Deep Fusion LSTMs for Text Semantic Matching

Pengfei Liu, Xipeng Qiu*, Jifan Chen, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,jfchen14,xjhuang}@fudan.edu.cn

Abstract

Recently, there is rising interest in modelling the interactions of text pair with deep neural networks. In this paper, we propose a model of deep fusion LSTMs (DF-LSTMs) to model the strong interaction of text pair in a recursive matching way. Specifically, DF-LSTMs consist of two interdependent LSTMs, each of which models a sequence under the influence of another. We also use external memory to increase the capacity of LSTMs, thereby possibly capturing more complicated matching patterns. Experiments on two very large datasets demonstrate the efficacy of our proposed architecture. Furthermore, we present an elaborate qualitative analysis of our models, giving an intuitive understanding how our model worked.

1 Introduction

Among many natural language processing (NLP) tasks, such as text classification, question answering and machine translation, a common problem is modelling the relevance/similarity of a pair of texts, which is also called text semantic matching. Due to the semantic gap problem, text semantic matching is still a challenging problem.

Recently, deep learning is rising a substantial interest in text semantic matching and has achieved some great progresses (Hu et al., 2014; Qiu and Huang, 2015; Wan et al., 2016). According to their interaction ways, previous models can be classified into three categories:

Weak interaction Models Some early works focus on sentence level interactions, such as ARC-I(Hu et al., 2014), CNTN(Qiu and Huang, 2015) and so on. These models first encode two sequences into continuous dense vectors by separated neural models, and then compute the matching score based on sentence encoding. In this paradigm, two sentences have no interaction until arriving final phase.

Semi-interaction Models Another kind of models use soft attention mechanism to obtain the representation of one sentence by depending on representation of another sentence, such as ABCNN (Yin et al., 2015), Attention LSTM (Rocktäschel et al., 2015; Hermann et al., 2015). These models can alleviate the weak interaction problem to some extent.

Strong Interaction Models Some models build the interaction at different granularity (word, phrase and sentence level), such as ARC-II (Hu et al., 2014), MultiGranCNN (Yin and Schütze, 2015), Multi-Perspective CNN (He et al., 2015), MV-LSTM (Wan et al., 2016), MatchPyramid (Pang et al., 2016). The final matching score depends on these different levels of interactions.

In this paper, we adopt a deep fusion strategy to model the strong interactions of two sentences. Given two texts $x_{1:m}$ and $y_{1:n}$, we define a matching vector $\mathbf{h}_{i,j}$ to represent the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$. $\mathbf{h}_{i,j}$ depends on the matching vectors $\mathbf{h}_{s,t}$ on previous interactions $1 \leq s < i$ and $1 \leq t < j$. Thus, text matching can be regarded as modelling the interaction of two texts in a recursive matching way.

Following this idea, we propose deep fusion long short-term memory neural networks (DF-LSTMs) to model the interactions recursively. More concretely, DF-LSTMs consist of two interconnected conditional LSTMs, each of which models a piece of text under the influence of another. The output vector of DF-LSTMs is fed into a task-specific output layer to compute the match-

*Corresponding author

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1034–1043,
Berlin, Germany, August 7-12, 2016.

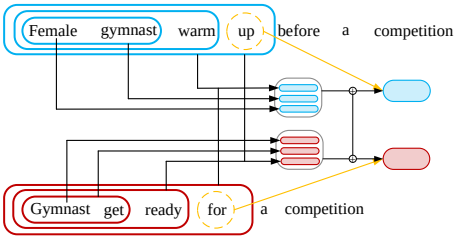


Figure 1: A motivated example to illustrate our recursive composition mechanism.

ing score.

The contributions of this paper can be summarized as follows.

1. Different with previous models, DF-LSTMs model the strong interactions of two texts in a recursive matching way, which consist of two inter- and intra-dependent LSTMs.

