Multiple Positional Self-Attention Network for Text Classification

Biyun Dai, Jinlong Li, Ruoyi Xu
School of Data Science
University of Science and Technology of China
Hefei, Anhui, China
byd@mail.ustc.edu.cn jlli@ustc.edu.cn xuruoyi@mail.ustc.edu.cn

Abstract

Self-attention mechanisms have recently caused many concerns on Natural Language Processing (NLP) tasks. Relative positional information is important to self-attention mechanisms. We propose *Faraway Mask* focusing on the $(2m+1)$-gram words and *Scaled-Distance Mask* putting the logarithmic distance punishment to avoid and weaken the self-attention of distant words respectively. To exploit different masks, we present *Positional Self-Attention Layer* for generating different *Masked-Self-Attentions* and a following *Position-Fusion Layer* in which fused positional information multiplies the *Masked-Self-Attentions* for generating sentence embeddings. To evaluate our sentence embeddings approach Multiple Positional Self-Attention Network (MPSAN), we perform the comparison experiments on sentiment analysis, semantic relatedness and sentence classification tasks. The result shows that our MPSAN outperforms state-of-the-art methods on five datasets and the test accuracy is improved by 0.81%, 0.6% on SST, CR datasets, respectively. In addition, we reduce training parameters and improve the time efficiency of MPSAN by lowering the dimension number of self-attention and simplifying fusion mechanism.

Introduction

Context representation provides critical information in Natural Language Processing (NLP) tasks. For different tasks and data, there are two commonly used architectures: Convolutional Neural Networks (CNN) (Kim 2014) and Recurrent Neural Networks (RNN) (Chung et al. 2014). RNN captures long-range dependencies through sequential architecture while CNN captures phrase features by encoding n-grams. More recently, attention mechanisms have been widely used in NLP tasks, such as neural machine translation (Luong, Pham, and Manning 2015) and sentiment analysis (Kokkinos and Potamianos 2017) etc. In contrast to RNN and CNN, attention mechanisms (Bahdanau, Cho, and Bengio 2015) allow modeling of dependencies without regard to the distances of the words in the input or output sequences (Kim et al. 2017).

As a variant of attention mechanism, self-attention has been used successfully in a variety of tasks including reading comprehension (Cheng, Dong, and Lapata 2016) and abstractive summarization (Parikh et al. 2016) etc. Recently, self-attention was used to replace RNN in context-aware encoding for learning long-range dependencies (Vaswani et al. 2017). The ability to learn long-range dependencies is affected by the length of the paths along which the forward and backward signals propagate in the network. The shorter path between tokens is, the easier it is to learn long-range dependency (Hochreiter et al. 2001), and self-attention works well in learning long-range dependencies (Vaswani et al. 2017).

However, there are two issues of self-attention mechanism to be addressed. One is that contrast to RNN, self-attention treats all input tokens as identical individuals and loses the temporal order information (Shen et al. 2018a), which might be important to the NLP tasks. Another is that the positional information is lost, i.e. self-attention does not take into account different importance to the tokens of different distances. For instance, adjacent words contribute more semantically to current phrase than distant ones (Guo, Zhang, and Liu 2019).

To address the first issue, there is a way of adding directional masks to furnish temporal order information (Shen et al. 2018a), but it didn't consider positional information. About the second issue, we roughly classify several methods into two categories: adaptive positional information method (Gehring et al. 2017) and pre-tuned positional information method (Guo, Zhang, and Liu 2019). Compared to adaptive positional information method, the pre-tuned method can't extract different positional information for different sentences. While compared to pre-tuned method, adaptive method lacks scalability. And all these pre-tuned methods and adaptive methods only explore positional information and overlook the temporal order information.

Therefore, we combined the adaptive and the pre-tuned positional information methods, temporal order information and positional information into a single model to benefit each other and to overcoming their inherent disadvantages. In our MPSAN, we designed two new masks that focus on distance information. And to integrate different masks containing different temporal order and positional information, we proposed a new *Position-Fusion Layer* with 3.24million fewer parameters than the *Fusion Gate* in DISAN (Shen et al. 2018a).

In general, our contributions are four-fold:

- We proposed two new positional masks: *Faraway Mask* which focuses only on the $(2m+1)$-gram words and ig-

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

nores the influence of the word itself, and *Scaled-Distance Mask* which puts the logarithmic distance punishment. The two masks avoid and weaken the self-attention of distant words, respectively.

- We presented the *Position-Fusion Layer* which generates a weighted sum of *Masked-Self-Attentions* and *Tokens-Position-Weight*. This layer could balance the impact among different masks, and would make the combination of the masks more suitable for the current sentence adaptively.
- We reduced training parameters by lowering the dimension number of self-attention and designing a novel fusion mechanism with *softmax* function in *Positional Self-Attention Layer* and *Position-Fusion Layer* respectively, and achieved higher test accuracy at the same time.
- We evaluated our MPSAN on sentiment analysis, semantic relatedness and sentence classification tasks. The results show that our MPSAN achieves better performances than the state-of-the-art self-attention based models on SST, CR datasets, with the test accuracies improved by 0.81%, 0.6%, respectively.

Related Work

As mentioned before, self-attention loses temporal order information and positional information, which might be crucial and has attracted many researchers to study.

