Star-Transformer

Qipeng Guo[†]
qpguo16@fudan.edu.cn

Xipeng Qiu[† *]
xpqiu@fudan.edu.cn

Pengfei Liu[†]
pfliu14@fudan.edu.cn

Yunfan Shao[†]
yfshao15@fudan.edu.cn

Xiangyang Xue[†]
xyxue@fudan.edu.cn

Zheng Zhang[‡]
zz@nyu.edu

[†]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[†]School of Computer Science, Fudan University
[‡]New York University

Abstract

Although *Transformer* has achieved great successes on many NLP tasks, its heavy structure with fully-connected attention connections leads to dependencies on large training data. In this paper, we present *Star-Transformer*, a lightweight alternative by careful sparsification. To reduce model complexity, we replace the fully-connected structure with a star-shaped topology, in which every two non-adjacent nodes are connected through a shared relay node. Thus, complexity is reduced from quadratic to linear, while preserving capacity to capture both local composition and long-range dependency. The experiments on four tasks (22 datasets) show that Star-Transformer achieved significant improvements against the standard Transformer for the modestly sized datasets.

1 Introduction

Recently, the fully-connected attention-based models, like Transformer (Vaswani et al., 2017), become popular in natural language processing (NLP) applications, notably machine translation (Vaswani et al., 2017) and language modeling (Radford et al., 2018). Some recent work also suggest that Transformer can be an alternative to recurrent neural networks (RNNs) and convolutional neural networks (CNNs) in many NLP tasks, such as GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-XL (Dai et al., 2019) and Universal Transformer (Dehghani et al., 2018).

More specifically, there are two limitations of the Transformer. First, the computation and mem-

---

[*] Corresponding Author.
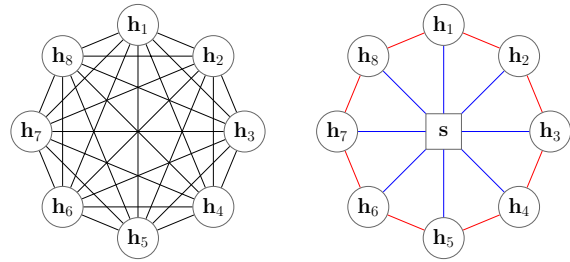[‡] work done at NYU Shanghai, now with AWS Shanghai AI Lab.

Figure 1: Left: Connections of one layer in Transformer, circle nodes indicate the hidden states of input tokens. Right: Connections of one layer in Star-Transformer, the square node is the virtual relay node. Red edges and blue edges are ring and radical connections, respectively.

ory overhead of the Transformer are quadratic to a sequence length. This is especially problematic with long sentences. Transformer-XL (Dai et al., 2019) provides a solution which achieves the acceleration and performance improvement, but it is specifically designed for the language modeling task. Second, studies indicate that pre-training on large corpus brings significant improvement for Transformer (Radford et al., 2018; Devlin et al., 2018). Transformer failed on many tasks, however, if it is not pre-trained and when training data is limited.

A key obervation is that Transformer is weakly bounded by prior knowledge. For example, the local compositionality is already a robust inductive bias for modeling the text sequence. The Transformer learns this bias from scratch, along with non-local compositionality, thereby increasing the learning cost. The key insight is then whether leveraging strong prior knowledge can help to "lighten up" the architecture.

To address the above limitation, we pro-

posed a new lightweight model named "Star-Transformer". The core idea is to sparsify the architecture by moving the fully-connected topology into a star-shaped structure. Fig-1 gives an overview. Star-Transformer has two kinds of connections. The radical connections preserve the non-local communication and remove the redundancy in fully-connected network. The ring connections embody the local-compositionality prior, which has the same role as in CNNs/RNNs. The direct outcome of our design is the improvement of both efficiency and learning cost: the computation cost is reduced from quadratic to linear as a function of input sequence length. An inherent advantage is that the ring connections can effectively reduce the burden of the unbias learning of local and non-local compositionality and improve the generalization ability of the model. What remains to be tested is whether one shared relay node is capable of capturing the long-range dependencies.

We evaluate the Star-Transformer on three NLP tasks including Text Classification, Natural Language Inference, and Sequence Labelling. Experimental results show that Star-Transformer outperforms the standard Transformer consistently and has less computation complexity. An additional analysis on a simulation task indicates that Star-Transformer preserve the ability to handle with long-range dependencies which is a crucial feature of the standard Transformer.

