Adaptive Learning of Local Semantic and Global Structure Representations for Text Classification

Jianyu Zhao[1,2,*], Zhiqiang Zhan[1,2,*], Qichuan Yang[3], Yang Zhang[4], Changjian Hu[4], Zhensheng Li[4], Liuxin Zhang[4], and **Zhiqiang He[4]**
[1]University of Chinese Academy of Sciences, Beijing 100049, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[3]Beihang University, Beijing 100103, China
[4]Lenovo Research, Beijing 100085, China
zhaojy7ict@gmail.com

Abstract

Representation learning is a key issue for most Natural Language Processing (NLP) tasks. Most existing representation models either learn little structure information or just rely on pre-defined structures, leading to degradation of performance and generalization capability. This paper focuses on learning both local semantic and global structure representations for text classification. In detail, we propose a novel Sandwich Neural Network (SNN) to learn semantic and structure representations automatically without relying on parsers. More importantly, semantic and structure information contribute unequally to the text representation at corpus and instance level. To solve the fusion problem, we propose two strategies: Adaptive Learning Sandwich Neural Network (AL-SNN) and Self-Attention Sandwich Neural Network (SA-SNN). The former learns the weights at corpus level, and the latter further combines attention mechanism to assign the weights at instance level. Experimental results demonstrate that our approach achieves competitive performance on several text classification tasks, including sentiment analysis, question type classification and subjectivity classification. Specifically, the accuracies are MR (82.1%), SST-5 (50.4%), TREC (96%) and SUBJ (93.9%).

1 Introduction

Representation learning plays an important role in various NLP tasks, especially in text classification (Bengio et al., 2013; Le and Mikolov, 2014). Existing representation models for text classification can be categorized into four types: *bag-of-words representation models*, *sequence representation models*, *structure representation models* and *attention-based models*. *Bag-of-words representation models* (Salton et al., 1975) are able to represent the sentence accounting for the different words, but fail to encode word order and syntactic structures. *Sequence representation models* (Kim, 2014; Kalchbrenner et al., 2014) consider word order without using structure information. *Structure representation models*, such as Tree-LSTM (Tai et al., 2015) and DSCNN (Zhang et al., 2016), take the structure information into account. *Attention-based models* (Yang et al., 2017; Lin et al., 2017) use attention mechanism to build representations by scoring input words respectively.

However, most structure representation methods either learn little structure information or rely heavily on parsers, leading to relatively low performance or gradient vanishing problem. Moreover, to our best knowledge, no one has considered effective fusion of semantic and structure information. These hinder the performance of sentence representation for text classification.

To address these issues, we propose a novel model named Sandwich Neural Network, which consists of a Convolutional Neural Network (CNN) layer in the middle of two Long Short-Term Memory (LSTM) layers like a sandwich. We define local semantic representation as n-grams feature and global structure representation as the dependency relationship between words or phrases in the sentence. SNN can generate both local semantic representation and global structure representation without relying on parsers. Then two dynamic fusion methods are developed to make full use of these information.

*They contribute equally to this paper.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2033–2043
Santa Fe, New Mexico, USA, August 20-26, 2018.

The contributions of this paper can be summarized into two parts:

1) We propose a novel SNN to extract semantic and structure representations, which enriches information of sentence representation. Thus SNN improves the performance of text classification.

2) In order to make full use of these two representations, we design and implement two fusion methods to learn the weights of semantic and structure representations at corpus and instance level respectively. The former employs adaptive learning strategy to automatically learn a pair of weights for the whole corpus. The latter integrates attention mechanism to adjust the weights subtly for each single instance.

The rest of our paper is organized as follows: Section 2 elaborates the related work about structure representation and attention mechanism. Section 3 details the proposed SNN and two fusion methods. Section 4 conducts experiments on four datasets to verify the effectiveness of our model. Section 5 draws a conclusion and discusses the future work.

2 Related Work

Text classification develops from rule-based method, statistical machine learning method to deep learning method (Aggarwal and Zhai, 2012). Many researches on text classification focus on the sentence representation (Liu et al., 2018). Structure representation and attention mechanism are important for sentence representation.

