DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding

Tao Shen† Tianyi Zhou‡ Guodong Long†
Jing Jiang† Shirui Pan† Chengqi Zhang†

†Centre of Artificial Intelligence, University of Technology Sydney
‡Paul G. Allen School of Computer Science & Engineering, University of Washington
tao.shen@student.uts.edu.au, tianyizh@uw.edu
{guodong.long, jing.jiang, shirui.pan, chengqi.zhang}@uts.edu.au

Abstract

Recurrent neural nets (RNN) and convolutional neural nets (CNN) are widely used in NLP tasks to capture the long-term and local dependencies respectively. Attention mechanisms have recently attracted enormous interest due to their highly parallelizable computation, significantly less training time, and flexibility in modeling dependencies. We propose a novel attention mechanism in which the attention between elements from input sequence(s) is directional and multi-dimensional, i.e., feature-wise. A light-weight neural net, "Directional Self-Attention Network (DiSAN)", is then proposed to learn sentence embedding, based solely on the proposed attention without any RNN/CNN structure. DiSAN is only composed of a directional self-attention block with temporal order encoded, followed by a multi-dimensional attention that compresses the sequence into a vector representation. Despite this simple form, DiSAN outperforms complicated RNN/CNN models on both prediction quality and efficiency. It achieves the best test accuracy among all sentence encoding methods and improves the most recent best result by about 1.0% on the Stanford Natural Language Inference (SNLI) dataset, and shows the state-of-the-art test accuracy on the Stanford Sentiment Treebank (SST), Sentences Involving Compositional Knowledge (SICK), TREC Question-type Classification and Multi-Genre Natural Language Inference (MultiNLI) datasets.

1 Introduction

Context dependency plays a significant role in language understanding and provides critical information to natural language processing (NLP) tasks. For different tasks and data, researchers often switch between two types of deep neural network (DNN): recurrent neural network (RNN) with sequential architecture capturing long-range dependencies, e.g., long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) and gated recurrent unit (GRU) (Chung et al. 2014); and convolutional neural network (CNN) (Kim 2014) whose hierarchical structure is good at extracting local or position-invariant features. However, which one to choose in practice is an open question, and the choice largely relies on the empirical knowledge.

Recent works have found that equipping RNN or CNN with an attention mechanism can achieve the state-of-the-art performance on a large number of NLP tasks, including neural machine translation (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015), natural language inference (Liu et al. 2016), conversation generation (Shang, Lu, and Li 2015), question answering (Hermann et al. 2015; Sukhbaatar et al. 2015), machine reading comprehension (Seo et al. 2017), and sentiment classification (Kokkinos and Potamianos 2017). Attention uses a hidden layer to compute a categorical distribution over elements from the input sequence to reflect their importance weights. It allows RNN/CNN to maintain a variable-length memory, so that elements from the input sequence can be selected by their importance/relevance and merged into the output. In contrast to RNN and CNN, the attention mechanism is trained to capture the dependencies that make significant contributions to the task, regardless of the distance between the elements in the sequence. It can thus provide complementary information to the distance-aware dependencies modeled by RNN/CNN. In addition, computing attention only requires matrix multiplication, which is highly parallelizable compared to the sequential computation of RNN and two-dimensional convolution of CNN.

In a very recent work (Vaswani et al. 2017), an attention mechanism is solely used to construct a sequence to sequence (seq2seq) model that achieves a new state-of-the-art quality score in the neural machine translation (NMT) task. The seq2seq model, "Transformer", has an encoder-decoder structure that is only composed of stacked attention networks, without using either recurrence or convolution. The proposed attention, "multi-head attention", projects the input sequence to multiple subspaces, applies scaled dot-product attention (Rush, Chopra, and Weston 2015) to its representation in each subspace, and lastly concatenates their output. By doing this, it can combine different attentions from multiple subspaces. This mechanism is used in Transformer to compute both the self-context fusion features inside the encoder/decoder, and the bottleneck features between the encoder and decoder.

The attention mechanism has more flexibility in sequence length than RNN/CNN, and is more task/data-driven when modeling dependencies. Unlike sequential models, its computation can be easily and significantly accelerated by existing distributed/parallel computing schemes. However, to the best of our knowledge, a neural net entirely based on attention has not been designed for other NLP tasks except NMT, especially those that cannot be cast into a seq2seq problem.

Compared to RNN, a disadvantage of most attention mechanisms is that the temporal order information is lost, which however might be important to the task. This explains why positional encoding is applied to each sequence before being processed by attention in Transformer. How to model order information within attention is still an open problem.

The goal of this paper is to develop a unified RNN/CNN-free attention network that can be generally utilized to learn the sentence encoding model for different NLP tasks, such as natural language inference, sentiment classification, document classification and sentence relatedness. We focus on the sentence encoding model because it is the basic module of most DNNs used in the NLP literature.

We propose a novel attention mechanism that differs from previous ones in that it is 1) multi-dimensional: the attention w.r.t. each pair of elements from the source(s) is a vector, where each entry is the attention computed on each feature; and 2) directional: it uses one or multiple positional masks to model the asymmetric attention between two elements. We compute feature-wise attention since each element in a sequence is usually represented by a vector, e.g., word/character embedding (Kim et al. 2016), and attention on different features can contain different information about dependency. We apply masks to attention distribution since they can easily encode rich structured prior knowledge such as temporal order and dependency parsing. This design mitigates the weakness of attention in modeling order information, and takes full advantage of distributed/parallel computing. In contrast, involving a semantic parsing tree structure in the recursive models (Tai, Socher, and Manning 2015; Teng and Zhang 2017) substantially reduces the efficiency.

We build a light-weight and RNN/CNN-free neural network called "Directional Self-Attention Network (DiSAN)" for sentence encoding. This network relies entirely on the proposed attention and does not use any RNN/CNN structure. In DiSAN, the input sequence is processed by directional (forward and backward) self-attention to model context dependency and produce embedding for each element. Then, a multi-dimensional attention computes the embedding of the whole sequence, which can be invoked by a classification/regression model to compute the final prediction for a particular task. Unlike Transformer, neither stacking of attention blocks nor an encoder-decoder structure is required. The simple architecture of DiSAN leads to fewer parameters, less computation and easier parallelization.

In experiments, we compare DiSAN with the currently popular methods in natural language inference and sentiment classification tasks. DiSAN achieves the highest test accuracy on the Stanford Natural Language Inference (SNLI) dataset among sentence-encoding models and improves the currently best result by about 1.0%. It outperforms other methods and shows the state-of-the-art performance on the Stanford Sentiment Treebank (SST) dataset for sentiment classification. Meanwhile, it has fewer parameters and exhibits much higher computational efficiency than the models it outperforms, e.g., RNN, LSTM, CNN, memory networks, and tree-based models.

Annotation: 1) Lowercase letter denotes a vector; 2) bold lowercase letter denotes a sequence of vectors (stored as a matrix); and 3) capital letter denotes a matrix or a tensor.