In this paper, we claim three contributions as the following:

- Compared to the standard Transformer, Star-Transformer has a lightweight structure but with an approximate ability to model the long-range dependencies. It reduces the number of connections from n^2 to $2n$, where n is the sequence length.

- The Star-Transformer divides the labor of semantic compositions between the radical and the ring connections. The radical connections focus on the non-local compositions and the ring connections focus on the local composition. Therefore, Star-Transformer works for modestly sized datasets and does not rely on heavy pre-training.

- We design a simulation task "Masked Summation" to probe the ability dealing with long-range dependencies. In this task,

we verify that both Transformer and Star-Transformer are good at handling long-range dependencies compared to the LSTM and BiLSTM.

2   Related Work

Recently, neural networks have proved very successful in learning text representation and have achieved state-of-the-art results in many different tasks.

Modelling Local Compositionality   A popular approach is to represent each word as a low-dimensional vector and then learn the local semantic composition functions over the given sentence structures. For example, Kim (2014); Kalchbrenner et al. (2014) used CNNs to capture the semantic representation of sentences, whereas Cho et al. (2014) used RNNs.

These methods are biased for learning local compositional functions and are hard to capture the long-term dependencies in a text sequence. In order to augment the ability to model the non-local compositionality, a class of improved methods utilizes various self-attention mechanisms to aggregate the weighted information of each word, which can be used to get sentence-level representations for classification tasks (Yang et al., 2016; Lin et al., 2017; Shen et al., 2018a). Another class of improved methods augments neural networks with a re-reading ability or global state while processing each word (Cheng et al., 2016; Zhang et al., 2018).

Modelling Non-Local Compositionality   There are two kinds of methods to model the non-local semantic compositions in a text sequence directly.

One class of models incorporate syntactic tree into the network structure for learning sentence representations (Tai et al., 2015; Zhu et al., 2015).

Another type of models learns the dependencies between words based entirely on self-attention without any recurrent or convolutional layers, such as Transformer (Vaswani et al., 2017), which has achieved state-of-the-art results on a machine translation task. The success of Transformer has raised a large body of follow-up work. Therefore, some Transformer variations are also proposed, such as GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-XL (Dai et al., 2019) , Universal Transformer (Dehghani et al., 2018) and CN³ (Liu et al., 2018a).

However, those Transformer-based methods usually require a large training corpus. When applying them on modestly sized datasets, we need the help of semi-supervised learning and unsupervised pretraining techniques (Radford et al., 2018).

Graph Neural Networks Star-Transformer is also inspired by the recent graph networks (Gilmer et al., 2017; Kipf and Welling, 2016; Battaglia et al., 2018; Liu et al., 2018b), in which the information fusion progresses via message-passing across the whole graph.

The graph structure of the Star-Transformer is star-shaped by introducing a virtual relay node. The radical and ring connections give a better balance between the local and non-local compositionality. Compared to the previous augmented models (Yang et al., 2016; Lin et al., 2017; Shen et al., 2018a; Cheng et al., 2016; Zhang et al., 2018), the implementation of Star-Transform is purely based on the attention mechanism similar to the standard Transformer, which is simpler and well suited for parallel computation.

Due to its better parallel capacity and lower complexity, the Star-Transformer is faster than RNNs or Transformer, especially on modeling long sequences.

3 Model

3.1 Architecture

The Star-Transformer consists of one relay node and n satellite nodes. The state of i-th satellite node represents the features of the i-th token in a text sequence. The relay node acts as a virtual hub to gather and scatter information from and to all the satellite nodes.

Star-Transformer has a star-shaped structure, with two kinds of connections in the: the radical connections and the ring connections.

Radical Connections For a network of n satellite nodes, there are n radical connections. Each connection links a satellite node to the shared relay node. With the radical connections, every two non-adjacent satellite nodes are two-hop neighbors and can receive non-local information with a two-step update.

Ring Connections Since text input is a sequence, we bake such prior as an inductive bias. Therefore, we connect the adjacent satellite nodes

to capture the relationship of local compositions. The first and last nodes are also connected. Thus, all these local connections constitute a ring-shaped structure. Note that the ring connections allow each satellite node to gather information from its neighbors and plays the same role to CNNs or bidirectional RNNs.