To encode temporal order information into attention-based models, Shen *et al.* proposed directional mask in Directional Self-Attention Network (DiSAN) (Shen et al. 2018a). In DiSAN, by adding the forward and backward directional masks to the logit of self-attention, words in a specific direction in the sentence were masked to avoid self-attention. In forward mask, there was only attention of later token j to early token i, and vice versa in backward mask, which achieved the injection of temporal order information. Recently, Shen *et al.* has developed the Reinforced Self-Attention Network (ReSAN) (Shen et al. 2018b), in which reinforcement learning was used to select tokens from two copies (head tokens and dependent tokens) of the input sequence in parallel. Furthermore, ReSAN used self-attention to model the sparse dependencies between the head and dependent tokens. In a nutshell, the reinforcement learning in ReSAN is employed to catch the temporal order information between tokens. However, the two models above all didn't consider positional information.

To inject positional information, there has been several ways, which can be roughly classified into two categories: adaptive positional information method and pre-tuned positional information method. The adaptive positional information method represents positional information by some trainable variables. For example, Gehring *et al.* proposed an entirely convolutional and attentional architecture (Gehring et al. 2017), and to capture positional information, they designed the position embeddings represented by a trainable embedding matrix. The empirical result shows that the position embeddings benefit their architecture. And Shaw *et al.* also proposed relative position encoding by adding two trainable vectors to self-attention to model localness (Shaw, Uszkoreit, and Vaswani 2018).

However, there are two possible defects with adaptive positional information method. One is obviously the loss of order information (Shen et al. 2018a), which also applies in the following pre-tuned method. In Gehring's model, adding position embeddings only plays a role in distinguishing different sequence positions. Another is the lack of scalability (Vaswani et al. 2017), which will occur when the well-trained models deal with longer sentences that have never been processed. As a result, the words at the end of the sentence will be randomly assigned positional information.

By contrast, pre-tuned positional information method represents positional information by a variable which could not be altered in training progress. Based on Gehring's adaptive position embeddings, Vaswani *et al.* used the *sine* and *cosine* functions to generate position embeddings (Vaswani et al. 2017). And motivated by DISAN, Im *et al.* proposed *Distance Mask* (Im and Cho 2017), which considers the words' relative distance information through adding distance-based punishment. The empirical result showed that *Distance Mask* achieves a better performance on long sentences. More recently, Yang *et al.* and Guo *et al.* introduced a Gaussian prior to self-attention, in which adjacent words contributed more semantically to central words (Yang et al. 2018) (Guo, Zhang, and Liu 2019). The former is effective on NMT tasks, while the latter achieves new SOTA performance on NLI tasks. Yang *et al.* also proposed a novel Convolutional Self-Attention Network (CSAN) to capture neighboring dependencies (Yang et al. 2019).

Although we have seen several ways from above to fetch positional information and temporal order information from sentences, how to integrate different positional information is also very important to sentence representation. To realize information fusion, Shen *et al.* (Shen et al. 2018a) and Im *et al.* (Im and Cho 2017) all designed a *Fusion Gate*, where they used *sigmoid* function to combine the *Masked-Self-Attention* output and the hidden state. And then the concatenation of Backward and Forward self-attentions is used as input to the *Multi-Dimensional Attention*, which is 3-layer fully connected structure with unchanged size of hidden nodes and constant size of output nodes. Apparently, this way brings about too many parameters and can be optimized.

Multiple Positional Self-Attention Network

Overall Architecture

An overview of MPSAN is illustrated in Figure 1. Given an input sentence, we first use pre-trained word embeddings to obtain the embedding representation. Then the embedding representation is utilized by the *Positional Self-Attention Layer* to obtain multiple positional representations for one sentence through applying different masks on self-attention including *Faraway Mask*, *Backward Mask*, *Forward Mask*, *Distance Mask* and *Scaled-Distance Mask*. The multiple positional representations are passed to the *Position-Fusion Layer*, and their weighted sum is treated as the sentence embeddings. Finally, a multi-dimensional attention is employed

before a softmax classifier.

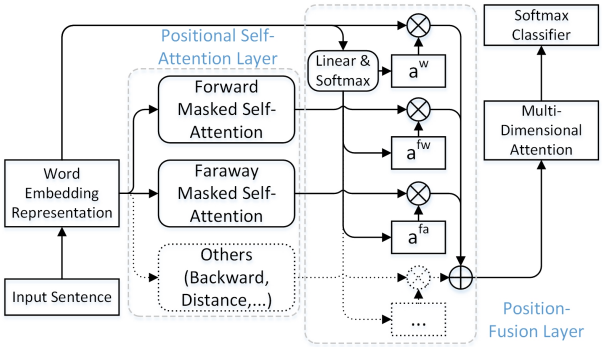

Figure 1: Structure of the MPSAN

Positional Self-Attention Layer

Positional Self-Attention Layer is composed by multiple *Masked Positional Self-Attention* units, which is illustrated in Figure 2. Before computing self-attention, we apply the linear function with an activation to calculate the sentence dense representation $\boldsymbol{h}$, defined as formula (1):

$$\boldsymbol{h} = [h_1, h_2, \cdots, h_n] = \sigma(W_h^T \boldsymbol{w} + b_h) \tag{1}$$

where $W_h \in \mathbb{R}^{d_e \times d_h}$ and $b_h \in \mathbb{R}^{d_h}$ are trainable variables, T is the transpose operation, d_e, d_h, and n are the size of word embeddings, the size of hidden nodes and the number of words, respectively. $\boldsymbol{w} = [w_1, w_2, \cdots, w_n]$ denotes the dense representation of the input sentence and $\sigma(\cdot)$ indicates activation function.

