A Sentence Interaction Network for Modeling Dependence between Sentences

Biao Liu[1], **Minlie Huang**[1]*, **Song Liu**[2], **Xuan Zhu**[2], **Xiaoyan Zhu**[1]
[1]State Key Lab. of Intelligent Technology and Systems
[1]National Lab. for Information Science and Technology
[1]Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Samsung R&D Institute, Beijing, China
[1]liubiao2638@gmail.com, {aihuang, zxy-dcs}@tsinghua.edu.cn

Abstract

Modeling interactions between two sentences is crucial for a number of natural language processing tasks including Answer Selection, Dialogue Act Analysis, etc. While deep learning methods like Recurrent Neural Network or Convolutional Neural Network have been proved to be powerful for sentence modeling, prior studies paid less attention on interactions between sentences. In this work, we propose a Sentence Interaction Network (SIN) for modeling the complex interactions between two sentences. By introducing "interaction states" for word and phrase pairs, SIN is powerful and flexible in capturing sentence interactions for different tasks. We obtain significant improvements on Answer Selection and Dialogue Act Analysis without any feature engineering.

1 Introduction

There exist complex interactions between sentences in many natural language processing (NLP) tasks such as Answer Selection (Yu et al., 2014; Yin et al., 2015), Dialogue Act Analysis (Kalchbrenner and Blunsom, 2013), etc. For instance, given a question and two candidate answers below, though they are all talking about cats, only the first answer correctly answers the question about cats' appearance. It is important to appropriately model the relation between two sentences in such cases.

Q	*What do cats look like?*
A1	*Cats have large eyes and furry bodies.*
A2	*Cats like to play with boxes and bags.*

* Correspondence author

For sentence pair modeling, some methods first project the two sentences to fix-sized vectors separately without considering the interactions between them, and then fed the sentence vectors to other classifiers as features for a specific task (Kalchbrenner and Blunsom, 2013; Tai et al., 2015). Such methods suffer from being unable to encode context information during sentence embedding.

A more reasonable way to capture sentence interactions is to introduce some mechanisms to utilize information from both sentences at the same time. Some methods attempt to introduce an attention matrix which contains similarity scores between words and phrases to approach sentence interactions (Socher et al., 2011; Yin et al., 2015). While the meaning of words and phrases may drift from contexts to contexts, simple similarity scores may be too weak to capture the complex interactions, and a more powerful interaction mechanism is needed.

In this work, we propose a Sentence Interaction Network (SIN) focusing on modeling sentence interactions. The main idea behind this model is that each word in one sentence may potentially influence every word in another sentence in some degree (the word "influence" here may refer to "answer" or "match" in different tasks). So, we introduce a mechanism that allows information to flow from every word (or phrase) in one sentence to every word (or phrase) in another sentence. These "information flows" are real-valued vectors describing how words and phrases interact with each other, for example, a word (or phrase) in one sentence can modify the meaning of a word (or phrase) in another sentence through such "information flows".

Specifically, given two sentences s_1 and s_2, for every word x_t in s_1, we introduce a "candidate interaction state" for every word x_τ in s_2. This

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 558–567,
Berlin, Germany, August 7-12, 2016.

state is regarded as the "influence" of x_τ to x_t, and is actually the "information flow" from x_τ to x_t mentioned above. By summing over all the "candidate interaction states", we generate an "interaction state" for x_t, which represents the influence of the whole sentence s_2 to word x_t . When feeding the "interaction state" and the word embedding together into Recurrent Neural Network (with Long Short-Time Memory unit in our model), we obtain a sentence vector with context information encoded. We also add a convolution layer on the word embeddings so that interactions between phrases can also be modeled.

SIN is powerful and flexible for modeling sentence interactions in different tasks. First, the "interaction state" is a vector, compared with a single similarity score, it is able to encode more information for word or phrase interactions. Second, the interaction mechanism in SIN can be adapted to different functions for different tasks during training, such as "word meaning adjustment" for Dialogue Act Analysis or "Answering" for Answer Selection.