With the radical and ring connections, Star-Transformer can capture both the non-local and local compositions simultaneously. Different from the standard Transformer, we make a division of labor, where the radical connections capture non-local compositions, whereas the ring connections attend to local compositions.

3.2 Implementation

The implementation of the Star-Transformer is very similar to the standard Transformer, in which the information exchange is based on the attention mechanism (Vaswani et al., 2017).

Multi-head Attention Just as in the standard Transformer, we use the scaled dot-product attention (Vaswani et al., 2017). Given a sequence of vectors $\mathbf{H} \in \mathbb{R}^{n \times d}$, we can use a query vector $\mathbf{q} \in \mathbb{R}^{1 \times d}$ to soft select the relevant information with attention.

$$\text{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \quad (1)$$

where $\mathbf{K} = \mathbf{H}\mathbf{W}^K, \mathbf{V} = \mathbf{H}\mathbf{W}^V$, and $\mathbf{W}^K, \mathbf{W}^V$ are learnable parameters.

To gather more useful information from $\mathbf{H}$, similar to multi-channels in CNNs, we can use multi-head attention with k heads.

$$\text{MultiAtt}(\mathbf{q}, \mathbf{H}) = (\mathbf{a}_1 \oplus \cdots \oplus \mathbf{a}_k)\mathbf{W}^O, \quad (2)$$

$$\mathbf{a}_i = \text{Att}(\mathbf{q}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V), i \in [1, k] \quad (3)$$

where $\oplus$ denotes the concatenation operation, and $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}^O$ are learnable parameters.

Update Let $\mathbf{s}^t \in \mathbb{R}^{1 \times d}$ and $\mathbf{H}^t \in \mathbb{R}^{n \times d}$ denote the states for the relay node and all the n satellite nodes at step t. When using the Star-Transformer to encode a text sequence of length n, we start from its embedding $\mathbf{E} = [\mathbf{e}_1; \cdots ; \mathbf{e}_n]$, where $\mathbf{e}_i \in \mathbb{R}^{1 \times d}$ is the embedding of the i-th token.

We initialize the state with $\mathbf{H}^0 = \mathbf{E}$ and $\mathbf{s}^0 = average(\mathbf{E})$.

The update of the Star-Transformer at step t can be divided into two alternative phases: (1) the update of the satellite nodes and (2) the update of the relay node.

At the first phase, the state of each satellite node $\mathbf{h}_i$ are updated from its adjacent nodes, including the neighbor nodes $\mathbf{h}_{i-1}, \mathbf{h}_{i+1}$ in the sequence, the relay node $\mathbf{s}^t$, its previous state, and its corresponding token embedding.

$$\mathbf{C}_i^t = [\mathbf{h}_{i-1}^{t-1}; \mathbf{h}_i^{t-1}; \mathbf{h}_{i+1}^{t-1}; \mathbf{e}^i; \mathbf{s}^{t-1}], \quad (4)$$

$$\mathbf{h}_i^t = \text{MultiAtt}(\mathbf{h}_i^{t-1}, \mathbf{C}_i^t), \quad (5)$$

where $\mathbf{C}_i^t$ denotes the context information for the i-th satellite node. Thus, the update of each satellite node is similar to the recurrent network, except that the update fashion is based on attention mechanism. After the information exchange, a layer normalization operation (Ba et al., 2016) is used.

$$\mathbf{h}_i^t = \text{LayerNorm}(\text{ReLU}(\mathbf{h}_i^t)), i \in [1, n]. \quad (6)$$

At the second phase, the relay node $\mathbf{s}^t$ summarizes the information of all the satellite nodes and its previous state.

$$\mathbf{s}^t = \text{MultiAtt}(\mathbf{s}^{t-1}, [\mathbf{s}^{t-1}; \mathbf{H}^t]), \quad (7)$$

$$\mathbf{s}^t = \text{LayerNorm}(\text{ReLU}(\mathbf{s}^t)). \quad (8)$$

By alternatively updating update the satellite and relay nodes, the Star-Transformer finally captures all the local and non-local compositions for an input text sequence.