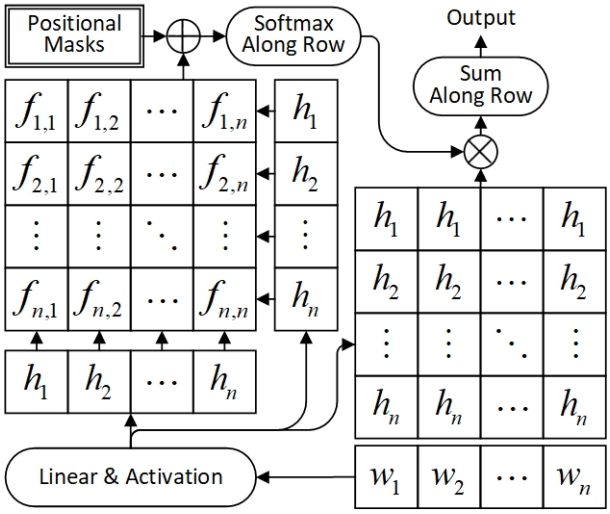

Figure 2: Masked Positional Self-Attention

Self-Attention Given the embedding representation of a source sequence $\boldsymbol{h} = [h_1, h_2, \cdots, h_n]$, a score between h_i and h_j is computed by a compatibility function $f(h_i, h_j)$, which measures the dependency between h_i and h_j, or the self-attention of h_j to h_i. Then the scores $[f(h_i, h_j)]_{i=1}^n$ are mapped to the probability distribution $p(z|h_j)$ for a given h_j by a softmax function. Here $p(z = i|h_j)$ indicates the importance of h_i to h_j where $z = i$. In a word, large $p(z = i|h_j)$ means h_i affects more on h_j, which makes it possible to extract the relationship between word pairs without the limitation of distance. The above process can be summarized by the following formulas (2), (3), (4).

$$p(z|h_j) = softmax([f(h_i, h_j)]_{i=1}^n) \tag{2}$$

$$s_j = \sum_{i=1}^{n} p(z = i|h_j) h_i = \mathbb{E}_{i \sim p(z|h_j)}(h_i) \tag{3}$$

$$\text{Self-Attention} = [s_1, s_2, \cdots, s_n] \tag{4}$$

The formulas (2), (3), (4) can be summarized as (5):

$$\text{Self-Attention}(\boldsymbol{h}) = [\sum_{i=1}^{n} \frac{\exp(f(h_i, h_j))h_i}{\sum_{k=1}^{n} \exp(f(h_k, h_j))}]_{j=1}^n \tag{5}$$

There are several types of self-attention, and they share the same form of self-attention, but are different in the compatibility function $f(h_i, h_j)$. In our MPSAN, the compatibility function is defined by formula (6):

$$f(h_i, h_j) = \sigma((Uh_i + Vh_j + b)/c) \tag{6}$$

where $U, V \in \mathbb{R}^{d_h}$, $b \in \mathbb{R}$ are trainable parameters, c is a hyper-parameter and it can be $||h_i|| \cdot ||h_j||$ or a constant. In our model, we set $c = 5$ to obtain stable output.

Positional Masks The positional masks are applied to discard some elements in specific locations of self-attention by adding $-\infty$ to the compatibility functions.

Then the *Masked-Self-Attention* (MSA) can be written as formula (7):

$$\boldsymbol{s} = \text{MSA}(\boldsymbol{h}) = [\sum_{i=1}^{n} \frac{\exp(f(h_i, h_j) + M_{ij})h_i}{\sum_{k=1}^{n} \exp(f(h_k, h_j) + M_{kj})}]_{j=1}^n \tag{7}$$

If $M_{ij} = -\infty$, we have $p(z = i|h_j) = 0$, so the self-attention of h_i to h_j is not considered. By doing that, we can focus on the special pairs of words which we are interested in.

Here are five types of the positional mask we considered in our MPSAN.

Faraway Mask The *Faraway Mask* $M_{ij}^{fa(m)}$, shown as formula (8), focuses on the $(2m+1)$-gram words, and ignores the influence of the word itself.

$$M_{ij}^{fa(m)} = \begin{cases} 0 & , \quad |i-j| \leq m, i \neq j \\ -\infty & , \quad otherwise \end{cases} \tag{8}$$

Backward Mask The *Backward Mask* M_{ij}^{bw}, shown as formula (9), prevents words that appear after the target word.

$$M_{ij}^{bw} = \begin{cases} 0 & , \quad i < j \\ -\infty & , \quad otherwise \end{cases} \tag{9}$$

Forward Mask The *Forward Mask* M_{ij}^{fw}, shown as formula (10), is the same with the *Backward Mask* except different direction.

$$M_{ij}^{fw} = \begin{cases} 0 & , \quad i > j \\ -\infty & , \quad otherwise \end{cases} \tag{10}$$