2.1 Sentence Structure Representation

Current researches about sentence representation pay much attention to structure representation. Structure representation models can be divided into three types: *hierarchical models*, *tree-based models* and *reinforcement learning (RL) models*. *Hierarchical models* combine CNN and RNN to learn structure representation, such as C-LSTM (Zhou et al., 2015) and DSCNN (Zhang et al., 2016). C-LSTM utilizes CNN to extract a sequence of higher-level phrase representations. Then the representations are fed into an LSTM to obtain sentence representation with long dependency structure. DSCNN utilizes CNN to extract features from hidden states of an LSTM layer, which is able to catch long dependency structure at a certain level. *Tree-based models*, such as Tree-LSTM (Tai et al., 2015) and TBCNN (Mou et al., 2015), rely on existing parsers. Tree-LSTM utilizes LSTM along the tree structure and treats root as sentence representation. TBCNN combines CNN with tree structure and uses pooling results as sentence representation. More recently, *RL models* are attempted in structure representation, such as ID-LSTM (Zhang et al., 2018) and HS-LSTM (Zhang et al., 2018), which are integrated seamlessly with a policy network and a classification network. These works contribute significantly to sentence structure representation.

However, *hierarchical models* are not able to learn adequate structure information and lack effective fusion of semantic and structure representations; *tree-based models* are time-consuming and rely on parsers which are unable to guarantee an accurate syntactic structure; *RL models* require much experience to initialize and design reward function, which yields moderate performance.

2.2 Attention Mechanism

Attention mechanism for NLP is first proposed by Bahdanau et al. (2014) to improve machine translation task. The neural machine translation model consists of two RNNs: an encoder and a decoder. When decoding the hidden state s_i, the decoder calculates attention distribution $\boldsymbol{\alpha}$ on the whole source sentence. The attention mechanism is defined as follows:

$$\begin{aligned} \boldsymbol{\alpha}_{ij} &= exp(\boldsymbol{e}_{ij}) / \sum_{k=1}^{T_x} exp(\boldsymbol{e}_{ik}) \\ \boldsymbol{e}_{ij} &= \sigma(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j) \end{aligned} \tag{1}$$

where x is the source sentence, T_x is the length of source sentence, σ is an activation function, $\boldsymbol{s}_{i-1}$ is the hidden state of the decoder RNN at time step $i-1$, $\boldsymbol{h}_j$ is the hidden state of the encoder RNN at time step j.

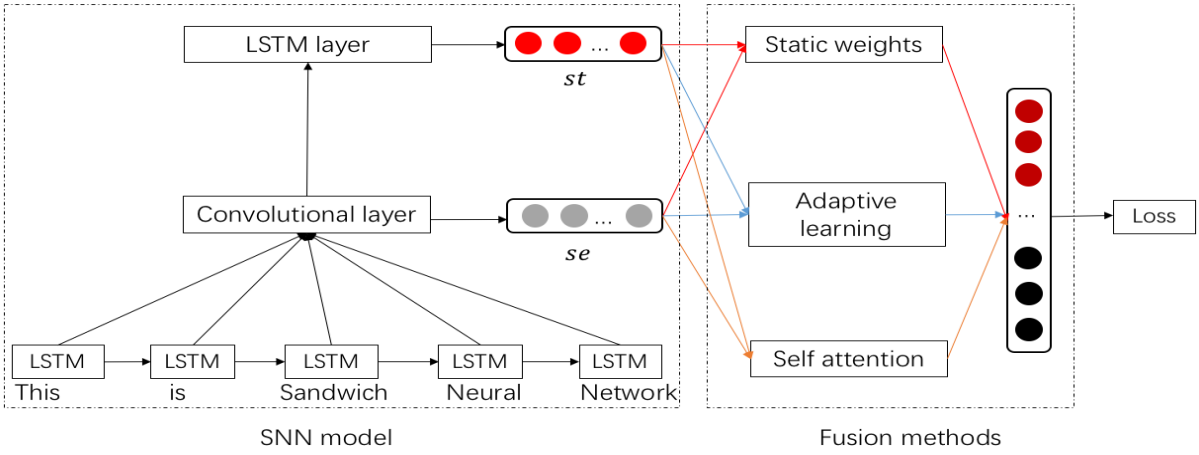


Figure 1: SNN and fusion methods.