Position Embeddings To incorporate the sequence information, we also add the learnable position embeddings, which are concatenated with the token embeddings and shared across all the time steps.

The overall update algorithm of the Star-Transformer is shown in the Alg-1.

3.3 Output

After T rounds of update, the final states of $\mathbf{H}^T$ and $\mathbf{s}^T$ can be used for various tasks such as sequence labeling and classification. For different tasks, we feed them to different task-specific modules. For classification, we generate the fix-length sentence-level vector representation by applying a max-pooling across the final layer and mixing it with $\mathbf{s}^T$, this vector is fed into a Multiple Layer Perceptron (MLP) classifier. For the sequence labeling task, the $\mathbf{H}^T$ provides features corresponding to all the input tokens.

Algorithm 1 The Update of Star-Transformer

1: // *Initialization*
2: $\mathbf{h}_1^0, \cdots, \mathbf{h}_n^0 \leftarrow \mathbf{e}_1, \cdots, \mathbf{e}_n$
3: $\mathbf{s}^0 \leftarrow average(\mathbf{e}_1, \cdots, \mathbf{e}_n)$
4: **for** t from 1 **to** T **do**
5:      // *update the satellite nodes*
6:      **for** i from 1 **to** n **do**
7:          $\mathbf{C}_i^t = [\mathbf{h}_{i-1}^{t-1}; \mathbf{h}_i^{t-1}; \mathbf{h}_{i+1}^{t-1}; \mathbf{e}^i; \mathbf{s}^{t-1}]$
8:          $\mathbf{h}_i^t = \text{MultiAtt}(\mathbf{h}_i^{t-1}, \mathbf{C}_i^t)$
9:          $\mathbf{h}_i^t = \text{LayerNorm}(\text{ReLU}(\mathbf{h}_i^t))$
10:      // *update the relay node*
11:      $\mathbf{s}^t = \text{MultiAtt}(\mathbf{s}^{t-1}, [\mathbf{s}^{t-1}; \mathbf{H}^t])$
12:      $\mathbf{s}^t = \text{LayerNorm}(\text{ReLU}(\mathbf{s}^t))$

---

4 Comparison to the standard Transformer

Since our goal is making the Transformer lightweight and easy to train with modestly sized dataset, we have removed many connections compared with the standard Transformer (see Fig-1). If the sequence length is n and the dimension of hidden states is d, the computation complexity of one layer in the standard Transformer is $O(n^2d)$. The Star-Transformer has two phases, the update of ring connections costs $O(5nd)$ (the constant 5 comes from the size of context information $\mathbf{C}$), and the update of radical connections costs $O(nd)$, so the total cost of one layer in the Star-Transformer is $O(6nd)$.

In theory, Star-Transformer can cover all the possible relationships in the standard Transformer. For example, any relationship $\mathbf{h}_i \rightarrow \mathbf{h}_j$ in the standard Transformer can be simulated by $\mathbf{h}_i \rightarrow \mathbf{s} \rightarrow \mathbf{h}_j$. The experiment on the simulation task in Sec-5.1 provides some evidence to show the virtual node $\mathbf{s}$ could handle long-range dependencies. Following this aspect, we can give a rough analysis of the path length of dependencies in these models. As discussed in the Transformer paper (Vaswani et al., 2017), the maximum dependency path length of RNN and Transformer are $O(n)$, $O(1)$, respectively. Star-Transformer can pass the message from one node to another node via the relay node so that the maximum dependency path length is also $O(1)$, with a constat two comparing to Transformer.

Compare with the standard Transformer, all positions are processed in parallel, pair-wise connections are replaced with a "gather and dispatch" mechanism. As a result, we accelerate the Trans-

Dataset		Train	Dev.	Test	$|V|$	H DIM	#head	head DIM
Masked Summation		10k	10k	10k	-	100	10	10
SST (Socher et al., 2013)		8k	1k	2k	20k	300	6	50
MTL-16 [†] (Liu et al., 2017)	Apparel Baby Books Camera DVD Electronics Health IMDB Kitchen Magazines MR Music Software Sports Toys Video	1400	200	400	8k~28k	300	6	50
SNLI (Bowman et al., 2015)		55k	10k	10k	34k	600	6	100
PTB POS (Marcus et al., 1993)		38k	5k	5k	44k	300	6	50
CoNLL03 (Sang and Meulder, 2003)		15k	3k	3k	25k	300	6	50
OntoNotes NER (Pradhan et al., 2012)		94k	14k	8k	63k	300	6	50