2 Background

2.1 Sentence Encoding

In the pipeline for NLP tasks, a sentence is denoted by a sequence of discrete tokens (e.g., words or characters) $\boldsymbol{v} = [v_1, v_2, \dots, v_n]$, where v_i could be a one-hot vector whose dimension length equals the number of distinct tokens N. A pre-trained token embedding (e.g., word2vec (Mikolov et al. 2013) or GloVe (Pennington, Socher, and Manning 2014)) is applied to $\boldsymbol{v}$ and transforms all discrete tokens to a sequence of low-dimensional dense vector representations $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$ with $x_i \in \mathbb{R}^{d_e}$. This preprocessing can be written as $\boldsymbol{x} = W^{(e)}\boldsymbol{v}$, where word embedding weight matrix $W^{(e)} \in \mathbb{R}^{d_e \times N}$ and $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$.

Most DNN sentence-encoding models for NLP tasks take $\boldsymbol{x}$ as the input and further generate a vector representation u_i for each x_i by context fusion. Then a sentence encoding is obtained by mapping the sequence $\boldsymbol{u} = [u_1, u_2, \dots, u_n]$ to a single vector $s \in \mathbb{R}^d$, which is used as a compact encoding of the whole sentence in NLP problems.

2.2 Attention

Attention has been proposed to compute the alignment score between elements from two sources. In particular, given the token embeddings of a source sequence $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$ and the vector representation of a query q, attention computes the alignment score between x_i and q by a compatibility function $f(x_i, q)$, which measures the dependency between x_i and q, or the attention of q to x_i. A $\mathrm{softmax}$ function then transforms the scores $f(x_i, q), i = 1, \dots, n$ to probability distribution $p(z|\boldsymbol{x}, q)$ by normalizing over all the n tokens of $\boldsymbol{x}$. Here z is an indicator of which token in $\boldsymbol{x}$ is important to q in the task. That is, large $p(z = i|\boldsymbol{x}, q)$ means x_i contributes important information to q. The above process can be summarized by the following equations.

$$a = [f(x_i, q)]_{i=1}^n , \tag{1}$$

$$p(z|\boldsymbol{x}, q) = \mathrm{softmax}(a). \tag{2}$$

Specifically,

$$p(z = i|\boldsymbol{x}, q) = \frac{\exp(f(x_i, q))}{\sum_{i=1}^n \exp(f(x_i, q))}. \tag{3}$$

The output of this attention mechanism is a weighted sum of embeddings for all tokens in $\boldsymbol{x}$, where the weights are given by $p(z|\boldsymbol{x}, q)$. It places large weights on tokens important to q, and can be written as the expectation of a token sampled according to its importance, i.e.,

$$s = \sum_{i=1}^n p(z = i|\boldsymbol{x}, q)x_i = \mathbb{E}_{i \sim p(z|\boldsymbol{x},q)}(x_i), \tag{4}$$

where s can be used as the sentence encoding of $\boldsymbol{x}$.

Additive attention (or MLP attention) (Bahdanau, Cho, and Bengio 2015; Shang, Lu, and Li 2015) and multiplicative attention (or dot-product attention) (Vaswani et al. 2017; Sukhbaatar et al. 2015; Rush, Chopra, and Weston 2015) are the two most commonly used attention mechanisms. They share the same unified form of attention introduced above,

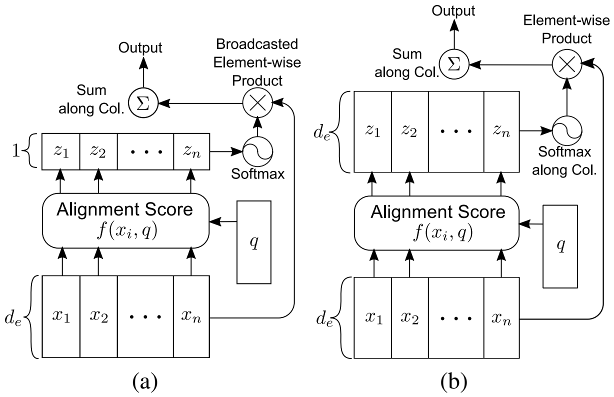


Figure 1: (a) Traditional (additive/multiplicative) attention and (b) multi-dimensional attention. z_i denotes alignment score $f(x_i, q)$, which is scalar in (a) but vector in (b).

but are different in the compatibility function $f(x_i, q)$. Additive attention is associated with

$$f(x_i, q) = w^T \sigma(W^{(1)} x_i + W^{(2)} q), \tag{5}$$

where $\sigma(\cdot)$ is an activation function and $w \in \mathbb{R}^{d_e}$ is a weight vector. Multiplicative attention uses inner product/cosine similarity for $f(x_i, q)$, i.e.,

$$f(x_i, q) = \left\langle W^{(1)} x_i, W^{(2)} q \right\rangle. \tag{6}$$

In practice, additive attention often outperforms multiplicative one in prediction quality, but the latter is faster and more memory-efficient due to optimized matrix multiplication.

2.3 Self-attention

Self-attention is a special case of the attention mechanism introduced above. It replaces q with a token embedding x_j from the source input itself. It relates elements at different positions from a single sequence by computing the attention between each pair of tokens x_i and x_j. It is very expressive and flexible for both long-range and local dependencies, which used to be respectively modeled by RNN and CNN. Moreover, it has much smaller computational complexity and fewer parameters than RNN/CNN. In recent works, we have already witnessed its success across a variety of NLP tasks, such as reading comprehension (Hu, Peng, and Qiu 2017) and machine translation (Vaswani et al. 2017).

3 Two Attention Mechanisms

In this section, we introduce two novel attention mechanisms, multi-dimensional attention in Section 3.1 (with two extensions to self-attention in Section 3.2) and directional self-attention in Section 3.3. They are the main components of DiSAN and may be of independent interest to other neural nets for other NLP problems in which attention is needed.

3.1 Multi-dimensional Attention

Multi-dimensional attention is a natural extension of additive attention (or MLP attention) on the feature level. Instead of computing a single scalar score $f(x_i, q)$ for each token x_i as shown in Eq.(5), multi-dimensional attention computes a feature-wise score vector for x_i by replacing weight vector w in Eq.(5) with a matrix W,

$$f(x_i, q) = W^T \sigma \left(W^{(1)} x_i + W^{(2)} q \right), \tag{7}$$

where $f(x_i, q)$ is a vector of the same length d_e as x_i and q, and all the weight matrices $W, W^{(1)}, W^{(2)} \in \mathbb{R}^{d_e \times d_e}$. We further add two bias terms for in and out activation $\sigma(\cdot)$, i.e.,

$$f(x_i, q) = W^T \sigma \left(W^{(1)} x_i + W^{(2)} q + b^{(1)} \right) + b. \tag{8}$$

We then compute a categorical distribution $p(z_k | \boldsymbol{x}, q)$ over all the n tokens for each feature $k \in [d_e]$. A large $p(z_k = i | \boldsymbol{x}, q)$ means that feature k of token i is important to q.