2. Compared to the previous works on text matching, we perform extensive empirical studies on two very large datasets. Experiment results demonstrate that our proposed architecture is more effective.

3. We present an elaborate qualitative analysis of our model, giving an intuitive understanding how our model worked.

2 Recursively Text Semantic Matching

To facilitate our model, we firstly give some definitions.

Given two sequences $X = x_1, x_2, \cdots, x_m$ and $Y = y_1, y_2, \cdots, y_n$, most deep neural models try to represent their semantic relevance by a **matching vector** $\mathbf{h}(X, Y)$, which is followed by a score function to calculate the matching score.

The weak interaction methods decompose matching vector by $\mathbf{h}(X, Y) = f(\mathbf{h}(X), \mathbf{h}(Y))$, where function $f(\cdot)$ may be one of some basic operations or the combination of them: concatenation, affine transformation, bilinear, and so on.

In this paper, we propose a strong interaction of two sequences to decompose matching vector $\mathbf{h}(X, Y)$ in a recursive way. We refer to the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$ as $\mathbf{h}_{i,j}(X, Y)$, which depends on previous interactions $\mathbf{h}_{s,t}(X, Y)$ for $1 \leq s < i$ and $1 \leq t < j$.

Figure 1 gives an example to illustrate this. For sentence pair X ="`Female gymnast warm up before a competition`", Y="`Gymnast get ready for a competition`", considering the interaction ($\mathbf{h}_{4,4}$) between $x_{1:4}$ = "`Female gymnast warm up`" and $y_{1:4}$ = "`Gymnast get ready for`", which is composed by the interactions between their subsequences $(\mathbf{h}_{1,4}, \cdots, \mathbf{h}_{3,4}, \mathbf{h}_{4,1}, \cdots, \mathbf{h}_{4,3})$. We can see that a strong interaction between two sequences can be decomposed in recursive topology structure.

The matching vector $\mathbf{h}_{i,j}(X, Y)$ can be written as

$$\mathbf{h}_{i,j}(X, Y) = \mathbf{h}_{i,j}(X|Y) \oplus \mathbf{h}_{i,j}(Y|X), \quad (1)$$

where $\mathbf{h}_{i,j}(X|Y)$ refers to conditional encoding of subsequence $x_{1:i}$ influenced by $y_{1:j}$. Meanwhile, $\mathbf{h}_{i,j}(Y|X)$ is conditional encoding of subsequence $y_{1:j}$ influenced by subsequence $x_{1:i}$; $\oplus$ is concatenation operation.

These two conditional encodings depend on their history encodings. Based on this, we propose deep fusion LSTMs to model the matching of texts by recursive composition mechanism, which can better capture the complicated interaction of two sentences due to fully considering the interactions between subsequences.

3 Long Short-Term Memory Network

Long short-term memory neural network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies. LSTM maintains a memory cell that updates and exposes its content only when deemed necessary.

While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The elements of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$.

The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}, \quad (2)$$

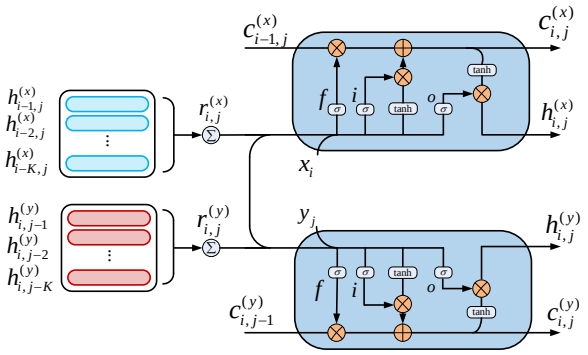


Figure 2: Illustration of DF-LSTMs unit.

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (3)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (4)$$

where $\mathbf{x}_t$ is the input at the current time step; $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Intuitively, the forget gate controls the amount of which each unit of the memory cell is erased, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state.