Table 1: An overall of datasets and its hyper-parameters, "H DIM, #head, head DIM" indicates the dimension of hidden states, the number of heads in the Multi-head attention, the dimension of each head, respectively. MTL-16[†] consists of 16 datasets, each of them has 1400/200/400 samples in train/dev/test.

former 10 times on the simulation task and 4.5 times on real tasks. The model also preserves the ability to handle long input sequences. Besides the acceleration, the Star-Transformer achieves significant improvement on some modestly sized datasets.

5 Experiments

We evaluate Star-Transformer on one simulation task to probe its behavior when challenged with long-range dependency problem, and three real tasks (Text Classification, Natural Language Inference, and Sequence Labelling). All experiments are ran on an NVIDIA Titan X GPU. Datasets used in this paper are listed in the Tab-1. We use the Adam (Kingma and Ba, 2014) as our optimizer. On the real task, we set the embedding size to 300 and initialized with GloVe (Pennington et al., 2014). And the symbol "Ours + Char" means an additional character-level pre-trained embedding JMT (Hashimoto et al., 2017) is used. Therefore, the total size of embedding should be 400 which as a result of the concatenation of GloVe and JMT. We also fix the embedding layer of the Star-Transformer in all experiments.

Since semi- or unsupervised model is also a feasible solution to improve the model in a parallel direction, such as the ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), we exclude these models in the comparison and focus on the relevant architectures.
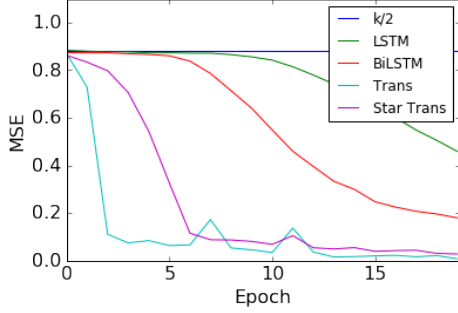
5.1 Masked Summation

In this section, we introduce a simulation task on the synthetic data to probe the efficiency and non-local/long-range dependencies of LSTM, Transformer, and the Star-Transformer. As mentioned in (Vaswani et al., 2017), the maximum path length of long-range dependencies of LSTM and Transformer are $O(n)$ and $O(1)$, where n is the sequence length. The maximum dependency path length of Star-Transformer is $O(2)$ via the relay node. To validate the ability to deal with long-range dependencies, we design a simulation task named "Masked Summation". The input of this task is a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, it has n columns and each column has d elements. The first dimension indicates the mask value $\mathbf{X}_{i0} \in \{0, 1\}$, 0 means the column is ignored in summation. The rest $d - 1$ elements are real numbers in drawn uniformly from the range $[0, 1)$. The target is a $d - 1$ dimensional vector which equals the summation of all the columns with the mask value 1. There is an implicit variable k to control the number of 1 in the input. Note that a simple baseline is always guessing the value $k/2$.

The evaluation metric is the Mean Square Error (MSE), and the generated dataset has (10k/10k/10k) samples in (train/dev/test) sets. The Fig-2 show a case of the masked summation task.
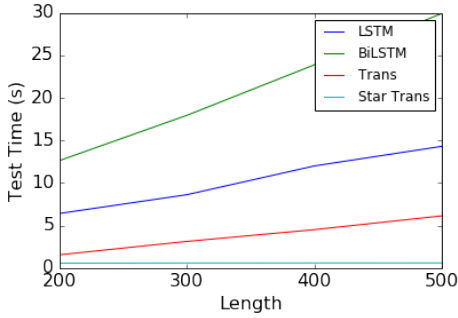
The mask summation task asks the model to recognize the mask value and gather columns in different positions. When the sequence length n is significantly higher than the number of the columns k, the model will face the long-range dependencies problem. The Fig-3a shows the per-

Input

Target

Figure 2: An example of the masked summation task, the input is a sequence of n vectors, each vector has d dimension, and there are total k vectors which have the mask value equals 1. The target is the summation of such masked vectors. In this figure, $n = 8, k = 3, d = 3$.