Our main contributions are as follows:

- We propose a Sentence Interaction Network (SIN) which utilizes a new mechanism to model sentence interactions.
- We add convolution layers to SIN, which improves the ability to model interactions between phrases.
- We obtain significant improvements on Answer Selection and Dialogue Act Analysis without any handcrafted features.

The rest of the paper is structured as follows: We survey related work in Section 2, introduce our method in Section 3, present the experiments in Section 4, and summarize our work in Section 5.

2 Related Work

Our work is mainly related to deep learning for sentence modeling and sentence pair modeling.

For sentence modeling, we have to first represent each word as a real-valued vector (Mikolov et al., 2010; Pennington et al., 2014) , and then compose word vectors into a sentence vector. Several methods have been proposed for sentence modeling. Recurrent Neural Network (RNN) (Elman, 1990; Mikolov et al., 2010) introduces a hidden state to represent contexts, and repeatedly feed the hidden state and word embeddings to the network to update the context representation. RNN suffers from gradient vanishing and exploding problems which limit the length of reachable context. RNN with Long Short-Time Memory Network unit (LSTM) (Hochreiter and Schmidhuber, 1997; Gers, 2001) solves such problems by introducing a "memory cell" and "gates" into the network. Recursive Neural Network (Socher et al., 2013; Qian et al., 2015) and LSTM over tree structures (Zhu et al., 2015; Tai et al., 2015) are able to utilize some syntactic information for sentence modeling. Kim (2014) proposed a Convolutional Neural Network (CNN) for sentence classification which models a sentence in multiple granularities.

For sentence pair modeling, a simple idea is to first project the sentences to two sentence vectors separately with sentence modeling methods, and then feed these two vectors into other classifiers for classification (Tai et al., 2015; Yu et al., 2014; Yang et al., 2015). The drawback of such methods is that separately modeling the two sentences is unable to capture the complex sentence interactions. Socher et al. (2011) model the two sentences with Recursive Neural Networks (Unfolding Recursive Autoencoders), and then feed similarity scores between words and phrases (syntax tree nodes) to a CNN with dynamic pooling to capture sentence interactions. Hu et al. (2014) first create an "interaction space" (matching score matrix) by feeding word and phrase pairs into a multilayer perceptron (MLP), and then apply CNN to such a space for interaction modeling. Yin et al. (2015) proposed an Attention based Convolutional Neural Network (ABCNN) for sentence pair modeling. ABCNN introduces an attention matrix between the convolution layers of the two sentences, and feed the matrix back to CNN to model sentence interactions. There are also some methods that make use of rich lexical semantic features for sentence pair modeling (Yih et al., 2013; Yang et al., 2015), but these methods can not be easily adapted to different tasks.

Our work is also related to context modeling. Hermann et al. (2015) proposed a LSTM-based method for reading comprehension. Their model is able to effectively utilize the context (given by a document) to answer questions. Ghosh et al. (2016) proposed a Contextual LSTM (CLSTM) which introduces a topic vector into LSTM for context modeling. The topic vector in CLSTM is

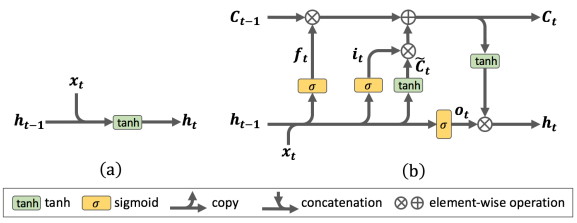

Figure 1: RNN (a) and LSTM (b) [1]

computed according to those already seen words, and therefore reflects the underlying topic of the current word.

3 Method

3.1 Background: RNN and LSTM

Recurrent Neural Network (RNN) (Elman, 1990; Mikolov et al., 2010), as depicted in Figure 1(a), is proposed for modeling long-distance dependence in a sequence. Its hidden layer is connected to itself so that previous information is considered in later times. RNN can be formalized as

$$h_t = f(W_x x_t + W_h h_{t-1} + b_h)$$

where x_t is the input at time step t and h_t is the hidden state. Though theoretically, RNN is able to capture dependence of arbitrary length, it tends to suffer from the gradient vanishing and exploding problems which limit the length of reachable context. In addition, an additive function of the previous hidden layer and the current input is too simple to describe the complex interactions within a sequence.