The update of each LSTM unit can be written precisely as

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t). \quad (5)$$

Here, the function $\mathbf{LSTM}(\cdot,\cdot,\cdot)$ is a shorthand for Eq. (2-4).

LSTM can map the input sequence of arbitrary length to a fixed-sized vector, and has been successfully applied to a wide range of NLP tasks, such as machine translation (Sutskever et al., 2014), language modelling (Sutskever et al., 2011), text matching (Rocktäschel et al., 2015) and text classification (Liu et al., 2015).

4 Deep Fusion LSTMs for Recursively Semantic Matching

To deal with two sentences, one straightforward method is to model them with two separate LSTMs. However, this method is difficult to model local interactions of two sentences.

Following the recursive matching strategy, we propose a neural model of deep fusion LSTMs (DF-LSTMs), which consists of two interdependent LSTMs to capture the inter- and intra-interactions between two sequences. Figure 2 gives an illustration of DF-LSTMs unit.

To facilitate our model, we firstly give some definitions. Given two sequences $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_m$, we let $\mathbf{x}_i \in \mathbb{R}^d$ denotes the embedded representation of the word x_i. The standard LSTM has one temporal dimension. When dealing with a sentence, LSTM regards the position as time step. At position i of sentence $x_{1:n}$, the output $\mathbf{h}_i$ reflects the meaning of subsequence $x_{1:i} = x_1, \cdots, x_i$.

To model the interaction of two sentences in a recursive way, we define $\mathbf{h}_{i,j}$ to represent the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$, which is computed by

$$\mathbf{h}_{i,j} = \mathbf{h}^{(x)}_{i,j} \oplus \mathbf{h}^{(y)}_{i,j}, \quad (6)$$

where $\mathbf{h}^{(x)}_{i,j}$ denotes the encoding of subsequence $x_{1:i}$ in the first LSTM influenced by the output of the second LSTM on subsequence $y_{1:j}$; $\mathbf{h}^{(y)}_{i,j}$ is the encoding of subsequence $y_{1:j}$ in the second LSTM influenced by the output of the first LSTM on subsequence $x_{1:i}$.

More concretely,

$$(\mathbf{h}^{(x)}_{i,j}, \mathbf{c}^{(x)}_{i,j}) = \mathbf{LSTM}(\mathcal{H}_{i,j}, \mathbf{c}^{(x)}_{i-1,j}, \mathbf{x}_i), \quad (7)$$

$$(\mathbf{h}^{(y)}_{i,j}, \mathbf{c}^{(y)}_{i,j}) = \mathbf{LSTM}(\mathcal{H}_{i,j}, \mathbf{c}^{(y)}_{i,j-1}, \mathbf{x}_j), \quad (8)$$

where $\mathcal{H}_{i,j}$ is information consisting of history states before position (i, j).

The simplest setting is $\mathcal{H}_{i,j} = \mathbf{h}^{(x)}_{i-1,j} \oplus \mathbf{h}^{(y)}_{i,j-1}$. In this case, our model can be regarded as grid LSTMs (Kalchbrenner et al., 2015).

However, there are total $m \times n$ interactions in recursive matching process, LSTM could be stressed to keep these interactions in internal memory. Therefore, inspired by recent neural memory network, such as neural Turing machine(Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), we introduce two external memories to keep the history information, which can relieve the pressure on low-capacity internal memory.

Following (Tran et al., 2016), we use external memory constructed by history hidden states, which is defined as

$$\mathbf{M}_t = \{\mathbf{h}_{t-K}, \ldots, \mathbf{h}_{t-1}\} \in \mathbb{R}^{K \times d}, \quad (9)$$

where K is the number of memory segments, which is generally instance-independent and predefined as hyper-parameter; d is the size of each segment; and $\mathbf{h}_t$ is the hidden state at time t emitted by LSTM.