(a) MSE loss on the masked summation when $n = 200, k = 10, d = 10$. The $k/2$ line means the MSE loss when the model always guess the exception value $k/2$.

(b) Test Time, $k = 10, d = 10$

Model	Acc
BiLSTM (Li et al., 2015)	49.8
Tree-LSTM (Tai et al., 2015)	51.0
CNN-Tensor (Lei et al., 2015)	51.2
Emb + self-att (Shen et al., 2018a)	48.9
BiLSTM + self-att (Yoon et al., 2018)	50.4
CNN + self-att (Yoon et al., 2018)	50.6
Dynamic self-att (Yoon et al., 2018)	50.6
DiSAN (Shen et al., 2018a)	51.7
Transformer	50.4
Ours	**52.9**

Table 2: Test Accuracy on SST dataset.

5.2 Text Classification

Text classification is a basic NLP task, and we select two datasets to observe the performance of our model in different conditions, Stanford Sentiment Treebank(SST) dataset (Socher et al., 2013) and MTL-16 (Liu et al., 2017) consists of 16 small datasets on various domains. We truncate the sequence which its length higher than 256 to ensure the standard Transformer can run on a single GPU card.

For classification tasks, we use the state of the relay node s^T plus the feature of max pooling on satellite nodes $\max(\mathbf{H}^T)$ as the final representation and feed it into the softmax classifier. The description of hyper-parameters is listed in Tab-1 and Appendix.

Results on SST and MTL-16 datasets are listed in Tab-2,3, respectively. On the SST, the Star-Transformer achieves 2.5 points improvement against the standard Transformer and beat the most models.

Also, on the MTL-16, the Star-Transformer outperform the standard Transformer in all 16 datasets, the improvement of the average accuracy is 4.2. The Star-Transformer also gets better results compared with existing works. As we mentioned in the introduction, the standard Transformer requires large training set to reveal its power. Our experiments show the Star-

formance curves of models on various lengths. Although the task is easy, the performance of LSTM and BiLSTM dropped quickly when the sequence length increased. However, both Transformer and Star-Transformer performed consistently on various lengths. The result indicates the Star-Transformer preserves the ability to deal with the non-local/long-range dependencies.

Besides the performance comparison, we also study the speed with this simulation task since we could ignore the affection of padding, masking, and data processing. We also report the inference time in the Fig-3b, which shows that Transformer is faster than LSTM and BiLSTM a lot, and Star-Transformer is faster than Transformer, especially on the long sequence.

Dataset	Acc (%)				Test Time (ms)			Len.
	Star-Transformer	Transformer	BiLSTM	SLSTM	Star-Transformer	Transformer	BiLSTM	
Apparel	88.25	82.25	86.05	85.75	11	34	114	65
Baby	87.75	84.50	84.51	86.25	11	50	141	109
Books	87.00	81.50	82.12	83.44	11	55	135	131
Camera	92.25	86.00	87.05	90.02	11	52	125	116
DVD	86.00	77.75	83.71	85.52	11	58	143	139
Electronics	82.75	81.50	82.51	83.25	10	49	136	105
Health	87.50	83.50	85.52	86.50	10	45	131	85
IMDB	88.00	82.50	86.02	87.15	12	77	174	201
Kitchen	85.50	83.00	82.22	84.54	11	47	130	92
Magazines	92.75	89.50	92.52	93.75	11	51	136	115
MR	79.00	77.25	75.73	76.20	12	14	26	22
Music	83.50	79.00	78.74	82.04	11	53	137	118
Software	90.50	85.25	86.73	87.75	11	53	140	117
Sports	86.25	84.75	84.04	85.75	10	49	134	103
Toys	88.00	82.00	85.72	85.25	11	46	137	96
Video	86.75	84.25	84.73	86.75	11	56	146	131
Average	**86.98**	82.78	84.01	85.38	**10.94**	49.31	130.3	109.1

Table 3: Test Accuracy over MTL-16 datasets. "Test Time" means millisecond per batch on the test set (batch_size is 128). "Len." means the average sequence length on the test set.

Transformer could work well on the small dataset which only has 1400 training samples. Results of the time-consuming show the Star-Transformer could be 4.5 times fast than the standard Transformer on average.