Distance Mask The *Distance Mask* M^d_{ij}, shown as formula (11), adds the distance punishment to the self-attention. The punishment score between the i_{th} word and the j_{th} word is defined by $-|i-j|$.

$$M^d_{ij} = \begin{cases} -|i-j| & , \quad i \neq j \\ 0 & , \quad otherwise \end{cases} \tag{11}$$

Scaled-Distance Mask In contrast to *Distance Mask*, the *Scaled-Distance Mask* adopts $-\log|i-j|$ for punishment score between the i_{th} word and the j_{th} word, shown as formula (12). This logarithmic distance is steep when i and j are close and is smooth when i and j are distant. By adding *Scaled-Distance Mask* to the compatibility function, the farther the word pair is, the smaller the self-attention. It means, as the distance from the target word increases, the influence of nearby words has a significant decline, and the influence of distant words is almost the same.

$$M^{sd}_{ij} = \begin{cases} -\log|i-j| & , \quad i \neq j \\ 0 & , \quad otherwise \end{cases} \tag{12}$$

Because of the *Distance Mask* and *Scaled-Distance Mask* are the penalties of self-attentions, we will not use them independently, instead, we will apply multiple masks through adding M_{ij} to $f(h_i, h_j)$. For example, $\boldsymbol{s}^{fa(m)+sd}$ indicates using $(2m+1)$-gram *Faraway Mask* and *Scaled-Distance Mask* on self-attention in the same time.

Our MPSAN applies two *Faraway Masks* $M^{fa(2)}$, $M^{fa(3)}$, *Backward Mask* M^{bw} and *Forward Mask* M^{fw}. In addition, *Scaled-Distance Mask* M^{sd} is added in *Backward/Forward Mask*.

Position-Fusion Layer

Since different positional information has been extracted in *Positional Self-Attention Layer*, the fusion of different positional information could be important to sentence representation. We supposed that every word in a sentence has a certain positional information and the representation $\boldsymbol{w}$ of original input sentence contains the most abundant positional information. So here we use *softmax* function to extract the positional information from original input tokens, which we called *Tokens-Position-Weight*, shown as formula (13):

$$\boldsymbol{a} = softmax(W_F^T \boldsymbol{w} + b_F) \tag{13}$$

where $\boldsymbol{w} \in \mathbb{R}^{d_e \times n}$ is the sentence representation composed of word embeddings, $W_F \in \mathbb{R}^{d_e \times k}$ and $b_F \in \mathbb{R}^{k \times n}$ are the weight and bias of the *Position-Fusion Layer*, k is the number of multiple self-attentions. The *softmax* function is applied on each column of $W_F^T \boldsymbol{w} + b_F$.

The output of *Position-Fusion Layer* is a weighted sum of *Masked-Self-Attention*s and *Tokens-Position-Weight*, which is represented by formula (14).

$$\boldsymbol{o} = [\boldsymbol{s}^{(1)}; \boldsymbol{s}^{(2)}; \cdots; \boldsymbol{s}^{(k-1)}; \boldsymbol{w}] * \boldsymbol{a} \tag{14}$$

where $\boldsymbol{s}^{(1)}; \boldsymbol{s}^{(2)}; \cdots; \boldsymbol{s}^{(k-1)} \in \mathbb{R}^{d_h \times n}$ are the multiple *Masked-Self-Attention*s from *Positional Self-Attention Layer*, $\boldsymbol{w} \in \mathbb{R}^{d_e \times n}$ is the sentence representation composed of word embeddings, and $\boldsymbol{a}$ is the *Tokens-Position-Weight* in (13). In practice, we set $d_e = d_h$ to make $\boldsymbol{s}^{(1)}; \cdots; \boldsymbol{s}^{(k-1)}$ and $\boldsymbol{w}$ can be weighted together, and the output $\boldsymbol{o} \in \mathbb{R}^{d_h \times n}$.

We supposed the self-attention results have multiple information in different dimensions, so we use d_h groups of (13). Considering we have decided the combination of different masks in subsection 'Positional Self-Attention Layer', we have $\boldsymbol{a} \in \mathbb{R}^{5d_h \times n}$. The rewriting of formula (13), (14) are as follows:

$$\boldsymbol{a} = \text{softmax}(W_P^T \boldsymbol{w} + b_P) \tag{15}$$

$$\boldsymbol{o} = [\boldsymbol{s}^{fa(2)}; \boldsymbol{s}^{fa(3)}; \boldsymbol{s}^{bw+sd}; \boldsymbol{s}^{fw+sd}; \boldsymbol{w}] * \boldsymbol{a} \tag{16}$$

where $\boldsymbol{w} \in \mathbb{R}^{d_e \times n}$ is the sentence representation composed of word embeddings, $W_P \in \mathbb{R}^{d_e \times 5d_h}$ and $b_F \in \mathbb{R}^{5d_h \times n}$ are the weight and bias of the *Position-Fusion Layer*, and $\boldsymbol{a}$ is in $\mathbb{R}^{5d_h \times n}$. $\boldsymbol{s}^{fa(2)}, \boldsymbol{s}^{fa(3)}, \boldsymbol{s}^{bw+sd}, \boldsymbol{s}^{fw+sd} \in \mathbb{R}^{d_h \times n}$ are four results of *Masked-Self-Attention*s. Finally, $\boldsymbol{o} \in \mathbb{R}^{d_h \times n}$ is the weighted sum.