Because of its promising performance, attention mechanism attracts lots of attention. For example, Rush et al. (2015) apply it to solve sentence summarization. Xu et al. (2015) utilize it to generate caption of images. Tan et al. (2017) employ it to handle reading comprehension task.

3 Proposed Model

In order to address problems as discussed in Section 2.1, we are inspired by Section 2.2 to propose our novel methods. As shown in Figure 1, there exists three major parts: (1) SNN model: build the semantic and structure representations through SNN; (2) Fusion methods: fuse the semantic and structure representations with different methods; (3) Loss function: train the model with cross-entropy objective function. The core of our methods can be formulated as $\boldsymbol{M} = \phi(\boldsymbol{se}, \boldsymbol{st})$, where ϕ is a fusion function which combines the semantic representation $\boldsymbol{se}$ with the structure representation $\boldsymbol{st}$.

3.1 Obtaining the semantic and structure representations

As shown in Figure 1, SNN consists of three parts: First LSTM layer, CNN layer and Second LSTM layer. The first LSTM layer utilizes the hidden states of LSTM as tuned word representations to take the context into account and the tuned representations are the input of CNN layer. The CNN layer is designed to learn local n-gram high-level semantic representations like standard CNN (Kim, 2014) through convolution and max pooling. The second LSTM layer feeds results from the convolution into LSTM and use the last hidden state of LSTM to obtain global structure representations.

First LSTM layer

Let the input of our model be a sentence of length $s : [w_1, w_2, ..., w_s]$, c be the total number of word embedding versions and $\boldsymbol{x}_i^{(j)}$ is the i^{th} word's embedding of the j^{th} version. The common word embedding versions are Word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014).

The first layer of our model consists of LSTM networks processing multiple versions of word embeddings. For each version of word embedding, we construct an LSTM network where the input $\boldsymbol{x}_t \in \mathbb{R}^d$ is the d-dimensional word embedding for w_t. The LSTM layer will produce a hidden state $\boldsymbol{h}_t \in \mathbb{R}^d$ at each time step. We collect hidden states as the output of LSTM layers:

$$\boldsymbol{h}^{(i)} = [\boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)}, ..., \boldsymbol{h}_t^{(i)}, ..., \boldsymbol{h}_s^{(i)}] \tag{2}$$

for $i = 1, 2, ..., c$.

Then these hidden states are fed into CNN layer as input.

CNN layer

To utilize multiple kinds of word embeddings, we apply a filter $\boldsymbol{F} \in \mathbb{R}^{c \times d \times l}$, where l is the window size. Hidden state sequence $\boldsymbol{h}^{(i)}$ produced by the i^{th} version of word embedding forms one channel of the feature maps. Then these feature maps are stacked c-channel feature maps $\boldsymbol{X} \in \mathbb{R}^{c \times d \times s}$. Afterwards,

Sentence	Ground truth	Reason
What is the conversion rate between dollars and pounds ?	numeric	key word "rate"
What does target heart rate mean ?	describe	key structure "what does ... mean" and word "rate" is a disturbance

Table 1: Examples of TREC classification

filter F convolves with the window vectors (l-gram) at each position to generate a feature map $\boldsymbol{c} \in \mathbb{R}^{s-l+1}$; c_k is the element of the feature map $\boldsymbol{c}$ for window vector $\boldsymbol{X}_{k:k+l-1}$ at position k and it is produced as follows:

$$c_k = f(\sum_{i,j,r}(\boldsymbol{F} \odot \boldsymbol{X}_{k:k+l-1})_{i,j,r}) \tag{3}$$

where $\odot$ denotes element-wise multiplication.

The n feature maps generated from n filters can be rearranged through column vector concatenation method to form a new representation,

$$\boldsymbol{W} = [\boldsymbol{c}_1; \boldsymbol{c}_2; ...; \boldsymbol{c}_n] \tag{4}$$

Each row $\boldsymbol{W}_j$ of $\boldsymbol{W} \in \mathbb{R}^{(s-l+1)\times n}$ is the feature map generated from n filters for the window vector at position j. The new successive higher-level representations are then fed into the second LSTM layer.