We apply the same procedure Eq.(1)-(3) in traditional attention to the k^{th} dimension of $f(x_i, q)$. In particular, for each feature $k \in [d_e]$, we replace $f(x_i, q)$ with $[f(x_i, q)]_k$, and change z to z_k in Eq.(1)-(3). Now each feature k in each token i has an importance weight $P_{ki} \triangleq p(z_k = i | \boldsymbol{x}, q)$. The output s can be written as

$$s = \left[\sum\nolimits_{i=1}^{n} P_{ki} \boldsymbol{x}_{ki} \right]_{k=1}^{d_e} = \left[\mathbb{E}_{i \sim p(z_k | \boldsymbol{x}, q)} (\boldsymbol{x}_{ki}) \right]_{k=1}^{d_e}. \tag{9}$$

We give an illustration of traditional attention and multi-dimensional attention in Figure 1. In the rest of this paper, we will ignore the subscript k indexing feature dimension for simplification if no confusion is possible. Hence, the output s can be written as an element-wise product $s = \sum_{i=1}^{n} P_{\cdot i} \odot x_i$

Remark: Word embedding usually suffers from the polysemy in natural language. Since traditional attention computes a single importance score for each word based on the word embedding, it cannot distinguish the meanings of the same word in different contexts. Multi-dimensional attention, however, computes a score for each feature of each word, so it can select the features that can best describe the word's specific meaning in any given context, and include this information in the sentence encoding output s.

3.2 Two types of Multi-dimensional Self-attention

When extended to self-attentions, we have two variants of multi-dimensional attention. The first one, called "**token2token**" multi-dimensional self-attention, explores the dependency between x_i and x_j from the same source $\boldsymbol{x}$, and generates context-aware coding for each element. It replaces q with x_j in Eq.(8):

$$f(x_i, x_j) = W^T \sigma \left(W^{(1)} x_i + W^{(2)} x_j + b^{(1)} \right) + b. \tag{10}$$

Similar to P in vanilla multi-dimensional attention, we compute a probability matrix $P^j \in \mathbb{R}^{d_e \times n}$ for each x_j such that $P^j_{ki} \triangleq p(z_k = i | \boldsymbol{x}, x_j)$. The output for x_j is:

$$s_j = \sum_{i=1}^{n} P^j_{\cdot i} \odot x_i \tag{11}$$

The output of token2token self-attention for all elements from $\boldsymbol{x}$ is $\boldsymbol{s} = [s_1, s_2, \ldots, s_n] \in \mathbb{R}^{d_e \times n}$.

The second one, called "**source2token**" multi-dimensional self-attention, explores the dependency between x_i and the whole sequence $\boldsymbol{x}$, and compresses the sequence $\boldsymbol{x}$ into a vector. It removes q from Eq.(8):

$$f(x_i) = W^T \sigma \left(W^{(1)} x_i + b^{(1)} \right) + b. \tag{12}$$

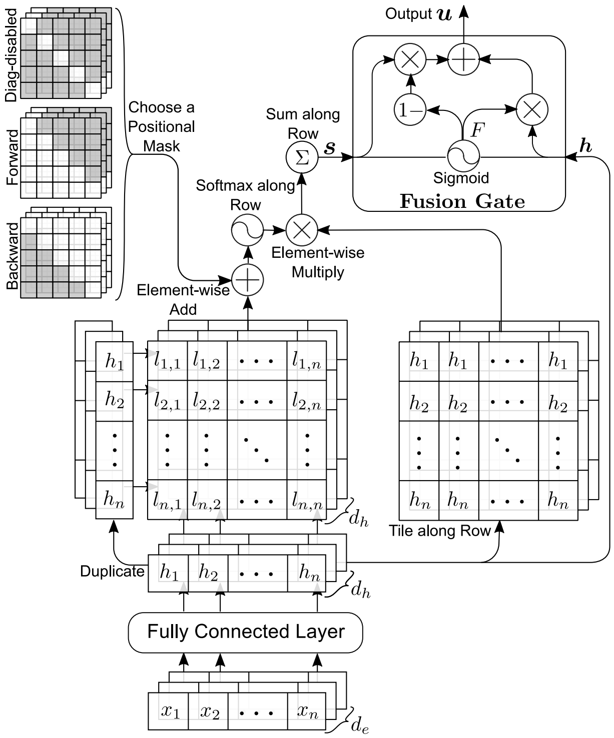


Figure 2: Directional self-attention mechanism: here we use $l_{i,j}$ to denote $f(h_i, h_j)$ in Eq. (15).

The probability matrix is defined by $P_{ki} \triangleq p(z_k = i|\boldsymbol{x})$ and is computed in the same way as P in vanilla multi-dimensional attention. The output s is also same, i.e.,

$$s = \sum_{i=1}^{n} P_{\cdot i} \odot x_i \tag{13}$$

We will use these two types (i.e. token2token and source2token) of multi-dimensional self-attention in different parts of our sentence encoding model DiSAN.

3.3 Directional Self-attention

Directional self-attention is composed of a fully connected layer whose input is the token embeddings $\boldsymbol{x}$, a masked token2token multi-dimensional self-attention block to explore the dependency and temporal order, and a fusion gate to combine the output and input of the attention block. Its structure is shown in Figure 2. It can be used as either a neural net or a module to compose a large neural net.

In directional self-attention, we first transform the input sequence $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$ to a sequence of hidden state $\boldsymbol{h} = [h_1, h_2, \dots, h_n]$ by a fully connected layer, i.e.,

$$\boldsymbol{h} = \sigma_h \left(W^{(h)} \boldsymbol{x} + b^{(h)} \right), \tag{14}$$

where $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$ and $\boldsymbol{h} \in \mathbb{R}^{d_h \times n}$, $W^{(h)}$ and $b^{(h)}$ are the learnable parameters, and $\sigma_h(\cdot)$ is an activation function.

We then apply token2token multi-dimensional self-attention to $\boldsymbol{h}$, and generate a context-aware vector representations $\boldsymbol{s}$ for all elements from the input sequence. We make two modifications to Eq.(10) to reduce number of parameters and make attention directional.

First, we set W in Eq.(10) to be a scalar c and divide the part in $\sigma(\cdot)$ by c, and we use $\tanh(\cdot)$ for $\sigma(\cdot)$. This reduces the number of parameters. In experiments, we always set $c = 5$, and obtain stable output.