RNN with Long Short-Time Memory Network unit (LSTM, Figure 1(b)) (Hochreiter and Schmidhuber, 1997; Gers, 2001) solves such problems by introducing a "memory cell" and "gates" into the network. Each time step is associated with a subnet known as a memory block in which a "memory cell" stores the context information and "gates" control which information should be added or discarded or reserved. LSTM can be formalized as

$$\begin{aligned} f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\ i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [x_t, h_{t-1}] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned}$$

where $*$ means element-wise multiplication, f_t, i_t, o_t is the forget, input and output gate that control which information should be forgot, input and output, respectively. $\tilde{C}_t$ is the candidate information to be added to the memory cell state C_t. h_t is the hidden state which is regarded as a representation of the current time step with contexts.

In this work, we use LSTM with peephole connections, namely adding C_{t-1} to compute the forget gate f_t and the input gate i_t, and adding C_t to compute the output gate o_t.

3.2 Sentence Interaction Network (SIN)

Sentence Interaction Network (SIN, Figure 2) models the interactions between two sentences in two steps.

First, we use a LSTM (referred to as LSTM_1) to model the two sentences s_1 and s_2 separately, and the hidden states related to the t-th word in s_1 and the τ-th word in s_2 are denoted as $z_t^{(1)}$ and $z_\tau^{(2)}$ respectively. For simplicity, we will use the position (t, τ) to denote the corresponding words hereafter.

Second, we propose a new mechanism to model the interactions between s_1 and s_2 by allowing information to flow between them. Specifically, word t in s_1 may be potentially influenced by all words in s_2 in some degree. Thus, for word t in s_1, a candidate interaction state $\tilde{c}_{t\tau}^{(i)}$ and an input gate $i_{t\tau}^{(i)}$ are introduced for each word τ in s_2 as follows:

$$\begin{aligned} \tilde{c}_{t\tau}^{(i)} &= \tanh(W_c^{(i)} \cdot [z_t^{(1)}, z_\tau^{(2)}] + b_c^{(i)}) \\ i_{t\tau}^{(i)} &= \sigma(W_i^{(i)} \cdot [z_t^{(1)}, z_\tau^{(2)}] + b_i^{(i)}) \end{aligned}$$

here, the superscript "i" indicates "interaction". $W_c^{(i)}, W_i^{(i)}, b_c^{(i)}, b_i^{(i)}$ are model parameters. The interaction state $c_t^{(i)}$ for word t in s_1 can then be formalized as

$$c_t^{(i)} = \sum_{\tau=1}^{|s_2|} \tilde{c}_{t\tau}^{(i)} * i_{t\tau}^{(i)}$$

where $|s_2|$ is the length of sentence s_2, and $c_t^{(i)}$ can be viewed as the total interaction information received by word t in s_1 from sentence s_2. The interaction states of words in s_2 can be similarly

[1] This figure referred to http://colah.github.io/posts/2015-08-Understanding-LSTMs/