Here, a max-over-time pooling layer is added after the convolution neural network. The pooling result of the feature map $\boldsymbol{c}$ is :

$$p = max(c_1, c_2, ..., c_{s-l+1}) \tag{5}$$

These pooling results are used as our local semantic representation $\boldsymbol{se} \in \mathbb{R}^n$:

$$\boldsymbol{se} = [p_1, p_2, ..., p_n] \tag{6}$$

Second LSTM layer

This layer feeds the high-level phrase representation $\boldsymbol{W}$ generated from CNN into LSTM. We use the same number of filters n to denote the dimension in this LSTM layer for simple and fair fusion and use the last hidden state of LSTM as global structure representation $\boldsymbol{st} \in \mathbb{R}^n$.

Thus, we get the local semantic representation $\boldsymbol{se}$ and the global structure representation $\boldsymbol{st}$. Then we combine $\boldsymbol{se}$ and $\boldsymbol{st}$ to get the concatenated weighted sentence representation.

3.2 Fusion methods

To make better use of semantic and structure representations and address the fusion problems discussed in Section 1, we explore three different fusion ways to learn the weights of semantic and structure representations.

Static weights

Combine semantic and structure representations to get the sentence representation. That is to say, this method gives equal weights to these two kinds of representations.

Adaptive learning at corpus level

Different properties (i.e. semantic and structure property) of a sentence contribute unequally according to the specific corpus. Table 1 shows examples of this phenomenon. To take the above observation into account, we design an adaptive learning method to learn the semantic weight g_{se} and the structure weight g_{st} at corpus level. To reflect the difference of these two properties, we use a simple but elegant and effective method: set g_{se} to be α and g_{st} to be $1 - \alpha$. They are parameters that can be learned automatically during the training process. Each element in the semantic representation $\boldsymbol{se}$ will multiply α to get the weighted semantic representation. It is the same with structure representation $\boldsymbol{st}$ and $1 - \alpha$. Figure 2 shows the adaptive learning method.

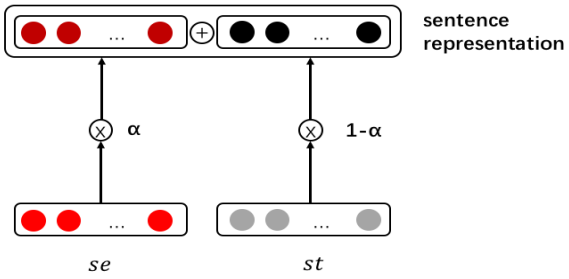


Figure 2: Adaptive learning at corpus level.

Self-attention at instance level

With the adaptive learning strategy, the weights are learned based on the whole corpus. However, the weights of semantic and structure information vary according to the sentence itself. For instance, "*The dog the stick the fire burned beat bit the cat*" is rich in structure information, and we should pay more attention to its structure. Motivated by the attention mechanism, we propose the self-attention strategy to learn weights at instance level. First, we average the word embeddings of the sentence to get a simple but useful sentence representation $\boldsymbol{S}$. Second, linear transformations are used to transform the representations into semantic space representation $\boldsymbol{S}_{se}$ and structure space representation $\boldsymbol{S}_{st}$. The transformed dimension is d. Third, we compute the similarity of representations through inner product. Last, softmax is used to normalize weights. Figure 3 shows the whole process. The computational procedure is as follows:

$$
\begin{aligned}
(att_{se}, att_{st}) &= softmax(p_{se}, p_{st}) \\
p_{se} &= \rho(\boldsymbol{S}_{se}, \boldsymbol{se}) \\
p_{st} &= \rho(\boldsymbol{S}_{st}, \boldsymbol{st}) \\
\boldsymbol{S}_{se} &= \boldsymbol{W}_{se} \times \boldsymbol{S} + \boldsymbol{b}_{se} \\
\boldsymbol{S}_{st} &= \boldsymbol{W}_{st} \times \boldsymbol{S} + \boldsymbol{b}_{st} \\
\boldsymbol{S} &= (\boldsymbol{x}_1 + \boldsymbol{x}_2 + ... + \boldsymbol{x}_s)/s
\end{aligned}
\tag{7}
$$

where $\boldsymbol{x}_i$ is the word embedding, size of d; $\boldsymbol{S}$ is a representation of the sentence, calculated by averaging the word embeddings, size of d; $\boldsymbol{S}_{se}$ and $\boldsymbol{S}_{st}$ are the transformed semantic and structure representations, size of n; $\boldsymbol{W}_{se}$ and $\boldsymbol{W}_{st}$ are the projection matrixes, size of $n \times d$; $\boldsymbol{b}_{se}$ and $\boldsymbol{b}_{st}$ are the projection biases, size of d; ρ is an average inner product operator; att_{se} and att_{st} are the attention weights.