We use a scaled $\tanh$ as the activation function for two reasons. First, compared to identity function, scaled $\tanh$ has fixed upper and lower bounds to prevent large difference among scores. Then, compared to $\tanh$ without scaling, scaled $\tanh$ has a larger range $(-c, c)$ for output. The large difference results in large difference in $\mathrm{softmax}$ probabilities. Therefore, scaled $\tanh$ controls a tradeoff between the heterogenization and homogenization of attention probabilities. Second, scaled $\tanh$ turns to be an effective nonlinear transformation for attention: in Eq. (15), 1) if $W^{(2)}h_j + b^{(1)}$ is close to zero, scaled $\tanh$ outputs a value similar to $W^{(1)}h_i$; 2) if $W^{(2)}h_j + b^{(1)}$ is a large positive value, scaled $\tanh$ tends to treat positive $W^{(1)}h_i$ with different values similarly, and negative $W^{(1)}h_i$ with different values differently; 3) if $W^{(2)}h_j + b^{(1)}$ is a large negative value, on the contrary, scaled $\tanh$ tends to treat negative $W^{(1)}h_i$ with different values similarly, and positive $W^{(1)}h_i$ with different values differently.

Second, we apply a positional mask to Eq.(10), so the attention between two elements can be asymmetric. Given a mask $M \in \{0, -\infty\}^{n \times n}$, we use bias $b = M_{ij}\mathbf{1}$ in Eq.(10), where $\mathbf{1}$ is an all-ones vector. Hence, Eq.(10) is modified to

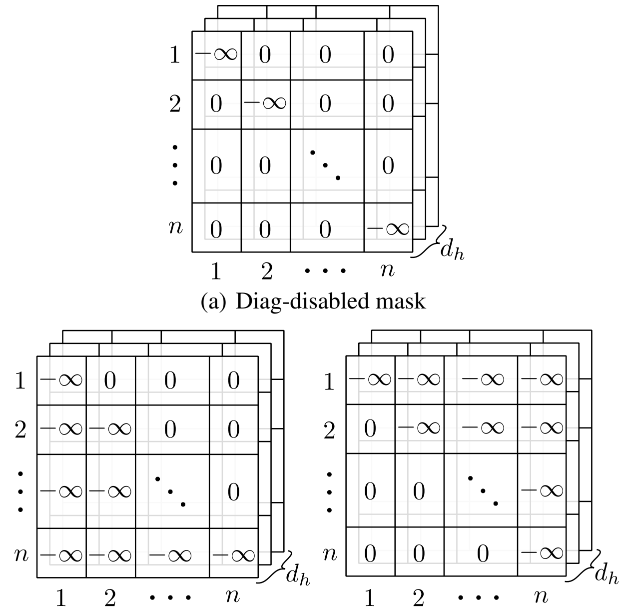
$$\begin{aligned} &f(h_i, h_j) = \\ &c \cdot \tanh\left([W^{(1)}h_i + W^{(2)}h_j + b^{(1)}]/c \right) + M_{ij}\mathbf{1}. \end{aligned} \tag{15}$$

To see why a mask can encode directional information, let us consider a case in which $M_{ij} = -\infty$ and $M_{ij} = 0$, which results in $[f(h_i, h_j)]_k = -\infty$ and unchanged $[f(h_j, h_i)]_k$. Since the probability $p(z_k = i|\boldsymbol{x}, x_j)$ is computed by $\mathrm{softmax}$, $[f(h_i, h_j)]_k = -\infty$ leads to $p(z_k = i|\boldsymbol{x}, x_j) = 0$. This means that there is no attention of x_j to x_i on each feature k. On the contrary, we have $p(z_k = j|\boldsymbol{x}, x_i) > 0$, which means that attention of x_i to x_j exists. Therefore, rich structured prior knowledge such as temporal order and dependency parsing can be easily encoded by the mask, and explored in generating sentence encoding. This is an important feature of directional self-attention that previous attention mechanisms do not have.

For self-attention, we usually need to disable the attention of each token to itself (Hu, Peng, and Qiu 2017). This is the same as applying a diagonal-disabled (i.e., diag-disabled) mask such that

$$M_{ij}^{diag} = \begin{cases} 0, & i \neq j \\ -\infty, & i = j \end{cases} \tag{16}$$

Moreover, we can use masks to encode temporal order information into attention. In this paper, we use two masks,



(a) Diag-disabled mask

(b) Forward mask

(c) Backward mask

Figure 3: Three positional masks: (a) is the diag-disabled mask M^{diag}; (b) and (c) are forward mask M^{fw} and backward mask M^{bw}.

i.e., forward mask M^{fw} and backward mask M^{bw},

$$M_{ij}^{fw} = \begin{cases} 0, & i < j \\ -\infty, & otherwise \end{cases} \tag{17}$$

$$M_{ij}^{bw} = \begin{cases} 0, & i > j \\ -\infty, & otherwise \end{cases} \tag{18}$$

In forward mask M^{fw}, there is the only attention of later token j to early token i from the input sequence, and vice versa for backward mask. We show these three masks in Figure 3.

Given input sequence $\boldsymbol{x}$ and a mask M, we compute $f(x_i, x_j)$ according to Eq.(15), and follow the standard procedure of token2token multi-dimensional self-attention to compute the probability matrix P^j for each $j \in [n]$. Each output s_j in $\boldsymbol{s}$ is computed as in Eq.(11).

The final output $\boldsymbol{u} \in \mathbb{R}^{d_h \times n}$ of directional self-attention is obtained by combining the output $\boldsymbol{s}$ and the input $\boldsymbol{h}$ of the token2token multi-dimensional self-attention block. This yields a temporal order encoded and context-aware vector representation for each element/token. The combination is accomplished by a dimension-wise fusion gate:

$$F = \text{sigmoid}\left(W^{(f1)}\boldsymbol{s} + W^{(f2)}\boldsymbol{h} + b^{(f)}\right) \tag{19}$$

$$\boldsymbol{u} = F \odot \boldsymbol{h} + (1 - F) \odot \boldsymbol{s} \tag{20}$$

where $W^{(f1)}, W^{(f2)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(f)} \in \mathbb{R}^{d_h}$ are the learnable parameters of the fusion gate.

4 Directional Self-Attention Network

We propose a light-weight network "Directional Self-Attention Network (DiSAN)" for sentence encoding. Its architecture is shown in Figure 4.

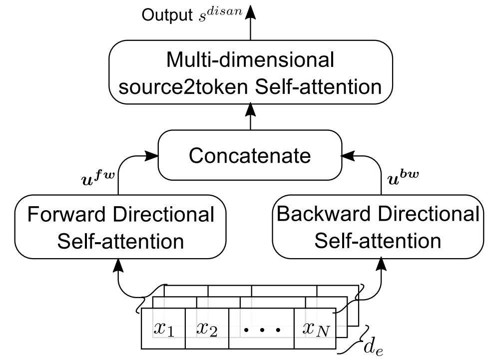


Figure 4: Directional Self-attention Network (DiSAN)

Given an input sequence of token embedding $\boldsymbol{x}$, DiSAN firstly applies two parameter-untied directional self-attention blocks with forward mask M^{fw} Eq.(17) and M^{bw} Eq.(18) respectively. The feed-forward procedure is given in Eq.(14)-(15) and Eq.(19)-(20). Let their output to be $\boldsymbol{u^{fw}}, \boldsymbol{u^{bw}} \in \mathbb{R}^{d_h \times n}$. We concatenate them vertically as $[\boldsymbol{u^{fw}}; \boldsymbol{u^{bw}}] \in \mathbb{R}^{2d_h \times n}$, and use this concatenated output as input to a source2token multi-dimensional self-attention block, whose output $s^{disan} \in \mathbb{R}^{2d_h}$ computed by Eq.(12)-(13) is the final sentence encoding result of DiSAN.