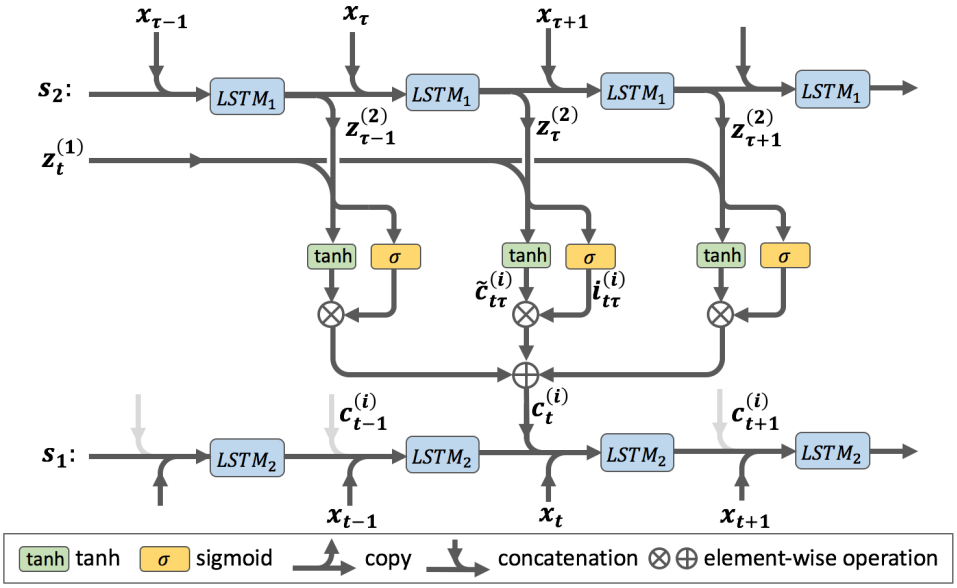

Figure 2: SIN for modeling sentence s_1 at timestep t. First, we model s_1 and s_2 separately with LSTM$_1$ and obtain the hidden states $z_t^{(1)}$ for s_1 and $z_\tau^{(2)}$ for s_2. Second, we compute interaction states based on these hidden states, and incorporate $c_t^{(i)}$ into LSTM$_2$. Information flows (interaction states) from s_1 to s_2 are not depicted here for simplicity.

computed by exchanging the position of $z_t^{(1)}$ and $z_\tau^{(2)}$ in $\tilde{c}_{t\tau}^{(i)}$ and $i_{t\tau}^{(i)}$ while sharing the model parameters.

We now introduce the interaction states into another LSTM (referred to as LSTM$_2$) to compute the sentence vectors. Therefore, information can flow between the two sentences through these states. For sentence s_1, at timestep t, we have

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [x_t, h_{t-1}, c_t^{(i)}, C_{t-1}] + b_f) \\
i_t &= \sigma(W_i \cdot [x_t, h_{t-1}, c_t^{(i)}, C_{t-1}] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [x_t, h_{t-1}, c_t^{(i)}] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [x_t, h_{t-1}, c_t^{(i)}, C_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

By averaging all hidden states of LSTM$_2$, we obtain the sentence vector v_{s_1} of s_1, and the sentence vector v_{s_2} of s_2 can be computed similarly. v_{s_1} and v_{s_2} can then be used as features for different tasks.

In SIN, the candidate interaction state $\tilde{c}_{t\tau}^{(i)}$ represents the potential influence of word τ in s_2 to word t in s_1, and the related input gate $i_{t\tau}^{(i)}$ controls the degree of the influence. The element-wise multiplication $\tilde{c}_{t\tau}^{(i)} * i_{t\tau}^{(i)}$ is then the actual influence. By summing over all words in s_2, the interaction state $c_t^{(i)}$ gives the influence of the whole sentence s_2 to word t.

3.3 SIN with Convolution (SIN-CONV)

SIN is good at capturing the complex interactions of words in two sentences, but not strong enough for phrase interactions. Since convolutional neural network is widely and successfully used for modeling phrases, we add a convolution layer before SIN to model phrase interactions between two sentences.

Let $v_1, v_2, ..., v_{|s|}$ be the word embeddings of a sentence s, and let $c_i \in \mathbb{R}^{wd}, 1 \leq i \leq |s| - w + 1$, be the concatenation of $v_{i:i+w-1}$, where w is the window size. The representation p_i for phrase $v_{i:i+w-1}$ is computed as:

$$p_i = \tanh(F \cdot c_i + b)$$

where $F \in \mathbb{R}^{d \times wd}$ is the convolution filter, and d is the dimension of the word embeddings.

In SIN-CONV, we first use a convolution layer to obtain phrase representations for the two sentences s_1 and s_2, and the SIN interaction procedure is then applied to these phrase representations as before to model phrase interactions. The average of all hidden states are treated as sentence vectors $v_{s_1}^{cnn}$ and $v_{s_2}^{cnn}$. Thus, SIN-CONV is SIN with word vectors substituted by phrase vectors. The

two phrase-based sentence vectors are then fed to a classifier along with the two word-based sentence vectors together for classification.

The LSTM and interaction parameters are not shared between SIN and SIN-CONV.