At position i, j, two memory blocks $\mathbf{M}^{(x)}, \mathbf{M}^{(y)}$ are used to store contextual information of x and y respectively.

$$\mathbf{M}_{i,j}^{(x)} = \{\mathbf{h}_{i-K,j}^{(x)}, \ldots, \mathbf{h}_{i-1,j}^{(x)}\}, \quad (10)$$

$$\mathbf{M}_{i,j}^{(y)} = \{\mathbf{h}_{i,j-K}^{(y)}, \ldots, \mathbf{h}_{i,j-1}^{(y)}\}, \quad (11)$$

where $\mathbf{h}^{(x)}$ and $\mathbf{h}^{(x)}$ are outputs of two conditional LSTMs at different positions.

The history information can be read from these two memory blocks. We denote a read vector from external memories as $\mathbf{r}_{i,j} \in \mathbb{R}^d$, which can be computed by soft attention mechanisms.

$$\mathbf{r}_{i,j}^{(x)} = \mathbf{a}_{i,j}^{(x)} \mathbf{M}_{i,j}^{(x)}, \quad (12)$$

$$\mathbf{r}_{i,j}^{(y)} = \mathbf{a}_{i,j}^{(y)} \mathbf{M}_{i,j}^{(y)}, \quad (13)$$

where $\mathbf{a}_{i,j} \in \mathbb{R}^K$ represents attention distribution over the corresponding memory $\mathbf{M}_{i,j} \in \mathbb{R}^{K \times d}$.

More concretely, each scalar $a_{i,j,k}$ in attention distribution $\mathbf{a}_{i,j}$ can be obtained:

$$a_{i,j,k}^{(x)} = \text{softmax}(g(\mathbf{M}_{i,j,k}^{(x)}, \mathbf{r}_{i-1,j}^{(x)}, \mathbf{x}_i)), \quad (14)$$

$$a_{i,j,k}^{(y)} = \text{softmax}(g(\mathbf{M}_{i,j,k}^{(y)}, \mathbf{r}_{i,j-1}^{(y)}, \mathbf{y}_j)), \quad (15)$$

where $\mathbf{M}_{i,j,k} \in \mathbb{R}^d$ represents the k-th row memory vector at position (i, j), and $g(\cdot)$ is an align function defined by

$$g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{v}^T \tanh(\mathbf{W}_a[\mathbf{x}; \mathbf{y}, \mathbf{z}]), \quad (16)$$

where $\mathbf{v} \in \mathbb{R}^d$ is a parameter vector and $\mathbf{W}_a \in \mathbb{R}^{d \times 3d}$ is a parameter matrix.

The history information $\mathcal{H}_{i,j}$ in Eq (7) and (8) is computed by

$$\mathcal{H}_{i,j} = \mathbf{r}_{i,j}^{(x)} \oplus \mathbf{r}_{i,j}^{(y)}. \quad (17)$$

By incorporating external memory blocks, DF-LSTMs allow network to re-read history interaction information, therefore it can more easily capture complicated and long-distance matching patterns. As shown in Figure 3, the forward pass of DF-LSTMs can be unfolded along two dimensional ordering.

4.1 Related Models

Our model is inspired by some recently proposed models based on recurrent neural network (RNN).

One kind of models is multi-dimensional recurrent neural network (MD-RNN) (Graves et al., 2007; Graves and Schmidhuber, 2009; Byeon et al., 2015) in machine learning and computer vision communities. As mentioned above, if we just use the neighbor states, our model can be regarded as grid LSTMs (Kalchbrenner et al., 2015).

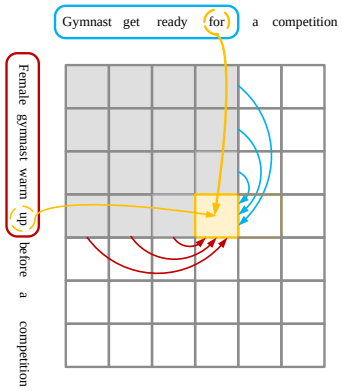


Figure 3: Illustration of unfolded DF-LSTMs.