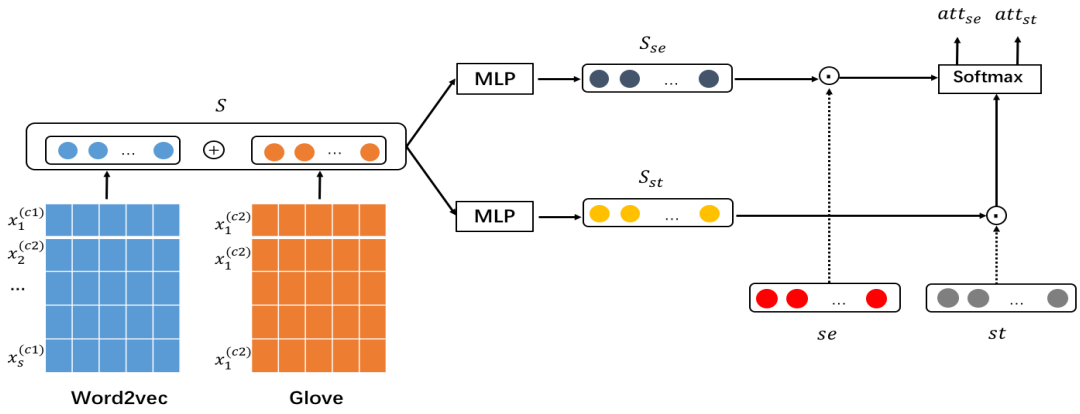


Figure 3: Self-attention at instance level.

In the above fusion methods, the adaptive learning strategy and self-attention strategy can learn weights on the two representations. Then, we concatenate weighted semantic and structure represen-

Data	Class	Len	Max Len	Size	Test
MR	2	20	56	10662	CV
SST-5	5	18	53	11855	2210
TREC	6	10	37	9592	500
SUBJ	2	23	120	10000	CV

Table 2: Statistics for experiment datasets. Class: number of classes. Len: average length of sentence. Max: max length of sentence. Size: Dataset size. Test: test set size (CV means no standard train/test split and thus 10-fold CV is used).

tations to get the final sentence representation.

$$\boldsymbol{M} = \begin{cases} [\boldsymbol{se}; \boldsymbol{st}], \text{for fixed weights} \\ [g_{se} * \boldsymbol{se}; g_{st} * \boldsymbol{st}], \text{for adaptive weights} \\ [att_{se} * \boldsymbol{se}; att_{st} * \boldsymbol{st}], \text{for self-attention} \end{cases} \tag{8}$$

where $\boldsymbol{M}$ is the sentence representation, and operator $[\boldsymbol{v1}; \boldsymbol{v2}]$ denotes concatenation of vector $\boldsymbol{v1}$ and $\boldsymbol{v2}$.

3.3 Training Models

After getting both semantic and structure representations, a fully-connected layer follows. To learn the model parameters, we minimize a cross-entropy objective function as follows:

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} y^i \times log(\hat{y}_i) \tag{9}$$

where y^i is the one-hot vector representation of real label of the i-th sentence, $\hat{y}_i$ is the predicted probability representation in the neural network, and m is the number of samples.

4 Experiments

4.1 Datasets

We test our model on several datasets. The detailed information is listed in Table 2.

• **Movie Review (MR)** proposed by Pang and Lee (2005) is a dataset for sentiment analysis of movie reviews. It contains two classes: positive and negative.

• **Stanford Sentiment Treebank with Five Labels (SST-5)** is another popular sentiment classification dataset proposed by Socher et al. (2013). The sentences are labeled in a fine-grained way: very negative, negative, neutral, positive, very positive.

• **Text REtrieval Conference (TREC)** dataset proposed by Li and Roth (2002) contains sentences that are questions in the following 6 classes: abbreviation, entity, description, location, numeric, human.

