it has shown the effectiveness in Section 4.

6 Conclusion

In this work, we propose a bilateral multi-perspective matching (BiMPM) model under the "matching-aggregation" framework. Different from the previous "matching-aggregation" approaches, our model matches sentences P and Q in two directions ($P \rightarrow Q$ and $P \leftarrow Q$). And, in each individual direction, our model matches the two sentences from multiple perspectives. We evaluated our model on three tasks: paraphrase identification, natural language inference and answer sentence selection. Experimental results on standard benchmark datasets show that our model achieves the state-of-the-art performance on all tasks.

we utilized the clean-version. Therefore, we only compare with approaches reporting performance on this dataset.

[5] http://trec.nist.gove/trec_eval/

References

[Bowman *et al.*, 2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[Bromley *et al.*, 1993] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

[Chen *et al.*, 2016] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[Cheng *et al.*, 2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[He and Lin, 2016] Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL*, 2016.

[Heilman and Smith, 2010] Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*, 2010.

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

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Liu *et al.*, 2016a] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Modelling interaction of sentence pair with coupled-lstms. *arXiv preprint arXiv:1605.05573*, 2016.

[Liu *et al.*, 2016b] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*, 2016.

[Miao *et al.*, 2016] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *ICML*, 2016.

[Mikolov *et al.*,] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*.

[Mou *et al.*, 2015] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*, 2015.

[Munkhdalai and Yu, 2016] Tsendsuren Munkhdalai and Hong Yu. Neural tree indexers for text understanding. *arXiv preprint arXiv:1607.04492*, 2016.

[Parikh *et al.*, 2016] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[Rao *et al.*, 2016] Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*, 2016.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[Santos *et al.*, 2016] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

[Sha *et al.*, 2016] Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *COLING*, 2016.

[Tan *et al.*, 2015] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[Vendrov *et al.*, 2015] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015.

[Wang and Ittycheriah, 2015] Zhiguo Wang and Abraham Ittycheriah. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*, 2015.

[Wang and Jiang, 2015] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[Wang and Jiang, 2016] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[Wang *et al.*, 2007] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*, 2007.

[Wang *et al.*, 2016a] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *ACL*, 2016.

[Wang *et al.*, 2016b] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*, 2016.

[Wang *et al.*, 2016c] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Semi-supervised clustering for short text via deep representation learning. In *CoNLL*, 2016.

[Wang *et al.*, 2016d] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. In *COLING*, 2016.

[Yang *et al.*, 2015] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*, 2015.

[Yin *et al.*, 2015] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.