4 Experiments

In this section, we test our model on two tasks: Answer Selection and Dialogue Act Analysis. Both tasks require to model interactions between sentences. We also conduct auxiliary experiments for analyzing the interaction mechanism in our SIN model.

4.1 Answer Selection

Selecting correct answers from a set of candidates for a given question is quite crucial for a number of NLP tasks including question-answering, natural language generation, information retrieval, etc. The key challenge for answer selection is to appropriately model the complex interactions between the question and the answer, and hence our SIN model is suitable for this task.

We treat Answer Selection as a classification task, namely to classify each question-answer pair as "correct" or "incorrect". Given a question-answer pair (q, a), after generating the question and answer vectors v_q and v_a using SIN, we feed them to a logistic regression layer to output a probability. And we maximize the following objective function:

$$p_\theta(q, a) = \sigma(W \cdot [v_q, v_a]) + b)$$
$$\mathcal{L} = \sum_{(q,a)} \hat{y}_{q,a} \log p_\theta(q, a) + (1 - \hat{y}_{q,a}) \log(1 - p_\theta(q, a))$$

where $\hat{y}_{q,a}$ is the true label for the question-answer pair (q, a) (1 for correct, 0 for incorrect). For SIN-CONV, the sentence vector v_q^{cnn} and v_a^{cnn} are also fed to the logistic regression layer.

During evaluation, we rank the answers of a question q according to the probability $p_\theta(q, a)$. The evaluation metrics are mean average precision (MAP) and mean reciprocal rank (MRR).

4.1.1 Dataset

The WikiQA[2](Yang et al., 2015) dataset is used for this task. Following Yin et al. (2015), we filtered out those questions that do not have any correct answers from the development and test set. Some statistics are shown in Table 1.

	Q	QA pair	A/Q	correct A/Q
Train	2,118	20,360	9.61	0.49
Dev	126	1,130	8.97	1.11
Test	243	2,351	9.67	1.21

Table 1: Statistics of WikiQA (Q=Question, A=Answer)

4.1.2 Setup

We use the 100-dimensional GloVe vectors[3] (Pennington et al., 2014) to initialize our word embeddings, and those words that do not appear in Glove vectors are treated as unknown. The dimension of all hidden states is set to 100 as well. The window size of the convolution layer is 2. To avoid overfitting, dropout is introduced to the sentence vectors, namely setting some dimensions of the sentence vectors to 0 with a probability p (0.5 in our experiment) randomly. No handcrafted features are used in our methods and the baselines.

Mini-batch Gradient Descent (30 question-answer pairs for each mini batch), with AdaDelta tuning learning rate, is used for model training. We update model parameters after every mini batch, check validation MAP and save model after every 10 batches. We run 10 epochs in total, and the model with highest validation MAP is treated as the optimal model, and we report the corresponding test MAP and MRR metrics.

4.1.3 Baselines

We compare our SIN and SIN-CONV model with 5 baselines listed below:

- LCLR: The model utilizes rich semantic and lexical features (Yih et al., 2013).
- PV: The cosine similarity score of paragraph vectors of the two sentences is used to rank answers (Le and Mikolov, 2014).
- CNN: Bigram CNN (Yu et al., 2014).
- ABCNN: Attention based CNN, no handcrafted features are used here (Yin et al., 2015).
- LSTM: The question and answer are modeled by a simple LSTM. Different from SIN, there is no interaction between sentences.

[2]http://aka.ms/WikiQA

[3]http://nlp.stanford.edu/projects/glove/

4.1.4 Results

Results are shown in Table 2. SIN performs much better than LSTM, PV and CNN, this justifies that the proposed interaction mechanism well captures the complex interactions between the question and the answer. But SIN performs slightly worse than ABCNN because it is not strong enough at modeling phrases. By introducing a simple convolution layer to improve its phrase-modeling ability, SIN-CONV outperforms all the other models.

For SIN-CONV, we do not observe much improvements by using larger convolution filters (window size ≥ 3) or stacking more convolution layers. The reason may be the fact that interactions between long phrases is relatively rare, and in addition, the QA pairs in the WikiQA dataset may be insufficient for training such a complex model with long convolution windows.