What is different is the dependency relations between the current state and history states. Our model uses external memory to increase its memory capacity and therefore can store large useful interactions of subsequences. Thus, we can discover some matching patterns with long dependence.

Another kind of models is memory augmented RNN, such as long short-term memory-network (Cheng et al., 2016) and recurrent memory network (Tran et al., 2016), which extend memory network (Bahdanau et al., 2014) and equip the RNN with ability of re-reading the history information. While they focus on sequence modelling, our model concentrates more on modelling the interactions of sequence pair.

5 Training

5.1 Task Specific Output

There are two popular types of text matching tasks in NLP. One is ranking task, such as community question answering. Another is classification task, such as textual entailment.

We use different ways to calculate matching score for these two types of tasks.

1. For ranking task, the output is a scalar matching score, which is obtained by a linear transformation of the matching vector obtained by FD-LSTMs.

2. For classification task, the outputs are the probabilities of the different classes, which

is computed by a softmax function on the matching vector obtained by FD-LSTMs.

5.2 Loss Function

Accordingly, we use two loss functions to deal with different sentence matching tasks.

Max-Margin Loss for Ranking Task Given a positive sentence pair (X, Y) and its corresponding negative pair $(X, \hat{Y})$. The matching score $s(X, Y)$ should be larger than $s(X, \hat{Y})$.

For this task, we use the contrastive max-margin criterion (Bordes et al., 2013; Socher et al., 2013) to train our model on matching task.

The ranking-based loss is defined as

$$L(X, Y, \hat{Y}) = max(0, 1 - s(X, Y) + s(X, \hat{Y})). \quad (18)$$

where $s(X, Y)$ is predicted matching score for (X, Y).

Cross-entropy Loss for Classification Task Given a sentence pair (X, Y) and its label l. The output $\hat{l}$ of neural network is the probabilities of the different classes. The parameters of the network are trained to minimise the cross-entropy of the predicted and true label distributions.

$$L(X, Y; \boldsymbol{l}, \hat{\boldsymbol{l}}) = -\sum_{j=1}^{C} \boldsymbol{l}_j \log(\hat{\boldsymbol{l}}_j), \quad (19)$$

where $\boldsymbol{l}$ is one-hot representation of the ground-truth label l; $\hat{\boldsymbol{l}}$ is predicted probabilities of labels; C is the class number.

5.3 Optimizer

To minimize the objective, we use stochastic gradient descent with the diagonal variant of AdaGrad (Duchi et al., 2011).

To prevent exploding gradients, we perform gradient clipping by scaling the gradient when the norm exceeds a threshold (Graves, 2013).

5.4 Initialization and Hyperparameters

Orthogonal Initialization We use orthogonal initialization of our LSTMs, which allows neurons to react to the diverse patterns and is helpful to train a multi-layer network (Saxe et al., 2013).

Unsupervised Initialization The word embeddings for all of the models are initialized with the 100d GloVe vectors (840B token version, (Pennington et al., 2014)). The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$.

Hyper-parameters	MQA	RTE
K	9	9
Embedding size	100	100
Hidden layer size	50	100
Initial learning rate	0.05	0.005
Regularization	$5E{-}5$	$1E{-}5$

Table 1: Hyper-parameters for our model on two tasks.

Hyperparameters For each task, we used a stacked DF-LSTM and take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.05, 0.0005, 0.0001]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5, 1E{-}6]$ and the values of K $[1, 3, 6, 9, 12]$. The final hyper-parameters are set as Table 1.