Remark: In DiSAN, forward/backward directional self-attention blocks work as context fusion layers. And the source2token multi-dimensional self-attention compresses the sequence into a single vector. The idea of using both forward and backward attentions is inspired by Bi-directional LSTM (Bi-LSTM) (Graves, Jaitly, and Mohamed 2013), in which forward and backward LSTM are used to encode long-range dependency. In Bi-LSTM, LSTM combines the context-aware output with the input by multi-gate. The fusion gate used in directional self-attention shares similar motivation. However, DiSAN has fewer parameters, simpler structure and better efficiency.

5 Experiments

In this section, we mainly apply DiSAN to natural language inference and sentiment classification tasks, and analyze the prediction results of DiSAN for both two tasks in detail. We also show experiments on three more NLP tasks to verify the sentence encoding capability of DiSAN, including sentence semantic relatedness, question-type classification and multi-genre natural language inference. Experiment results show that DiSAN achieves the state-of-the-art performance and significantly better efficiency than other baseline methods on benchmark datasets for these tasks.

Training Setup: We use cross-entropy loss plus L2 regularization with decay factor γ as optimization objective. We minimize it by Adadelta (Zeiler 2012) (an optimizer of mini-batch SGD) with batch size 64. We use Adadelta rather than Adam (Kingma and Ba 2015), because in our experiments DiSAN optimized by Adadelta can achieve more stable performance and better accuracy than Adam optimized one. All weight matrices are initialized by Glorot Initialization (Glorot and Bengio 2010), and the biases are initialized to be 0.

Model Name	$\|\boldsymbol{\theta}\|$	T(s)/epoch	Train Accu(%)	Test Accu(%)
Unlexicalized features (Bowman et al. 2015)			49.4	50.4
+ Unigram and bigram features (Bowman et al. 2015)			99.7	78.2
100D LSTM encoders (Bowman et al. 2015)	0.2m		84.8	77.6
300D LSTM encoders (Bowman et al. 2016)	3.0m		83.9	80.6
1024D GRU encoders (Vendrov et al. 2016)	15m		98.8	81.4
300D Tree-based CNN encoders (Mou et al. 2016)	3.5m		83.3	82.1
300D SPINN-PI encoders (Bowman et al. 2016)	3.7m		89.2	83.2
600D Bi-LSTM encoders (Liu et al. 2016)	2.0m		86.4	83.3
300D NTI-SLSTM-LSTM encoders (Munkhdalai and Yu 2017b)	4.0m		82.5	83.4
600D Bi-LSTM encoders+intra-attention (Liu et al. 2016)	2.8m		84.5	84.2
300D NSE encoders (Munkhdalai and Yu 2017a)	3.0m		86.2	84.6
Word Embedding with additive attention	0.45m	216	82.39	79.81
Word Embedding with multi-dimensional attention	0.54m	261	86.22	83.12
600D Bi-LSTM encoders with multi-dimensional attention	2.88m	2080	90.39	84.53
Two self-attention with multi-dimensional attention	2.35m	592	90.18	84.66
Directional self-attention network (DiSAN)	2.35m	587	91.08	**85.57**

Table 1: Experimental results of different methods on SNLI. $|\boldsymbol{\theta}|$: number of parameters (excluding word embedding part). **T(s)/epoch**: average time (second) per epoch. **Train Accu(%)** and **Test Accu(%)**: accuracy on training and test set.

We initialize the word embedding in $\boldsymbol{x}$ by 300D GloVe 6B pre-trained vectors (Pennington, Socher, and Manning 2014). The Out-of-Vocabulary words in training set are randomly initialized by uniform distribution between $(-0.05, 0.05)$. The word embedding's parameters are fine-tuned during training for a better performance (Tai, Socher, and Manning 2015).

We use Dropout (Srivastava et al. 2014) with keep probability 0.75 for language. The decay factor γ is 5×10^{-5}. Note that the dropout keep probability and γ varies with the scale of corresponding dataset. Initial learning rate is set to 0.5. Hidden units number $d_h = 300$. Activation functions $\sigma(\cdot)$ are *ELU* (exponential linear unit) (Clevert, Unterthiner, and Hochreiter 2016) if not specified. All models are implemented by TensorFlow[1] and are running on single Nvidia Geforce GTX 1080Ti GPU.

5.1 Natural Language Inference

The goal of Natural Language Inference (NLI) is to reason the semantic relationship between a promise sentence and a hypothesis sentence. The possible relation could be *entailment*, *neural* or *contradiction*. We compare different models on a widely used benchmark dataset Stanford Natural Language Inference (SNLI)[2] (Bowman et al. 2015) which consists of 549,367/9,842/9,824 (train/dev/test) promise-hypothesis pairs with relationship labels.

Following the standard procedure in (Bowman et al. 2015; 2016), we launch two sentence encoding models (e.g., DiSAN) with tied parameters for the promise sentence and hypothesis sentence respectively. Given the output encoding s^p for s^{disan} of promise and s^h for s^{disan} of hypothesis, the relation representation is the concatenation of s^p, s^h, $s^p - s^h$ and $s^p \odot s^h$, which is fed into a 300D fully connected layer and then a 3-unit output layer with $\mathrm{softmax}$ to compute the probability distribution over the three types of relations.

For thorough comparison, besides the neural nets proposed in previous works of NLI, we implement four extra neural nets baselines to compare with DiSAN. They help us to analyze the improvement contributed by each part of DiSAN and to verify that the two attention mechanisms in Section 3 can improve other networks.

- **Word Embedding with additive attention**.
- **Word Embedding with multi-dimensional attention:** DiSAN with directional self-attention blocks removed.
- **Bi-LSTM encoders with multi-dimensional attention:** a source2token multi-dimensional self-attention block is applied to the output of Bi-LSTM (300D forward LSTM + 300D backward LSTM).
- **Two self-attention with multi-dimensional attention:** DiSAN with the forward/backward masks M^{fw} and M^{bw} replaced by two diag-disabled masks M^{diag}, i.e., DiSAN without "directional".