4.2 Dialogue Act Analysis

Dialogue acts (DA), such as *Statement, Yes-No-Question, Agreement*, indicate the sentence pragmatic role as well as the intention of the speakers (Williams, 2012). They are widely used in natural language generation (Wen et al., 2015), speech and meeting summarization (Murray et al., 2006; Murray et al., 2010), etc. In a dialogue, the DA of a sentence is highly relevant to the content of itself and the previous sentences. As a result, to model the interactions and long-range dependence between sentences in a dialogue is crucial for dialogue act analysis.

Given a dialogue (n sentences) $d = [s_1, s_2, ..., s_n]$, we first use a LSTM (LSTM_1) to model all the sentences independently. The hidden states of sentence s_i obtained at this step are used to compute the interaction states of sentence s_{i+1}, and SIN will generate a sentence vector v_{s_i} using another LSTM (LSTM_2) for each sentence s_i in the dialogue (see Section 3.2) . These sentence vectors can be used as features for dialogue act analysis. We refer to this method as SIN (or SIN-CONV for adding a convolution layer).

For dialogue act analysis, we add a softmax layer on the sentence vector v_{s_i} to predict the probability distribution:

$$p_\theta(y_j|v_{s_i}) = \frac{\exp(v_{s_i}^T \cdot w_j + b_j)}{\sum_k \exp(v_{s_i}^T \cdot w_k + b_k)}$$

[4]With extra handcrafted features, ABCNN's performance is: MAP(0.692), MRR(0.711).

Model	MAP	MRR
LCLR	0.599	0.609
PV	0.511	0.516
CNN	0.619	0.628
ABCNN	0.660	0.677
LSTM	0.634	0.648
SIN	0.657	0.672
SIN-CONV	**0.674**	**0.693**

Table 2: Results on answer selection[4].

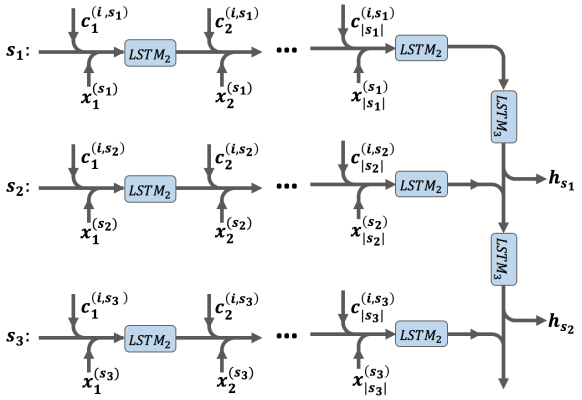

Figure 3: SIN-LD for dialogue act analysis. LSTM_1 is not shown here for simplicity. $x_t^{(s_j)}$ means word t in s_j, $c_t^{(i,s_j)}$ means the interaction state for word t in s_j.

where y_j is the j-th DA tag, w_j and b_j is the weight vector and bias corresponding to y_j. We maximize the following objective function:

$$\mathcal{L} = \sum_{d \in \mathcal{D}} \sum_{i=1}^{|d|} \log p_\theta(\hat{y}_{s_i}|v_{s_i})$$

where $\mathcal{D}$ is the training set, namely a set of dialogues, $|d|$ is the length of the dialogue, s_i is the i-th sentence in d, $\hat{y}_{s_i}$ is the true dialogue act label of s_i.

In order to capture long-range dependence in the dialogue, we can further join up the sentence vector v_{s_i} with another LSTM (LSTM_3). The hidden state h_{s_i} of LSTM_3 are treated as the final sentence vector, and the probability distribution is given by substituting v_{s_i} with h_{s_i} in $p_\theta(y_j|v_{s_i})$. We refer to this method as SIN-LD (or SIN-CONV-LD for adding a convolution layer), where LD means long-range dependence. Figure 3 shows the whole structure (LSTM_1 is not shown here for simplicity).