• **SUBJectivity (SUBJ)** classification dataset released by Pang and Lee (2004) contains two classes: subjective and objective.

4.2 Experimental Configuration

Literature usually adopts 100-300 as the dimension of word embedding. Here, we use two versions of word embeddings, Word2vec and Glove, with dimension 300. We use 100 convolution filters each for window sizes of 3,4,5. Rectified Linear Units (RELU) is chosen as the nonlinear function in the convolutional layer. For the second LSTM, we set the dimension to be 100. For regularization, both word-embedding and the penultimate layer use dropout (Hinton et al., 2012) with rate 0.5. We don't impose L2 regularization at each layer. We use the gradient-based optimizer Adam (Kingma and Ba, 2014) to minimize the loss between the predicted and true distributions, and the training is early stopped when the accuracy on validation set starts to drop. Additionally, we use the same configuration for all datasets.

Method	MR	SST-5	TREC	SUBJ
Standard-RNN (Socher et al., 2013)	–	43.2	–	–
Standard-LSTM (Tai et al., 2015)	77.4	46.4	–	92.2*
bi-LSTM (Tai et al., 2015)	79.7	49.1	–	92.8*
CNN (Kim, 2014)	81.5	48	93.6	93.4
DCNN (Kalchbrenner et al., 2014)	–	48.5	93	–
MVCNN (Yin and Schütze, 2016)	–	49.6	–	**93.9**
Tree-LSTM (Tai et al., 2015)	80.7	50.1	–	93.2*
TBCNN (Mou et al., 2015)	–	**51.4**	**96**	–
Dep-CNN (Ma et al., 2015)	81.9	49.5	95.4	–
DSCNN (Zhang et al., 2016)	81.5	49.7	95.4	93.2
C-LSTM (Zhou et al., 2015)	–	49.2	94.6	–
ID-LSTM (Zhang et al., 2018)	81.6	50.0	–	93.5
HS-LSTM (Zhang et al., 2018)	**82.1**	49.8	–	93.7
Self-Attentive (Lin et al., 2017)	80.1	47.2	–	92.5
SNN	81.7	49.8	95.6	93.8
AL-SNN	81.9	50	95.8	**93.9**
SA-SNN	**82.1**	50.4	**96**	**93.9**

Table 3: Experiment results of our methods compared with other models. Performance is measured in accuracy(%). The results with * is from Zhang et al. (2018). Models are categorized into 7 classes. The first block is RNN and its variants. The second is CNN and its variants. The third is tree-based model. The forth is the hierarchical model. The fifth is RL model. The sixth is attention-based model. And the last is our models. **Standard-RNN**: Standard Recursive Neural Network (Socher et al., 2013). **Standard-LSTM**: Standard Long Short-Term Memory Network (Tai et al., 2015). **bi-LSTM**: Bidirectional LSTM (Tai et al., 2015). **CNN**: Convolutional Neural Network (Kim, 2014) . **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **MVCNN**: Multichannel Variable-Size Convolution Neural Network (Yin and Schütze, 2016). **Tree-LSTM**: Tree-Structured LSTM (Tai et al., 2015). **TBCNN**: Tree-Based Convolutional Neural Network (Mou et al., 2015). **Dep-CNN**: Dependency-based Convolutional Neural Network (Ma et al., 2015). **DSCNN**: Dependency Sensitive Convolutional Neural Network (Zhang et al., 2016). **C-LSTM** :Convolutional LSTM (Zhou et al., 2015). **ID-LSTM**: Information Distilled LSTM (Zhang et al., 2018). **HS-LSTM**: Hierarchical Structured LSTM (Zhang et al., 2018). **Self-Attentive**: Structured Self-Attentive model (Lin et al., 2017).