Model	$\|\boldsymbol{\theta}\|$	**T(s)**	**Test Acc(%)**
RNN[a]	-	-	43.2
MV-RNN[a]	-	-	44.4
RNTN[a]	-	-	45.7
Bi-LSTM[b]	-	-	49.8
Tree-LSTM[c]	-	-	51.0
CNN-non-static[d]	-	-	48.0
CNN-Tensor[e]	-	-	51.2
BLSL[f]	-	-	51.1
LR-Bi-LSTM[g]	-	-	50.6
Self-Attention Based Models			
Multi-head-SAN[h]	1.5m	432	49.41
Bi-LSTM-SAN[h]	2.2m	1704	49.95
DiSAN[h]	1.8m	485	51.72
Bi-BloSAN[i]	2.2m	425	51.49
Our MPSAN	1.1m	359	**52.53**

Table 1: SST results. $|\boldsymbol{\theta}|$: the number of parameters (million level, not including word embedding part). **T(s)**: average seconds per epoch. **Test Acc(%)**: the accuracy on test set. "-" indicates that the data is not mentioned in the corresponding paper. The test accuracies of DiSAN, Bi-BloSAN and our MPSAN are the best results of ten runs, while other accuracies are extracted from the corresponding papers. The number of parameters and the average seconds per epoch of self-attention based models are obtained by the single run of their source codes. [a] (Socher et al. 2013), [b] (Li et al. 2015), [c] (Tai, Socher, and Manning 2015), [d] (Kim 2014), [e] (Lei, Barzilay, and Jaakkola 2015), [f] (Teng, Vo, and Zhang 2016), [g] (Qian et al. 2016), [h] (Shen et al. 2018a), [i] (Shen et al. 2018c).

Experiments and Results

We evaluate our model on six datasets including one sentiment analysis task and four sentence classification tasks and one semantic relatedness task.

For all tasks except semantic relatedness task, we use the pre-trained vectors GloVe-6B-300D (Pennington, Socher, and Manning 2014) to initialize word embeddings in our MPSAN. As for Out-Of-Vocabulary (OOV) words in training set, we use uniform distribution between $(-0.05, 0.05)$ to initialize those words randomly. We set the hidden units number to 300, which is equal to the word embedding size. All weight matrices in our model are initialized by Xavier (Glorot and Bengio 2010) initialization and all biases are zero-initialized. We add dropouts between the layers of our model, and the dropout-ratio is 0.7. All the activation functions $\sigma(\cdot)$ are Exponential Linear Unit (ELU) if they are not specified. We use the sum of cross-entropy loss and L2 regularization penalty as our loss function, and the L2 regularization decay factor is 10^{-7}. As for learning method, we use Adadelta, which is an optimizer of stochastic gradient descent, to minimize the loss function. The batch size of training is 64 and the learning rate is 0.5. All the training progress is completed on a single NVidia GTX-1080Ti GPU card with TensorFlow-1.4.0.

While in the semantic relatedness task, the loss function is the sum of KL-divergence and L2 regularization penalty. The dropout is set to 0.55, the L2 regularization weight decay factor is $5 * 10^{-4}$, and the learning rate is set to 0.7. The word embeddings and other network parameters are the same as other tasks.

Sentiment Analysis

The aim of sentiment analysis is to identify the emotion of a sentence. We use Stanford Sentiment Treebank (SST) for sentiment analysis (Socher et al. 2013). SST consists of 8544/1101/2210 (train/dev/test) sentences with five fine-grained labels including very positive, positive, neutral, negative and very negative.

Compared the test accuracy from the official leaderboard of SST in Table 1, MPSAN improves the best test accuracy of attention based models (achieved by DiSAN) by a significant gap of 0.81%. Moreover, we compare MPSAN with RNN-based models, including RNN, MV-RNN, RNTN, Bi-LSTM and Tree-LSTM, and MPSAN outperforms them by 9.33%, 8.13%, 6.83%, 2.73%, 1.53%, respectively. Furthermore, MPSAN achieves better test accuracies than CNN-based models like CNN-non-static and CNN-Tensor by 4.53% and 1.33%.

Mask Performance Analysis To evaluate the performance of masks, we make a comparison between different masks in Table 2. Except MPSAN, the seven models in Table 2 use different combinations of various masks.

BW&FW: Compared with Model 0, Model 1 has BW and FW, having 2.35% improvement. In addition, based on Model 3, Model 4 adds BW and FW, and have 0.86% improvement. So we come to the conclusion that employing BW and FW can improve the performance of the model.

FA: Compared with Model 0, Model 3 has two FAs (*Faraway Mask* with $m = 2$ and $m = 3$, recorded as $FA(2)$ and $FA(3)$, respectively), having 1.94% improvement. In other models, Model 4's test accuracy is 0.45% higher than Model 1's, and our MPSAN's test accuracy is 0.72% higher than Model 2's. The above comparisons indicate that FA can improve the performance of the model. We don't use $FA(1)$ because of the performance of $FA(1)$ is worse than $FA(2)$ and $FA(3)$. And we find using $FA(2)$ and $FA(3)$ at the same time is better than using them separately. Maybe we can find a better mask combination, but we don't study this further in this paper.