6 Experiment

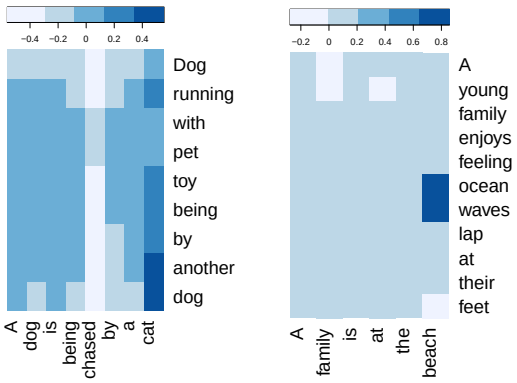
In this section, we investigate the empirical performances of our proposed model on two different text matching tasks: classification task (recognizing textual entailment) and ranking task (matching of question and answer).

6.1 Competitor Methods

- Neural bag-of-words (NBOW): Each sequence is represented as the sum of the embeddings of the words it contains, then they are concatenated and fed to a MLP.
- Single LSTM: Two sequences are encoded by a single LSTM, proposed by (Rocktäschel et al., 2015).
- Parallel LSTMs: Two sequences are first encoded by two LSTMs separately, then they are concatenated and fed to a MLP.
- Attention LSTMs: Two sequences are encoded by LSTMs with attention mechanism, proposed by (Rocktäschel et al., 2015).
- Word-by-word Attention LSTMs: An improved strategy of attention LSTMs, which introduces word-by-word attention mechanism and is proposed by (Rocktäschel et al., 2015).

Model	k	Train	Test
NBOW	100	77.9	75.1
single LSTM (Rocktäschel et al., 2015)	100	83.7	80.9
parallel LSTMs (Bowman et al., 2015)	100	84.8	77.6
Attention LSTM (Rocktäschel et al., 2015)	100	83.2	82.3
Attention(w-by-w) LSTM (Rocktäschel et al., 2015)	100	83.7	83.5
DF-LSTMs	100	85.2	**84.6**

Table 2: Accuracies of our proposed model against other neural models on SNLI corpus.



(a) 5-th neuron (b) 11-th neuron

Figure 4: Illustration of two interpretable neurons and some word-pairs captured by these neurons. The darker patches denote the corresponding activations are higher.

6.2 Experiment-I: Recognizing Textual Entailment

Recognizing textual entailment (RTE) is a task to determine the semantic relationship between two sentences. We use the Stanford Natural Language Inference Corpus (SNLI) (Bowman et al., 2015). This corpus contains 570K sentence pairs, and all of the sentences and labels stem from human annotators. SNLI is two orders of magnitude larger than all other existing RTE corpora. Therefore, the massive scale of SNLI allows us to train powerful neural networks such as our proposed architecture in this paper.

6.2.1 Results

Table 2 shows the evaluation results on SNLI. The 2nd column of the table gives the number of hidden states.

From experimental results, we have several experimental findings.

The results of DF-LSTMs outperform all the competitor models with the same number of hidden states while achieving comparable results to the state-of-the-art and using much fewer parameters, which indicate that it is effective to model the strong interactions of two texts in a recursive matching way.

All models outperform NBOW by a large margin, which indicate the importance of words order in semantic matching.

The strong interaction models surpass the weak interaction models, for example, compared with parallel LSTMs, DF-LSTMs obtain improvement by 7.0%.

6.2.2 Understanding Behaviors of Neurons in DF-LSTMs

To get an intuitive understanding of how the DF-LSTMs work on this problem, we examined the neuron activations in the last aggregation layer while evaluating the test set. We find that some cells are bound to certain roles.

We refer to $h_{i,j,k}$ as the activation of the k-th neuron at the position of (i, j), where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. By visualizing the hidden state $\mathbf{h}_{i,j,k}$ and analyzing the maximum activation, we can find that there exist multiple interpretable neurons. For example, when some contextualized local perspectives are semantically related at point (i, j) of the sentence pair, the activation value of hidden neuron $h_{i,j,k}$ tends to be maximum, meaning that the model could capture some reasoning patterns.

Figure 4 illustrates this phenomenon. In Figure 4(a), a neuron shows its ability to monitor the word pairs with the property of describing different things of the same type.