5.3 Natural Language Inference

Natural Language Inference (NLI) asks the model to identify the semantic relationship between a premise sentence and a corresponding hypothesis sentence. In this paper, we use the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) for evaluation. Since we want to study how the model encodes the sentence as a vector representation, we set Star-Transformer as a sentence vector-based model and compared it with sentence vector-based models.

In this experiment, we follow the previous work (Bowman et al., 2016) to use concat($\mathbf{r}_1, \mathbf{r}_2, \|\mathbf{r}_1 - \mathbf{r}_2\|, \mathbf{r}_1 - \mathbf{r}_2$) as the classification feature. The $\mathbf{r}_1, \mathbf{r}_2$ are representations of premise and hypothesis sentence, it is calculated by $\mathbf{s}^T + \max(\mathbf{H}^T)$ which is same with the classification task. See Appendix for the detail of hyper-parameters.

As shown in Tab-4, the Star-Transformer outperforms most typical baselines (DiSAN, SPINN) and achieves comparable results compared with the state-of-the-art model. Notably, our model beats standard Transformer by a large margin, which is easy to overfit although we have made a careful hyper-parameters' searching for Transformer.

Model	Acc
BiLSTM (Liu et al., 2016)	83.3
BiLSTM + self-att (Liu et al., 2016)	84.2
300D SPINN-PI (Bowman et al., 2016)	83.2
Tree-based CNN (Mou et al., 2016)	82.1
4096D BiLSTM-max (Conneau et al., 2017)	84.5
Residual encoders (Nie and Bansal, 2017)	86.0
300D DiSAN (Shen et al., 2018a)	85.6
Gumbel TreeLSTM (Choi et al., 2018)	86.0
Reinforced self-att (Shen et al., 2018b)	86.3
2400D Multiple DSA (Yoon et al., 2018)	87.4
Transformer	82.2
Star-Transformer	**86.0**

Table 4: Test Accuracy on SNLI dataset.

The SNLI dataset is not a small dataset in NLP area, so improving the generalization ability of the Transformer is a significant topic.

The best result in Tab-4 (Yoon et al., 2018) using a large network and fine-tuned hyperparameters, they get the best result on SNLI but an undistinguished result on SST, see Tab-2.

5.4 Sequence Labelling

To verify the ability of our model in sequence labeling, we choose two classic sequence labeling tasks: Part-of-Speech (POS) tagging and Named Entity Recognition (NER) task.

Three datasets are used as our benchmark: one POS tagging dataset from Penn Treebank (PTB) (Marcus et al., 1993), and two NER datasets from CoNLL2003 (Sang and Meulder, 2003), CoNLL2012 (Pradhan et al., 2012). We use the fi-

| Model | Adv Tech | | POS | NER | |
	char	CRF	PTB Acc	CoNLL2003 F1	CoNLL2012 F1
(Ling et al., 2015)	✓	✓	97.78	-	-
(Collobert et al., 2011)	✓	✓	97.29	89.59	-
(Huang et al., 2015)	✓	✓	97.55	90.10	-
(Chiu and Nichols, 2016a)	✓	✓	-	90.69	86.35
(Ma and Hovy, 2016)	✓	✓	97.55	91.06	-
(Nguyen et al., 2016)	✓	✓	-	91.2	-
(Chiu and Nichols, 2016b)	✓	✓	-	91.62	86.28
(Zhang et al., 2018)	✓	✓	97.55	91.57	-
(Akhundov et al., 2018)	✓	✓	97.43	91.11	87.84
Transformer			96.31	86.48	83.57
Transformer + Char	✓		97.04	88.26	85.14
Star-Transformer			97.14	90.93	86.30
Star-Transformer + Char	✓		97.64	91.89	87.64
Star-Transformer + Char + CRF	✓	✓	**97.68**	**91.98**	**87.88**

Table 5: Results on sequence labeling tasks. We list the "Advanced Techniques" except widely-used pre-trained embeddings (GloVe, Word2Vec, JMT) in columns. The "Char" indicates character-level features, it also includes the Capitalization Features, Suffix Feature, Lexicon Features, etc. The "CRF" means an additional Conditional Random Field (CRF) layer.