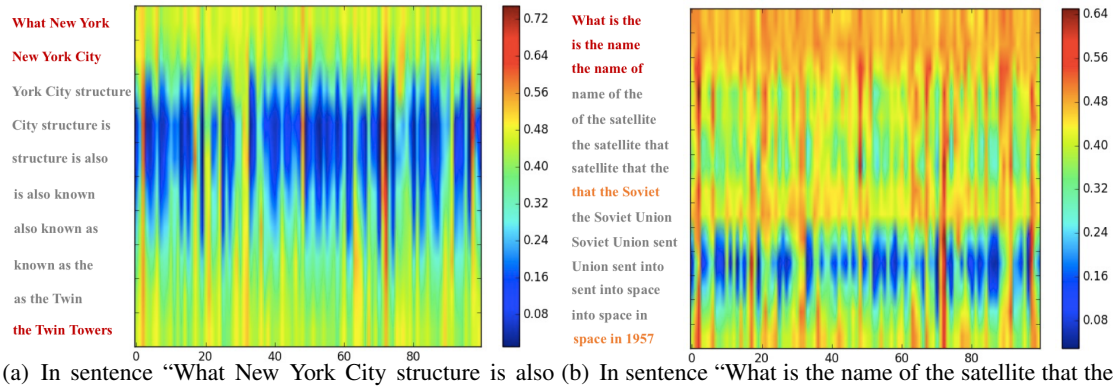
4.3 Results and Discussion

As shown in Table 3, the results demonstrate effectiveness of our model in comparison with other state-of-the-art methods. Among the models without relying on parsers, the other models get the highest accuracy in one certain task (e.g. HS-LSTM in MR and MVCNN in SUBJ), but our models get the highest accuracy on all datasets: MR, SST-5, TREC and SUBJ. It shows strong generalization capability. Additionally, compared with TBCNN, which relies heavily on parsers, our models get the same highest accuracy on TREC and are on a par with TBCNN's performance on SST-5.

The benefit of SNN is attributed to its ability to capture both semantic and structure representations. The benefit of adaptive learning is attributed to its ability to learn the weights of both representations according to the specific corpus. The benefit of self-attention is attributed to its ability to adjust the weights of both representations at instance level. In the following, we first give visualization of our models to show the learned structure. Then, we give some examples to show how our models make progress from SNN, adaptive learning to self-attention. At last, we make quantitative analysis to show the correlation between structure attention value and structure complexity.

4.3.1 Visualization of learned structure

The input gate value of LSTM is able to show how much a word representation contributes to the final sentence representation (Palangi et al., 2016). We draw a heat map of input gates of the second LSTM layer to show the learned structure. Figure 4 gives two examples from TREC. The color represents the values (from 0 to 1) of input gate. The higher means the n-gram part contributes more to the final sentence representation.



(a) In sentence "What New York City structure is also known as the Twin Towers", structure "What New York City" – "the Twin Towers" is learned. It is very related to sentence class "location".

(b) In sentence "What is the name of the satellite that the Soviet Union sent into space in 1957", structure "What is the name of" is important and related to sentence class "entity". More amazing, attributive clause structure "that" is learned. The content before "that" contributes more than the content after "that", which makes the representation focus more on what is related to the classification.

Figure 4: Visualization of learned structure.

Sentence	Dataset	Ground truth	DSCNN	SNN
What does your spleen do ?	TREC	describe	abbreviation	describe
Some actors have so much charisma that you'd be happy to listen to them reading the phone book.	SST-5	positive	negative	positive

Table 4: Examples that SNN makes right while DSCNN fails.

4.3.2 SNN analysis

SNN can learn both semantic and structure representations. As shown in Table 4: in the first sentence, word "spleen" does not appear in the train data, but its topically related words in medical domain like "AIDS", "HIV", "BPH", "SIDS" appear in abbreviation class frequently. Traditional models (e.g. DSCNN) ignore the structure "what does ... do" and misclassifies it into abbreviation class. In contrast, our model captures the structure and correctly classifies it into entity class.

4.3.3 AL-SNN analysis

AL-SNN can learn the weights of semantic and structure representations at corpus level. As shown in Table 5: g_{se} denotes semantic weight and g_{st} denotes structure weight as mentioned in Section 3. In the first sentence, the structure "what is ... the name of" appears with almost the same frequency (67 and 66 times) in human and entity class and appears very little in the other classes. Thus, more weights should be added on semantic representation. Through adaptive learning at corpus level, AL-SNN achieves it.