Dialogue Act	Example	Train(%)	Test(%)
Statement-non-Opinion	*Me, I'm in the legal department.*	37.0	31.5
Backchannel/Acknowledge	*Uh-huh.*	18.8	18.3
Statement-Opinion	*I think it's great*	12.8	17.2
Abandoned/Uninterpretable	*So,-*	7.6	8.6
Agreement/Accept	*That's exactly it.*	5.5	5.0
Appreciation	*I can imagine.*	2.4	1.8
Yes-No-Question	*Do you have to have any special training?*	2.3	2.0
Non-Verbal	*[Laughter], [Throat-clearing]*	1.8	2.3
Yes-Answers	*Yes.*	1.5	1.7
Conventional-closing	*Well, it's been nice talking to you.*	1.3	1.9
Other Labels(32)		9.1	9.8
Total number of sentences		196258	4186
Total number of dialogues		1115	19

Table 3: Dialogue act labels

4.2.1 Dataset

We use the Switch-board Dialogue Act (SwDA) corpus (Calhoun et al., 2010) in our experiments[5]. SwDA contains the transcripts of several people discussing a given topic on the telephone. There are 42 dialogue act tags in SwDA,[6] and we list the 10 most frequent tags in Table 3.

The same data split as in Stolcke et al. (2000) is used in our experiments. There are 1,115 dialogues in the training set and 19 dialogues in the test set[7]. We also randomly split the original training set as a new training set (1,085 dialogues) and a validation set (30 dialogues).

4.2.2 Setup

The setup is the same as that in Answer Selection except: (1) Only the most common 10,000 words are used, other words are all treated as unknown. (2) Each mini batch contains all sentences from 3 dialogues for Mini-batch Gradient Descent. (3) The evaluation metric is accuracy. (4) We run 30 epochs in total. (5) We use the last hidden state of LSTM_2 as sentence representation since the sentences here are much shorter compared with those in Answer Selection.

4.2.3 Baselines

We compare with the following baselines:

- unigram, bigram, trigram LM-HMM: HMM variants (Stolcke et al., 2000).
- RCNN: Recurrent Convolutional Neural Networks (Kalchbrenner and Blunsom, 2013). Sentences are first separately embedded with CNN, and then joined up with RNN.
- LSTM: All sentences are modeled separately by one LSTM. Different from SIN, there is no sentence interactions in this method.

Model	Accuracy(%)
unigram LM-HMM	68.2
bigram LM-HMM	70.6
trigram LM-HMM	71.0
RCNN	73.9
LSTM	72.8
SIN	74.8
SIN-CONV	75.1
SIN-LD	76.0
SIN-CONV-LD	**76.5**

Table 4: Accuracy on dialogue act analysis. Inter-annotator agreement is 84%.

4.2.4 Results

Results are shown in Table 4. HMM variants, RCNN and LSTM model the sentences separately during sentence embedding, and are unable to capture the sentence interactions. With our interaction mechanism, SIN outperforms LSTM, and proves that well modeling the interactions between sentences in a dialogue is important for dialogue act analysis. After introducing a convolution layer, SIN-CONV performs slightly better than SIN. SIN-LD and SIN-CONV-LD model the

[5] http://compprag.christopherpotts.net /swda.html.

[6] SwDA actually contains 43 tags in which "+" should not be treated as a valid tag since it means continuation of the previous sentence.

[7] http://web.stanford.edu/%7ejurafsky/ws97/

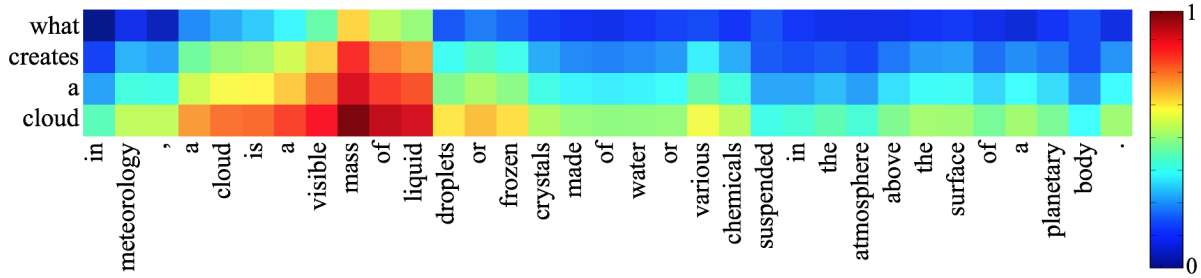

Figure 4: L_2-norm of the interaction states from question to answer (linearly mapped to $[0, 1]$).