Model	BW	FW	FA	Test Acc(%)
0	-	-	-	49.19
1	$\surd$	$\surd$	-	51.54
2	$\surd$+SD	$\surd$+SD	-	51.81
3	-	-	2,3	51.13
4	$\surd$	$\surd$	2,3	51.99
5	$\surd$+SD	$\surd$+SD	2+SD,3+SD	52.48
6	$\surd$+D	$\surd$+D	2,3	52.17
MPSAN	$\surd$+SD	$\surd$+SD	2,3	**52.53**

Table 2: Mask study. **BW** (*Backward Mask*), **FW** (*Forward Mask*), **FA** (*Faraway Mask*), **D** (*Distance Mask*), **SD** (*Scaled-Distance Mask*) represent different masks mentioned in subsection 'Positional Self-Attention Layer'. Under the column **FA**, "2" means use $M^{fa(2)}$, and "2,3" means use $M^{fa(2)}$ and $M^{fa(3)}$ at the same time, but in different *Masked Positional Self-Attention* (Figure 2) modules. "-" indicates that the corresponding mask is not used in the model while "$\surd$" indicates the mask is used. "+D" and "+SD" means adding *Distance Mask* or *Scaled-Distance Mask* on the corresponding mask, respectively. Model 1$\sim$6 have *Positional Self-Attention Layer* and *Position-Fusion Layer*, but Model 0 don't. For each model, the accuracy is the best result of ten runs.

D/SD: Compared with Model 4, Model 6 adds *Distance Mask* on BW and FW having 0.18% improvement and our MPSAN adds Scaled-*Distance Mask* on BW and FW having 0.54% improvement. So adding distance penalty on BW and FW can improve the performance of the model and adding *Scaled-Distance Mask* is slightly better than adding *Distance Mask* on BW and FW. Compared with our MPSAN, we also add *Scaled-Distance Mask* on two FAs in Model 5, but the test accuracy of the model slightly reduced. This may be because the FA extracts local information, while the BW and FW extracts long-range positional information, and the distance penalty is more suitable for long-range positional information. In addition, compared with *Distance Mask*, *Scaled-Distance Mask* is more suitable for the extraction of long-range positional information. So adding *Scaled-Distance Mask* on BW and FW can improve the performance of the model.

In summary, various masks have different capabilities, and a good combination is better than using them separately.

Techniques for Reducing Parameters Although the combination of different masks can achieve better results, the number of training parameters will increase as the number of masks increases. Excessive parameters can lead to training costs increasing, and even cause memory overflow that make the model untrainable. We propose two

techniques in *Position-Fusion Layer* and *Positional Self-Attention Layer* respectively to reduce the parameters of the model.

Parameter Reduction for Fusion Our *Tokens-Position-Weight* is a natural extension of the *Fusion Gate* in DiSAN. The parameters we consider here include the fusion part and *Multi-Dimensional Attention* (*MDA*) part.

In *Fusion Gate*, the *Masked-Self-Attention* output $\boldsymbol{s}$ in formula (7) and the input $\boldsymbol{h}$ are combined together through *sigmoid* function, and the number of parameters is $2 \times 2d_h^2$, where d_h is the size of hidden nodes. Then the output of self-attention layer is the concatenation of *Backward* and *Forward* self-attentions, so the size of output vector nodes is $2d_h$. After that, the *MDA* is applied, which is 3-layer fully connected structure with unchanged size of hidden nodes and constant size of output nodes, so the number of parameters is $2 \times (2d_h)^2 + (2d_h) \times d_h$. The total number of parameters is $2 \times 2d_h^2 + 2 \times (2d_h)^2 + (2d_h) \times d_h = 14d_h^2$.

In contrast, as for our *Tokens-Position-Weight*, the *Forward Masked-Self-Attention* output $\boldsymbol{s^{fw}}$, the *Backward Masked-Self-Attention* output $\boldsymbol{s^{bw}}$ and the sentence representation $\boldsymbol{w}$ can be combined together through *softmax* function, so the size of output vector nodes is d_h, half of *Fusion Gate*. The total number of parameters is $3 \times d_h^2 + 2 \times d_h^2 + d_h \times d_h = 6d_h^2$.

If now we have k *Masked-Self-Attention*s, then the number of parameters in *Fusion Gate* and *MDA* are $k \times 2d_h^2 + 2 \times (kd_h)^2 + (kd_h) \times d_h = (2k^2 + 3k)d_h^2$, while the number of parameters in *Tokens-Position-Weight* and *MDA* are $(k+1) \times d_h^2 + 2 \times d_h^2 + d_h \times d_h = (k+4)d_h^2$. Here we don't consider *bias* weights because they can't affect the quantity level of the parameters. In our MPSAN, we set $k = 4$ and $d_h = 300$, so the parameter number of *Fusion Gate* and *MDA* is $(2k^2 + 2k - 4)d_h^2 = 3.24$million more than the parameter number of *Tokens-Position-Weight* and *MDA*.

We evaluated the performance of the models on SST dataset in Table 3.