Sentence	Dataset	Ground truth	SNN	AL-SNN	g_{se}	g_{st}
What was the name of the plane Lindbergh flew solo across the Atlantic?	TREC	entity	human	entity	0.94	0.06
The acting is stiff, the story lacks all trace of wit, the sets look like they were borrowed from gilligan's inland, and – the cgi scooby might well be the worst special-effects creation of the year.	SST-5	very negative	negative	very negative	0.56	0.44

Table 5: Examples that AL-SNN makes right while SNN fails.

Sentence	Dataset	Ground truth	AL-SNN	SA-SNN	g_{st}	att_{st}
What is Hawaii 's state flower ?	TREC	entity	location	entity	0.06	0.33
The director byler may yet have a great movie in him, but charlotte sometimes is only half of one .	SST-5	negative	positive	negative	0.44	0.57

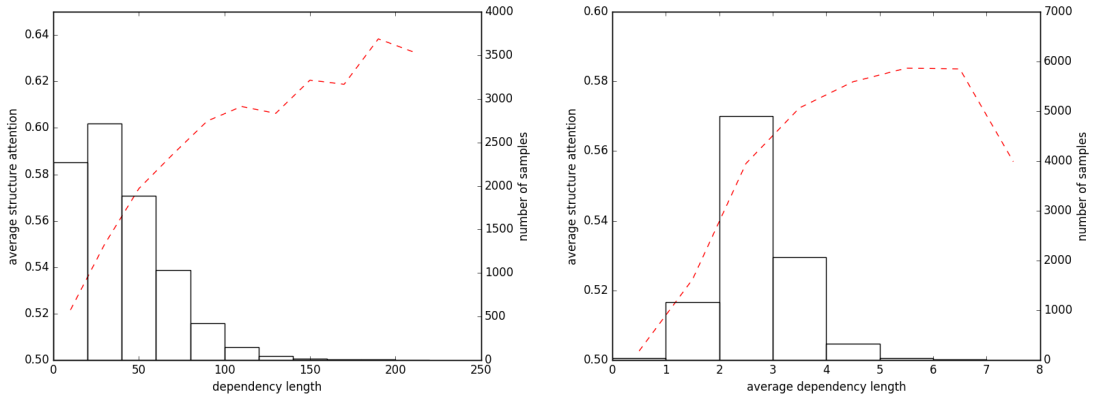
Table 6: Examples that SA-SNN makes right while AL-SNN fails.

4.3.4 SA-SNN analysis

SA-SNN can adjust weights at instance level. As shown in Table 6: g_{st} denotes structure weight and att_{st} denotes structure attention as mentioned in Section 3. In the first sentence, AL-SNN gives more weight to semantic representation at corpus level, focuses on word "Hawaii" and misclassifies the sentence into location class. Through self attention, SA-SNN gives more weight to structure, focuses on structure "what is ... ?" and correctly classifies the sentence into entity class.

4.3.5 Quantitative analysis

It is obvious that more complicated structure means higher structure weight. We use four statistics to represent structure complexity: dependency length (dl), average dependency length (adl), max dependency length (mdl) and sentence length (sl). Among them, average dependency length is dependency length divided by the number of dependency relationships of the sentence. Figure 5 depicts the correlation between the average structure attention (asa) of SA-SNN and the four statistics on SST-5. It shows a strongly positive correlation apart from a few small random variations caused by a limited sample size. Thus, it proves that our attention mechanism is very reasonable.



(a) the correlation between asa and dl

(b) the correlation between asa and adl

(c) the correlation between asa and mdl

(d) the correlation between asa and sl

Figure 5: The correlation between structure attention of SA-SNN and structure complexity on SST-5.

5 Conclusion

In this paper, we propose a novel SNN model and two effective fusion methods, i.e. AL-SNN and SA-SNN. By capturing and fusing both local semantic information and global structure information effectively, our model achieves state-of-the-art in several text classification tasks among the methods without relying on parsers, which verifies its great performance and strong generalization capability. Additionally, we give examples, visualization and quantitative analysis to make explanations in detail. In the future, we will pay more attention to knowledge-embedded methods to speed up training and improve performance.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions, which are helpful in improving the quality of the paper.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Y Bengio, A Courville, and P Vincent. 2013. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

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

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Li Liu, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikainen. 2018. A survey of recent advances in texture representation. *arXiv preprint arXiv:1801.10324*.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

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

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

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.