The activation in the patch, containing the word pair "`(cat, dog)`", is much higher than others. This is an informative pattern for the relation prediction of these two sentences, whose ground truth is contradiction. An interesting thing is there are two "`dog`" in sentence "`Dog running with pet toy being by another dog`". Our model ignores the useless word, which indicates this neuron selectively captures pattern by contextual understanding, not just word level interaction.

In Figure 4(b), another neuron shows that it can capture the local contextual interactions, such as "`(ocean waves, beach)`". These patterns can be easily captured by final layer and provide a strong support for the final prediction.

Index of Cell	Word or Phrase Pairs	Explanation
5-th	(jeans, shirt), (dog, cat) (retriever, cat), (stand, sitting)	different entities or events of the same type
11-th	(pool, swimming), (street, outside) (animal, dog), (grass,outside)	word pair related to lexical entailment
20-th	(skateboard, skateboarding), (running, runs) (advertisement, ad), (grassy, grass)	words with different morphology
49-th	(blue, blue), (wearing black, wearing white), (green uniform, red uniform)	words related to color
55-th	(a man, two other men), (a man, two girls) (Two women, No one)	subjects with singular or plural forms

Table 3: Multiple interpretable neurons and the word-pairs/phrase-pairs captured by these neurons. The third column gives the explanations of corresponding neuron's behaviours.

Table 3 illustrates multiple interpretable neurons and some representative word or phrase pairs which can activate these neurons. These cases show that our model can capture contextual interactions beyond word level.

6.2.3 Case Study for Attention Addressing Mechanism

External memory with attention addressing mechanism enables the network explicitly to utilize the history information of two sentences simultaneously. As a by-product, the obtained attention distribution over history hidden states also help us interpret the network and discover underlying dependencies present in the data.

To this end, we randomly sample two good cases with entailment relation from test data and visualize attention distributions over external memory constructed by last 9 hidden states. As shown in Figure 5(a), For the first sentence pair, when the word pair "`(competition, competition)`" are processed, the model simultaneously selects "`warm, before`" from one sentence and "`gymnast,ready,for`" from the other, which are informative patterns and indicate our model has the capacity of capturing phrase-phrase pair.

Another case in Figure 5(b) also shows by attention mechanism, the network can sufficiently utilize the history information and the fusion approach allows two LSTMs to share the history information of each other.

6.2.4 Error Analysis

Although our model DF-LSTMs are more sensitive to the discrepancy of the semantic capacity between two sentences, some cases still can not be solved by our model. For example, our model gives a wrong prediction of the sentence pair "`A golden retriever nurses puppies/Puppies next to their mother`", whose ground truth is entailment. The model fails to realize "`nurses`" means "`next to`".

Besides, despite the large size of the training corpus, it's still very difficult to solve some cases, which depend on the combination of the world knowledge and context-sensitive inferences. For example, given an entailment pair "`Several women are playing volleyball/The women are hitting a ball with their arms`", all models predict "`neutral`".

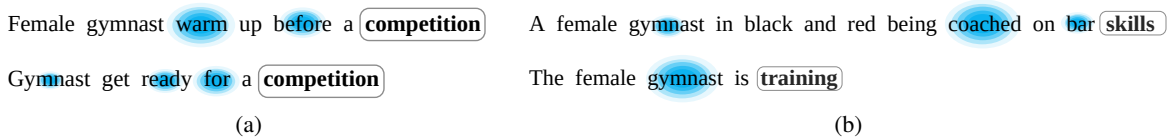
These analysis suggests that some architectural improvements or external world knowledge are necessary to eliminate all errors instead of simply scaling up the basic model.

6.3 Experiment-II: Matching Question and Answer

Matching question answering (MQA) is a typical task for semantic matching (Zhou et al., 2013). Given a question, we need select a correct answer from some candidate answers.