Model	$\lvert\boldsymbol{\theta}\rvert$	**T(s)**	**Test Acc(%)**
B&F + FG	1.45m	298	51.36
B&F + TPW	0.73m	276	51.76
mul + *FG*	4.33m	497	52.39
mul + *TPW*	1.09m	359	52.53

Table 3: A study on the fusion way. *B&F* indicates using *Backward* and *Forward Masked-Self-Attention* introduced in subsection 'Positional Self-Attention Layer' while "mul" indicates using the masks combination introduced in *Mask Performance Analysis*. *FG* indicates *Fusion Gate*, while *TPW* indicates *Tokens-Position-Weight*. "mul + *TPW*" is our MPSAN. The test accuracies are the best results of ten runs. The number of parameters $\lvert\boldsymbol{\theta}\rvert$ and the average seconds per epoch **T(s)** are obtained by the single run.

Compared to *Fusion Gate* methods, our *Tokens-Position-Weight* methods have a significant reduction in model parameters and a slightly decrease in time cost. In the meantime, our *Tokens-Position-Weight* can also bring some improvement on the test accuracy of the model.

Parameter Reduction in Compatibility Function We have $f(h_i, h_j) \in \mathbb{R}$ in formula (5), and we call it *one-dimensional self-attention*, recorded as *1-D*. When $U, V \in \mathbb{R}^{d_h \times d_h}, b \in \mathbb{R}^{d_h}$, we have $f(h_i, h_j) \in \mathbb{R}^{d_h}$, which is *multi-dimensional self-attention*, and we record it as *M-D*. We explored the effect of using *1-D* or *M-D* in Table 4.

Model	$\lvert\boldsymbol{\theta}\rvert$	**T(s)**	**Test Acc(%)**
M-D + B&F + TPW	1.09m	466	51.95
1-D + B&F + TPW	0.73m	276	51.76
M-D + mul + *TPW*	1.81m	953	52.67
1-D + mul + *TPW*	1.09m	359	52.53

Table 4: A study on the weights of the compatibility function. "*1-D* + mul + *TPW*" is our MPSAN. The test accuracies are the best results of ten runs. The number of parameters and the average seconds per epoch are obtained by the single run.

From Table 4, we can conclude that using *1-D* instead of *M-D* not only reduces the number of parameters, but also makes the model training faster and the performances of the models are similar. In a short word, reducing multiple positional information seems to be more effective than increasing the dimension number of self-attention so we use *1-D* in our model.

Model	$\lvert\boldsymbol{\theta}\rvert$	**T(s)**	**Test Acc(%)**
MPSAN	1.09m	359	52.53
MPSAN w/o fusion	0.55m	325	51.22
MPSAN w/o position	0.64m	204	49.19
MPSAN w/o attention	0.09m	115	44.93

Table 5: An ablation study of MPSAN

Component Analysis Furthermore, to evaluate the contribution of each component, we make an ablation study of MPSAN. As shown in Table 5, we remove each component one by one and record the number of parameters, average time per epoch and test accuracy.

In this paragraph, Position Fusion Layer and Positional Self attention Layer is abbreviated as *PFL* and *PSAL* respectively. Based on MPSAN, we remove 1) *PFL*, 2) *PFL* and *PSAL*, 3) all attention modules. In terms of accuracy, the results show that 1) attention-based module do contribute to the prediction and brings 7.6% improvement; 2) using *PFL* and *PSAL* improves accuracy by 3.34%; 3) By removing *PFL* from MPSAN, the accuracy declines by 1.31%. In terms of time cost, it shows that 1) *PFL* and *PSAL* occupy more time but have better test accuracy; 2) applying the *PFL* and *PSAL* takes up negligible time and improves significant performance.

Sentence Classification

We evaluate our MPSAN on four sentence classification tasks, and the results are shown in Table 6. Text REtrieval

Model	TREC	CR	MPQA	SUBJ
CBOW (Mikolov et al. 2013)	87.3	79.9	86.4	91.3
Skip-thought (Kiros et al. 2015)	92.2	81.3	87.5	93.6
DCNN (Kalchbrenner, Grefenstette, and Blunsom 2014)	/	/	/	93.0
AdaSent (Zhao, Lu, and Poupart 2015)	91.1 (1.0)	83.6 (1.6)	**90.4 (0.7)**	92.2 (1.2)
Bi-LSTM (Graves, Jaitly, and Mohamed 2013)	94.4 (0.3)	84.6 (1.6)	90.2 (0.9)	**94.7 (0.7)**
Multi-head (Vaswani et al. 2017)	93.4 (0.4)	82.6 (1.9)	89.8 (1.2)	94.0 (0.8)
DiSAN (Shen et al. 2018a)	94.2 (0.1)	84.8 (2.0)	90.1 (0.4)	94.2 (0.6)
Bi-BloSAN (Shen et al. 2018c)	**94.8 (0.2)**	84.8 (0.9)	**90.4 (0.8)**	94.5 (0.5)
MPSAN	**94.8 (0.2)**	**85.4 (1.4)**	**90.4 (0.8)**	94.6 (0.7)

Table 6: Sentence Classification Tasks. Except our MPSAN, the reported accuracies and standard deviations are recorded by Shen *et al.* in Bi-BloSAN (Shen et al. 2018c). Except the accuracies on TREC are the mean of five runs, the accuracies on the other datasets are the mean of 10-fold cross validation. All standard deviations are in parentheses.