Compared to the results from the official leaderboard of SNLI in Table 1, DiSAN outperforms previous works in both accuracy and time-efficiency. It improves the best latest test accuracy (achieved by a memories-based NSE encoder network) by a remarkable margin of 0.97%. DiSAN surpasses the RNN/CNN based models with more complicated architecture and more parameters by large margins, e.g., $+2.27\%$ to Bi-LSTM, $+1.37\%$ to Bi-LSTM with additive attention. It even outperforms models that have the assistance of a semantic parsing tree, i.e., $+3.47\%$ to Tree-based CNN, $+2.37\%$ to SPINN-PI.

In the results of the four baseline methods and DiSAN at the bottom of Table 1, we demonstrate that making attention multi-dimensional (feature-wise) or directional brings substantial improvement to different neural nets. First, a comparison between the first two models shows that changing

[1] https://www.tensorflow.org

[2] https://nlp.stanford.edu/projects/snli/

Model	Test Accu
MV-RNN (Socher et al. 2013)	44.4
RNTN (Socher et al. 2013)	45.7
Bi-LSTM (Li et al. 2015)	49.8
Tree-LSTM (Tai, Socher, and Manning 2015)	51.0
CNN-non-static (Kim 2014)	48.0
CNN-Tensor (Lei, Barzilay, and Jaakkola 2015)	51.2
NCSL (Teng, Vo, and Zhang 2016)	51.1
LR-Bi-LSTM (Qian, Huang, and Zhu 2017)	50.6
DiSAN	**51.72**

Table 2: Test accuracy of fine-grained sentiment classification on Stanford Sentiment Treebank (SST).

token-wise attention to multi-dimensional feature-wise attention leads to 3.31% improvement on a word-embedding based model. Also, a comparison between the third baseline and DiSAN shows that the directional self-attention block can even outperform Bi-LSTM layer in context encoding, improving test accuracy by 1.04%. A comparison between the fourth baseline and DiSAN shows that directional attention with forward and backward masks (temporal order encoded) can bring 0.91% improvement.

Additional advantages of DiSAN shown in Table 1 are its fewer parameters and compelling efficiency. It is $\times 3$ faster than widely used Bi-LSTM model. Comparing to other models with competitive performance, e.g., 600D Bi-LSTM encoders with intra-attention (2.8M), 300D NSE encoders (3.0M) and 600D Bi-LSTM encoders with multi-dimensional attention (2.88M), DiSAN only has 2.35M parameters.

5.2 Sentiment Classification

Sentiment classification aims to analyze the sentiment of a sentence or a paragraph, e.g., a movie or product review. We use Stanford Sentiment Treebank (SST)[3] (Socher et al. 2013) for the experiments, and only focus on the fine-grained movie review sentiment classification over five classes: very negative, negative, neutral, positive and very positive. We use the standard train/dev/test sets splitting with 8,544/1,101/2,210 samples. Similar to Section 5.1, we employ a single sentence encoding model to obtain a sentence representation s^{disan} of the movie review, then pass it into a 300D fully connected layer. Finally, a 5-unit output layer with $\mathrm{softmax}$ is used to calculate the probability distribution over the five classes.

In Table 2, we compare previous works with DiSAN on test accuracy. To the best of our knowledge, DiSAN improves the last best accuracy (given by CNN-Tensor) by 0.52%. Compared to tree-based models with heavy use of the structured prior, e.g., MV-RNN, RNTN and Tree-LSTM, DiSAN outperforms them by 7.32%, 6.02% and 0.72% respectively. In addition, DiSAN achieves better performance than CNN-based sequential models. More recent works tend to focus on lexicon-based sentiment classification, by exploring sentiment lexicons, negation words and intensity words. Nonetheless, DiSAN still outperforms these fancy models, such as NCSL ($+0.62\%$) and LR-Bi-LSTM ($+1.12\%$).

It is also interesting to see the performance of different models on sentences of different lengths. In Figure 5, we compare LSTM, Bi-LSTM, Tree-LSTM and DiSAN on different sentence lengths. In the range of $(5, 12)$, which is the range of length for most movie review sentences, DiSAN significantly outperforms others. Meanwhile, DiSAN also shows impressive performance for slightly longer sentences or paragraphs in the range of $(25, 38)$. DiSAN performs poorly when the sentence length ≥ 38, in which however only 3.21% of total movie review sentences lie.

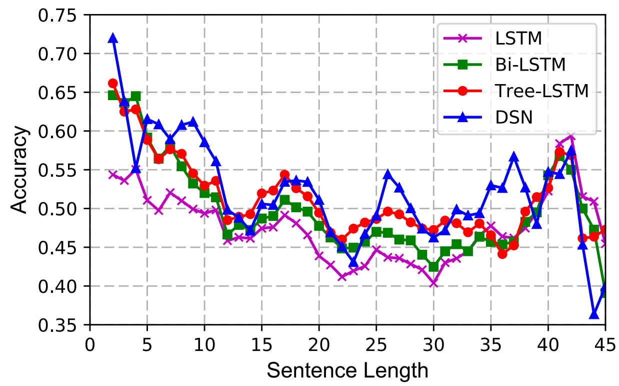


Figure 5: Fine-grained sentiment classification accuracy on sentences of different length. The results of LSTM, Bi-LSTM and Tree-LSTM are from Tai, Socher, and Manning (2015) and the result of DiSAN is the average over five random trials.

We also study the reason why the fine-grained sentiment classification accuracy is hard to exceed 50%. We show the confusion matrix in Table 3. It shows that most classification errors are caused by confusion between very positive and positive, between very negative and negative, and between neutral and not neutral.

Label \ Pred.	Very Neg.	Neg.	Neu.	Pos.	Very Pos.
Very Neg.	106	153	1	18	1
Neg.	72	447	20	91	3
Neu.	13	185	45	140	6
Pos.	1	65	15	351	78
Very Pos.	0	15	3	178	203

Table 3: Confusion matrix of DiSAN prediction on Stanford Sentiment Treebank (SST) test set. Pred., Neg., Neu. and Pos. are abbreviations of prediction, negative, neutral and positive.

5.3 Experiments on Other Three NLP Tasks

Besides natural language inference and sentiment classification tasks shown in the main paper, DiSAN also achieves the best or the state-of-the-art performance on three benchmark

[3]https://nlp.stanford.edu/sentiment/

Method	Pearson's r	Spearman's ρ	MSE
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	-	-
DT-RNN (Socher et al. 2014)	0.7923	0.7319	0.3822
SDT-RNN (Socher et al. 2014)	0.7900	0.7304	0.3848
LSTM (Tai, Socher, and Manning 2015)	0.8528	0.7911	0.2831
Bidirectional LSTM (Tai, Socher, and Manning 2015)	0.8567	0.7966	0.2736
Skip-Thought (Kiros et al. 2015)	0.8655	0.7995	0.2561
Constituency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8582	0.7966	0.2734
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	**0.2532**
DiSAN	**0.8704**	**0.8164**	0.2861

Table 4: Experimental results of different methods on SICK sentence relatedness.

datasets for other three NLP tasks, i.e., sentence semantic relatedness, question-type classification and multi-genre natural language inference. The following experiments show that DiSAN consistently improves the best latest results on different NLP tasks. Therefore, DiSAN can be applied to a broad range of problems and has the potential capability to improve the currently best results.