Q:	*what creates a cloud*
A:	*in meteorology , a cloud is a visible mass of liquid droplets or frozen crystals made of water or various chemicals suspended in the atmosphere above the surface of a planetary body.*

Table 5: A question-answer pair example.

long-range dependence in the dialogue with another LSTM, and obtain further improvements.

4.3 Interaction Mechanism Analysis

We investigate into the interaction states of SIN for Answer Selection to see how our proposed interaction mechanism works.

Given a question-answer pair in Table 5, for SIN, there is a candidate interaction state $\tilde{c}_{\tau t}^{(i)}$ and an input gate $i_{\tau t}^{(i)}$ from each word t in the question to each word τ in the answer. We investigate into the L_2-norm $||\tilde{c}_{\tau t}^{(i)} * i_{\tau t}^{(i)}||_2$ to see how words in the two sentences interact with each other. Note that we have linearly mapped the original L_2-norm value to $[0, 1]$ as follows:

$$f(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$$

As depicted in Figure 4, we can see that the word *"what"* in the question has little impact to the answer through interactions. This is reasonable since *"what"* appears frequently in questions, and does not carry much information for answer selection[8]. On the contrary, the phrase *"creates a cloud"*, especially the word *"cloud"*, transmits much information through interactions to the answer, this conforms with human knowledge since we rely on these words to answer the question as well.

In the answer, interactions concentrate on the phrase *"a cloud is a visible mass of liquid droplets"* which seems to be a good and complete answer to the question. Although there are also other highly related words in the answer, they are almost ignored. The reason may be failing to model such a complex phrase (three relatively simple sentences joined by "or") or the existence of the previous phrase which is already a good answer.

This experiment clearly shows how the interaction mechanism works in SIN. Through interaction states, SIN is able to figure out what the question is asking about, namely to detect those highly informative words in the question, and which part in the answer can answer the question.

5 Conclusion and Future Work

In this work, we propose Sentence Interaction Network (SIN) which utilizes a new mechanism for modeling interactions between two sentences. We also introduce a convolution layer into SIN (SIN-CONV) to improve its phrase modeling ability so that phrase interactions can be handled. SIN is powerful and flexible to model sentence interactions for different tasks. Experiments show that the proposed interaction mechanism is effective, and we obtain significant improvements on Answer Selection and Dialogue Act Analysis without any handcrafted features.

Previous works have showed that it is important to utilize the syntactic structures for modeling sentences. We also find out that LSTM is sometimes unable to model complex phrases. So, we are going to extend SIN to tree-based SIN for sentence modeling as future work. Moreover, applying the models to other tasks, such as semantic relatedness measurement and paraphrase identification, would

[8]Our statements focus on the interaction, in a sense of *"answering" or "matching"*. Definitely, such words like "what" and "why" are very important for answering questions from the general QA perspective since they determine the type of answers.

also be interesting attempts.

6 Acknowledgments

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2012CB316301/2013CB329403, the National Science Foundation of China under grant No. 61272227/61332007, and the Beijing Higher Education Young Elite Teacher Project. The work was also supported by Tsinghua University – Beijing Samsung Telecom R&D Center Joint Laboratory for Intelligent Media Computing.

References

Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The nxt-format switchboard corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419.

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

Felix Gers. 2001. Long short-term memory in recurrent neural networks. *Unpublished PhD dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

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*, pages 2042–2050.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 367–374. Association for Computational Linguistics.

Gabriel Murray, Giuseppe Carenini, and Raymond Ng. 2010. Generating and validating abstracts of meeting conversations: a user study. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 105–113. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu, Xuan Zhu, and Xiaoyan Zhu. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1365–1374.

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.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Jason D Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 23–24. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models.

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*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*.