Conference (TREC) is a question-type classification dataset (Li and Roth 2002). Customer Reviews (CR) consist of the reviews of various products (Hu and Liu 2004). Multi-Perspective Question Answering (MPQA) dataset is used for opinion polarity detection subtask in phrase level (Wiebe, Wilson, and Cardie 2005). SUBJectivity (SUBJ) dataset is involved in classifying a sentence as being whether subjective or objective (Pang and Lee 2004). MPSAN achieves the best prediction accuracy on CR, and state-of-the-art performances on TREC and MPQA.

Semantic Relatedness

The aim of semantic relatedness is to predict the similarity degree of a pair of given sentences. We use Sentences Involving Compositional Knowledge (SICK) dataset for comparison of different methods on semantic relatedness (Marelli et al. 2014). SICK is composed of 4500/500/4927 (train/dev/test) sentence pairs and denotes the similarity degree by a real number in [1,2,3,4,5].

We give the results of SICK dataset in Table 7, whose details are shown at the front. The results show that MPSAN outperforms the previous models in terms of MSE index, which implies that although our MPSAN has slightly lower correlation, but it has better accuracy in describing experiment sentence pairs.

Model	Pearson's r	Spearman's ρ	MSE
Bi-LSTM[a]	0.8473	0.7913	0.3276
Multi-CNN[b]	0.8374	0.7793	0.3395
Hrchy-CNN[c]	0.8436	0.7874	0.3162
Multi-head[d]	0.8521	0.7942	0.3258
DISAN[e]	**0.8695**	**0.8139**	0.2879
Bi-BloSAN[f]	0.8616	0.8038	0.3008
MPSAN	0.8674	0.8119	**0.2612**

Table 7: SICK results. The reported accuracies are the mean of five runs. [a](Graves, Jaitly, and Mohamed 2013), [b](Kim 2014), [c](Gehring et al. 2017), [d](Vaswani et al. 2017), [e](Shen et al. 2018a), [f](Shen et al. 2018c)

Conclusion

In this paper, we have proposed two new positional masks (*Faraway Mask* and *Scaled-Distance Mask*) to extract word positional relationships in sentences. Moreover, combining with *Forward Mask* and *Backward Mask*, we present a *Tokens-Position-Weight* to balance the impact between different masks. On the other hand, our MPSAN achieves competitive performances on five open datasets. MPSAN is much faster and more memory efficient than the state-of-the-art self-attention based models like DiSAN, Bi-BloSAN. In the future work, we attempt to find a more general pattern to combine self-attentions and masks to avoid complexity in mask combination method. And we will verify the performance of our MPSAN on more tasks like Natural Language Inference and Reading Comprehension.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (Grant No. 2017YFC0804001) and the National Natural Science Foundation of China (Grant No. 61573328).

References

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

Cheng, J.; Dong, L.; and Lapata, M. 2016. Long short-term memory-networks for machine reading. In *EMNLP*, 551–561.

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

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1243–1252. JMLR. org.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Pro-*

ceedings 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 *ASRU*, 273–278.

Guo, M.; Zhang, Y.; and Liu, T. 2019. Gaussian transformer: a lightweight approach for natural language inference. In *The Thirty-Third AAAI Conference on Artificial Intelligence*.

Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J.; et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hu, M., and Liu, B. 2004. Mining and summarizing customer reviews. In *SIGKDD*, 168–177.

Im, J., and Cho, S. 2017. Distance-based self-attention network for natural language inference. *arXiv: Computation and Language*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *ACL*.

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*, 1746–1751.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *NIPS*, 3294–3302.

Kokkinos, F., and Potamianos, A. 2017. Structural attention neural networks for improved sentiment analysis. In *EACL*, 586–591.

Lei, T.; Barzilay, R.; and Jaakkola, T. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*, 1565–1575.

Li, X., and Roth, D. 2002. Learning question classifiers. In *ACL*.

Li, J.; Luong, M.-T.; Jurafsky, D.; and Hovy, E. 2015. When are tree structures necessary for deep learning of representations? In *EMNLP*, 2304–2314.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

Marelli, M.; Menini, S.; Baroni, M.; and Bentivogli, L. 2014. bernardi, r., zamparelli, r.: A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, 216–223.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, 271.

Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. In *EMNLP*, 2249–2255.

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

Qian, Q.; Huang, M.; Lei, J.; and Zhu, X. 2016. Linguistically regularized lstms for sentiment classification. In *ACL*, 1679–1689.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 464–468.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, 5446–5455.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Wang, S.; and Zhang, C. 2018b. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *IJCAI*, 4345–4352.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2018c. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *ICLR*.

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

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*, 1556–1566.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Wiebe, J.; Wilson, T.; and Cardie, C. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2-3):165–210.

Yang, B.; Tu, Z.; Wong, D. F.; Meng, F.; Chao, L. S.; and Zhang, T. 2018. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4449–4458.

Yang, B.; Wang, L.; Wong, D. F.; Chao, L. S.; and Tu, Z. 2019. Convolutional self-attention network. *arXiv: Computation and Language*.

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