In this paper, we use the dataset collected from Yahoo! Answers with the getByCategory function provided in Yahoo! Answers API, which produces $963,072$ questions and corresponding best answers. We then select the pairs in which the length of questions and answers are both in the interval $[4, 30]$, thus obtaining $220,000$ question answer pairs to form the positive pairs.

For negative pairs, we first use each question's best answer as a query to retrieval top $1,000$ re-



(a) (b)

Figure 5: Examples of external memory positions attended when encoding the next word pair (bold and marked by a box)

Model	k	P@1(5)	P@1(10)
Random Guess	-	20.0	10.0
NBOW	50	63.9	47.6
single LSTM	50	68.2	53.9
parallel LSTMs	50	66.9	52.1
Attention LSTMs	50	73.5	62.0
Attention(w-by-w) LSTMs	50	75.1	64.0
DF-LSTMs	50	**76.5**	**65.0**

Table 4: Results of our proposed model against other neural models on Yahoo! question-answer pairs dataset.

sults from the whole answer set with Lucene, where 4 or 9 answers will be selected randomly to construct the negative pairs.

The whole dataset[1] is divided into training, validation and testing data with proportion $20:1:1$. Moreover, we give two test settings: selecting the best answer from 5 and 10 candidates respectively.

6.3.1 Results

Results of MQA are shown in the Table 4. we can see that the proposed model also shows its superiority on this task, which outperforms the state-of-the-arts methods on both metrics (P@1(5) and P@1(10)) with a large margin.

By analyzing the evaluation results of question-answer matching in Table 4, we can see strong interaction models (attention LSTMs, our DF-LSTMs) consistently outperform the weak interaction models (NBOW, parallel LSTMs) with a large margin, which suggests the importance of modelling strong interaction of two sentences.

7 Related Work

Our model can be regarded as a strong interaction model, which has been explored in previous methods.

One kind of methods is to compute similarities between all the words or phrases of the two sentences to model multiple-granularity interactions of two sentences, such as RAE (Socher et al., 2011), Arc-II (Hu et al., 2014),ABCNN (Yin et al., 2015),MultiGranCNN (Yin and Schütze, 2015), Multi-Perspective CNN (He et al., 2015), MV-LSTM (Wan et al., 2016).

Socher et al. (2011) firstly used this paradigm for paraphrase detection. The representations of words or phrases are learned based on recursive autoencoders.

Hu et al. (2014) proposed to an end-to-end architecture with convolutional neural network (Arc-II) to model multiple-granularity interactions of two sentences.

Wan et al. (2016) used LSTM to enhance the positional contextual interactions of the words or phrases between two sentences. The input of LSTM for one sentence does not involve another sentence.

Another kind of methods is to model the conditional encoding, in which the encoding of one sentence can be affected by another sentence. Rocktäschel et al. (2015) and Wang and Jiang (2015) used LSTM to read pairs of sequences to produce a final representation, which can be regarded as interaction of two sequences. By incorporating an attention mechanism, they got further improvements to the predictive abilities.

Different with these two kinds of methods, we model the interactions of two texts in a recursively matching way. Based on this idea, we propose a model of deep fusion LSTMs to accomplish recursive conditional encodings.

8 Conclusion and Future Work

In this paper, we propose a model of deep fusion LSTMs to capture the strong interaction for text semantic matching. Experiments on two large scale text matching tasks demonstrate the efficacy of our proposed model and its superiority to competitor models. Besides, our visualization analysis revealed that multiple interpretable neurons in our model can capture the contextual interactions of the words or phrases.

[1] http://nlp.fudan.edu.cn/data/.

In future work, we would like to investigate our model on more text matching tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. 2015. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555.

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

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.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. In *Artificial Neural Networks–ICANN 2007*, pages 549–558. Springer.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

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*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of International Joint Conference on Artificial Intelligence*.

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

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

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*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. *CoRR*, abs/1601.01272.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*.

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

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.

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

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *IJCAI*.