Sentence Semantic Relatedness The task of sentence semantic relatedness aims to predict a similarity rating of a given pair of sentences. We show the experimental comparison of different methods on Sentences Involving Compositional Knowledge (SICK)[4] dataset (Marelli et al. 2014). SICK is composed of 9,927 sentence pairs with 4,500/500/4,927 instances for train/dev/test. The regression neural network on top of DiSAN is replaced by a network introduced by Tai, Socher, and Manning (2015) for the same task. The results in Table 4 show that DiSAN outperforms models from previous works in terms of Pearson's $\boldsymbol{r}$ and Spearman's $\boldsymbol{\rho}$ indexes.

Method	Test Accu
cBoW (Zhao, Lu, and Poupart 2015)	87.3
RNN (Zhao, Lu, and Poupart 2015)	90.2
CNN (Kim 2014)	93.6
AdaSent (Zhao, Lu, and Poupart 2015)	92.4
Skip-Thought (Kiros et al. 2015)	92.2
DiSAN	**94.4**

Table 5: Experimental results of different methods on TREC question-type classification dataset.

Question-type Classification Question-type classification aims to classify a question into different types. We use TREC[5] (Li and Roth 2002) question-type classification dataset for evaluation. TREC has 5,452/500 samples for training/test, and all the questions are coarsely classified into six types. The results of all methods are shown in Table 5, which shows that DiSAN achieves much better test accuracy than previous methods applied to TREC. In addition, to the best of our knowledge, it improves the best latest result by a large margin 2.2%.

Multi-Genre Natural Language Inference Multi-Genre Natural Language Inference (MultiNLI)[6] (Williams, Nangia, and Bowman 2017) dataset consist of 433k sentence pairs annotated with textual entailment information. This dataset is similar to SNLI, but it covers more genres of spoken and written text, and supports a distinctive cross-genre generalization evaluation. MultiNLI is a quite new dataset, and its leaderboard does not include a session for the sentence-encoding only model. Hence, we only compare DiSAN with the baselines provided at the official website. The results of DiSAN and two sentence-encoding models on the leaderboard are shown in Table 6. DiSAN achieves the best matched and mismatched accuracy among these three models.

Method	Matched	Mismatched
cBoW	0.65200	0.64759
Bi-LSTM	0.67507	0.67248
DiSAN	**0.70977**	**0.71402**

Table 6: Experimental results of prediction accuracy of different methods on MultiNLI.

5.4 Case Study

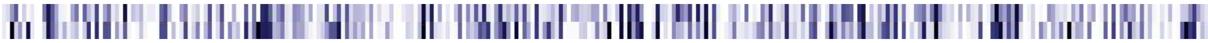
To gain a closer view of what dependencies in a sentence can be captured by DiSAN, we visualize the attention probability $p(z = i|\boldsymbol{x}, x_j)$ or alignment score in DiSAN by heatmaps to show the dependency of word j on i. In particular, we will focus primarily on the probability in forward/backward directional self-attention blocks (Figure 7), forward/backward fusion gates F in Eq.(19) (Figure 8), and the probability in source2token multi-dimensional self-attention block(Figure

[4]http://clic.cimec.unitn.it/composes/sick.html

[5]http://cogcomp.org/Data/QA/QC/

[6]https://www.nyu.edu/projects/bowman/multinli/

(a) *glass* in pair 1

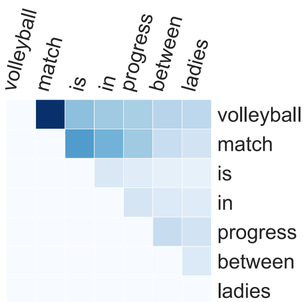


(b) *close* in pair 2

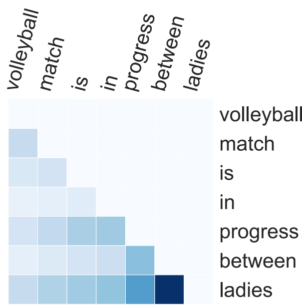
Figure 6: Two pairs of attention probability comparison of same word in difference sentence contexts.

7). For the first two, we desire to demonstrate the dependency between words, but attention probability in DiSAN is defined on each feature, so we average the probabilities along the feature dimension.

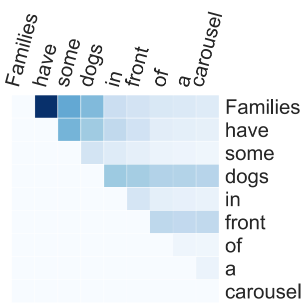
We select two sentences from SNLI test set as examples for this case study. Sentence 1 is *Families have some dogs in front of a carousel* and sentence 2 is *volleyball match is in progress between ladies*.



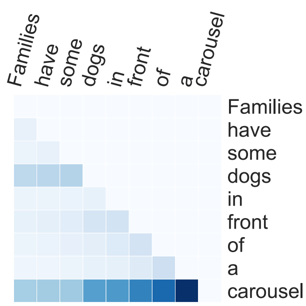
(a) Sentence 1, forward



(b) Sentence 1, backward



(c) Sentence 2, forward



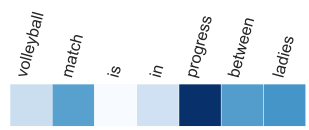
(d) Sentence 2, backward

Figure 7: Attention probability in forward/backward directional self-attention blocks.

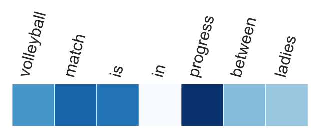
In Figure 7: 1) semantically important words such as nouns and verbs usually get large attention, but stop words (*am, is, are* etc.) do not; 2) globally important words, e.g., *volleyball, match, ladies* in sentence 1 and *dog, front, carousel* in sentence 2, get large attention from all other words; 3) if a word is important to only some of the other words (e.g. to constitute a phrase or sense-group), it gets large attention only from these words, e.g., attention between *progress, between* in sentence1, and attention between *families, have* in sentence 2.

This shows that directional information can help to generate context-aware word representation with temporal order encoded. For instance, for word *match* in sentence 1, its forward directional self-attention focuses more on word *volleyball*, while its backward attention focuses more on *progress* and *ladies*, so the representation of word *match* contains the essential information of the entire sentence, and simultaneously includes the positional order information.

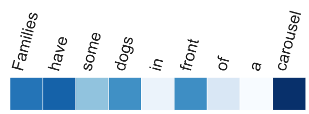
In addition, forward and backward directional self-attentions can focus on different parts of a sentence. For example, the forward one in sentence 2 pays more attention to the word *families*, whereas the backward one focuses more on the word *carousel*. Since forward and backward attention are computed separately, it avoids normalization over multiple significant words to weaken their weights. Note that this is a weakness of traditional attention compared to RNN, especially for long sentences.