Model	SNLI Acc	CoNLL03 Acc	MS MSE
Star-Transformer	86.0	90.93	0.0284
variant (a) -radical	84.0	89.35	0.1536
variant (b) -ring	77.6	79.36	0.0359

Table 6: Test Accuracy on SNLI dataset, CoNLL2003 NER task and the Masked Summation $n = 200, k = 10, d = 10$.

nal state of satellite nodes $\mathbf{H}^T$ to classify the label in each position. Since we believe that the complex neural network could be an alternative of the CRF, we also report the result without CRF layer.

As shown in Tab-5, Star-Transformer achieves the state-of-the-art performance on sequence labeling tasks. The "Star-Transformer + Char" has already beat most of the competitors. Star-Transformer could achieve such results without CRF, suggesting that the model has enough capability to capture the partial ability of the CRF. The Star-Transformer also outperforms the standard Transformer on sequence labeling tasks even the dataset has only 94k training samples.

5.5 Ablation Study

In this section, we perform an ablation study to test the effectiveness of the radical and ring connections.

We test two variants of our models, the first variants (a) remove the radical connections and only keep the ring connections. Without the radical connections, the maximum path length of this variant becomes $O(n)$. The second variant (b) removes the ring connections and remains the radical connections. Results in Tab-6 give some insights, the variant (a) loses the ability to handle long-range dependencies, so it performs worse on both the simulation and real tasks. However, the performance drops on SNLI and CoNLL03 is moderate since the remained ring connections still capture the local features. The variant (b) still works on the simulation task since the maximum path length stays unchanged. Without the ring connections, it loses its performance heavily on real tasks. Therefore, both the radical and ring connections are necessary to our model.

6 Conclusion and Future Works

In this paper, we present Star-Transformer which reduce the computation complexity of the standard Transformer by carefully sparsifying the topology. We compare the standard Transformer with other models on one toy dataset and 21 real datasets and find Star-Transformer outperforms the standard Transformer and achieves comparable results with state-of-the-art models.

This work verifies the ability of Star-Transformer by excluding the factor of unsupervised pre-training. In the future work, we will investigate the ability of Star-Transformer by unsupervised pre-training on the large corpus. Moreover, we also want to introduce more NLP prior knowledge into the model.

Acknowledgments

References

Adnan Akhundov, Dietrich Trautmann, and Georg Groh. 2018. Sequence labeling: A practical approach. *CoRR*, abs/1808.03926.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642. The Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *ACL (1)*. The Association for Computer Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on EMNLP*, pages 551–561.

Jason Chiu and Eric Nichols. 2016a. Sequential labeling with bidirectional lstm-cnns. In *Proc. International Conf. of Japanese Association for NLP*, pages 937–940.

Jason P. C. Chiu and Eric Nichols. 2016b. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*, pages 5094–5101. AAAI Press.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. *CoRR*, abs/1807.03819.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*, pages 1923–1933. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*, pages 1565–1575. The Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard H. Hovy. 2015. When are tree structures necessary for deep learning of representations? In *EMNLP*, pages 2304–2314. The Association for Computational Linguistics.

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

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*, pages 1520–1530. The Association for Computational Linguistics.

Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2018a. Contextualized non-local neural networks for sequence learning. *arXiv preprint arXiv:1811.08600*.

Pengfei Liu, Jie Fu, Yue Dong, Xipeng Qiu, and Jackie Chi Kit Cheung. 2018b. Multi-task learning over graph structures. *arXiv preprint arXiv:1811.10211*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL (1)*, pages 1–10. Association for Computational Linguistics.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL (1)*. The Association for Computer Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *ACL (2)*. The Association for Computer Linguistics.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: joint named entity recognition and disambiguation with rich linguistic features. *TACL*, 4:215–229.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *RepEval@EMNLP*, pages 41–45. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *EMNLP-CoNLL Shared Task*, pages 1–40. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*, pages 142–147. ACL.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455. AAAI Press.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018b. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. In *IJCAI*, pages 4345–4352. ijcai.org.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL (1)*, pages 1556–1566. The Association for Computer Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of NAACL*, pages 1480–1489.

Deunsol Yoon, Dongbok Lee, and SangKeun Lee. 2018. Dynamic self-attention : Computing attention over words dynamically for sentence embedding. *CoRR*, abs/1808.07383.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *ACL (1)*, pages 317–327. Association for Computational Linguistics.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612.