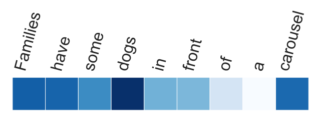
(a) Sentence 1, forward



(b) Sentence 1, backward



(c) Sentence 2, forward



(d) Sentence 2, backward

Figure 8: Fusion Gate F in forward/backward directional self-attention blocks.

In Figure 8, we show gate value F in Eq.(19). The gate combines the input and output of self-attention: it tends to selects the input representation $\boldsymbol{h}$ instead of the output $\boldsymbol{s}$ in directional self-attention, if the corresponding weight in F is large. This shows that the gate values for meaningless words, especially stop words is small. The stop words themselves cannot contribute important information: only their semantic relations to other words might help to understand the sentence. Hence, the gate tends to use their context-aware representation given by directional self-attention.

In addition, We visualize source2token multi-dimensional self-attention of two cases from SNLI dataset here. In Figure 6, we show attention scores on each feature of the same word in different sentences (with different context). We study two pairs of sentences, where the two sentences in each pair contain the word with same morphology but different semantics, i.e., polysemy.

The first pair has two sentences: one is *The **glass** bottle is big*, and the other is *A man is pouring a **glass** of tea*. They share the same word is *glass* with different meanings. The second pair has two sentences: one is *The restaurant is about to* ***close*** and the other is *A biker is* ***close*** *to the fountain*, where the same word is *close*.

In Figure 6, for each pair, we show the two multi-dimensional attention score vectors of the same word in the two sentences, by their heatmaps. It can be seen that the two attention vectors for the same words are very different due to their different meanings in different contexts. In addition, for a word in a sentence, the attention scores on different features are very different. This indicates that the multi-dimensional attention vector is not redundant: it can encode more information than one single score used in traditional attention. Note that, in traditional attention, we only have two single scores to encode the differences between two different contexts rather than two vectors. Hence, multi-dimensional attention is able to capture subtle difference of the same word in different sentences.

6 More Related Work

The attention mechanism is a novel strategy to improve the neural network performance in a wide variety of natural language processing (NLP) tasks. Neural machine translation (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015) is first task employing the attention to alleviate the bottleneck of sequence compressed by LSTM (Sutskever, Vinyals, and Le 2014). Similar to machine translation task, the sequence to sequence tasks, such as conversation generation (Shang, Lu, and Li 2015) etc. are implemented with attention, which can achieve impressive effectiveness. To obtain long-term dependency context-aware sequence representation, structured attention (Kim et al. 2017; Liu and Lapata 2017) and self-alignment attention (Hu, Peng, and Qiu 2017) are proposed, using cosine similarity between each pair of element form input sequence to mitigate the weakness of LSTM on long-term dependency. Our directional self-attention network exists with three-fold differences compared to the Transformer. First, we focus more on the context-aware word and semantic sentence embedding instead of sequence to sequence task in Transformer. Second, we use a novel dimension-wise relevance metric instead of calculating barely single cosine similarity between query and keys mentioned in Transformer. Third, we employ the forward and backward masks to perform the positional encoding rather than masking with pre-set positional codings.

Because sentence encoding is a vital preliminary for a majority of NLP tasks, numerous sentence encoding methods have been proposed. RNN (Elman 1990), especially LSTM (Hochreiter and Schmidhuber 1997) is the most pervasive approach to generate sentence encoding. In the light of the decent performance of LSTM, Bi-LSTM (Graves, Jaitly, and Mohamed 2013) based sentence encoding method is proposed to capture forward and backward context knowledge for sentence representation. Then, to embed the semantic structure information into the sentence representation, recursive neural network (Socher et al. 2013) and Tree-LSTM (Tai, Socher, and Manning 2015) are proposed based on the sentence dependency or constituency parsing result. Besides, attention based sentence encoding with mean pooling (Liu et al. 2016) is proposed to automatically focus a significant part of input sequence then generates more effective sentence representation.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bjerva, J.; Bos, J.; Van der Goot, R.; and Nissim, M. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *SemEval@ COLING*, 642–646.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Bowman, S. R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C. D.; and Potts, C. 2016. A fast unified model for parsing and sentence understanding. In *ACL*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS*.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*.

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

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Graves, A.; Jaitly, N.; and Mohamed, A.-r. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 273–278. IEEE.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 1693–1701.

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

Hu, M.; Peng, Y.; and Qiu, X. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798*.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *AAAI*, 2741–2749.

Kim, Y.; Denton, C.; Hoang, L.; and Rush, A. M. 2017. Structured attention networks. In *ICLR*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, 3294–3302.

Kokkinos, F., and Potamianos, A. 2017. Structural attention neural networks for improved sentiment analysis. *arXiv preprint arXiv:1701.01811*.

Lei, T.; Barzilay, R.; and Jaakkola, T. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*.

Li, X., and Roth, D. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, 1–7. Association for Computational Linguistics.

Li, J.; Luong, M.-T.; Jurafsky, D.; and Hovy, E. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Liu, Y., and Lapata, M. 2017. Learning structured text representations. *arXiv preprint arXiv:1705.09207*.

Liu, Y.; Sun, C.; Lin, L.; and Wang, X. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; and Zamparelli, R. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, 216–223.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Mou, L.; Men, R.; Li, G.; Xu, Y.; Zhang, L.; Yan, R.; and Jin, Z. 2016. Natural language inference by tree-based convolution and heuristic matching. In *ACL*.

Munkhdalai, T., and Yu, H. 2017a. Neural semantic encoders. In *EACL*.

Munkhdalai, T., and Yu, H. 2017b. Neural tree indexers for text understanding. In *EACL*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Qian, Q.; Huang, M.; and Zhu, X. 2017. Linguistically regularized lstms for sentiment classification. In *ACL*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Shang, L.; Lu, Z.; and Li, H. 2015. Neural responding machine for short-text conversation. In *ACL*.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; Potts, C.; et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, 1642.

Socher, R.; Karpathy, A.; Le, Q. V.; Manning, C. D.; and Ng, A. Y. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2:207–218.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, 2440–2448.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Teng, Z., and Zhang, Y. 2017. Bidirectional tree-structured lstm with head lexicalization. In *ACL*.

Teng, Z.; Vo, D.-T.; and Zhang, Y. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *EMNLP*, 1629–1638.

Vaswani, A.; Shazeer; Noam; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vendrov, I.; Kiros, R.; Fidler, S.; and Urtasun, R. 2016. Order-embeddings of images and language. In *ICLR*.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhao, H.; Lu, Z.; and Poupart, P. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*, 4069–4076.

Zhao, J.; Zhu, T.; and Lan, M. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *SemEval@ COLING*, 271–277.