Appunti del paper *Star Transformer*

A.A. 2023/2024

Rosso Carlo

Contents

1 Introduction **2**

2 Related Work **2**

2.1 CNN/RNN 2

2.2 Kernel Methods 2

2.3 Tranformer 2

2.4 Graph Neural Networks 2

3 model **3**

3.1 Architecture 3

3.2 Implementation 3

Multi-head Attention 3

Update 3

1 Introduction

Some recent work suggest that Tranformer can be an alternative to recurrent neural netwerks (RNNs) and convolutional neural networks (CNNs). For example GPT, BERT, Transformer-XL and Universal Transformer.
However there are two limitations: the computation and memory overhead of the Tranformer are quadratic to the sequence length; Tranformer needs to be trained on large-scale datasets to achieve good performance.
A key observation is that Tranformer does not exploit prior knowledge well. The key insight is then whether leveraging strong prior knowledge can help to "lighten up" the architecture.
Star Transformer has two kinds of connections:

- **Radial connections**: preserve the non-local communication and remove the redundancy in fully-connected networks (making the computation and memory complexity linear to the sequence length);
- **Ring connections**: embody the local-compositionality prior, which has the same role as in CNNs/RNNs.

What remains to be tested is whether one shared relay node is capable of capturing the long-range dependencies.

2 Related Work

2.1 CNN/RNN

A popular approach si to represent each word as a low-dimentional vector and then learn local semantic composition functions over the given funtions over the given sentence structures. These methods are biased for learning local compositional functions and are hard to capture the long-term dependencies in a text sequence.

2.2 Kernel Methods

One method to model the non-local semantic comopsitions in a text sequence directly is to incorporate syntactic tree into the network structure for leanring sentence representations (there is a citation).

2.3 Tranformer

Another type of models learns the dependencies between words based entirely on self-attention without any recurrent or convolutional layers. However those Tranformer-based methods usually requires a large training corpus.

2.4 Graph Neural Networks

Star-Tranformer is also inspired by the recent graph networks, in which the information fusion progresses via message-passing across the whole graph. Due to its better parallel capacity and lower complexity, the Star-Tranformer is faster than RNNs or Tranformer, 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 two kinds of connections:

- **Radial Connections**: for a netweork of n satellite nodes, there are n radial connections. With the radial 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.

3.2 Implementation

The implementation is very similar to BP-Transformer and to Transformer itself.

Multi-head Attention

Just as in the standard Tranformer, we use the scaled dot-product attention. Given a sequence of vectors $H \in \mathbb{R}^{n \times d}$, we can use a query vector $q \in \mathbb{R}^{1 \times d}$ to soft select the relevant information with attention.

$$\text{Attention}(q, L, V) = \text{softmax}\left(\frac{qK^T}{\sqrt{d}}\right)V, \tag{3.1}$$

where $K = HW^K$, $V = HW^V$ and W^K and W^V are learnable parameters.
To gatehr more useful information from H, similar to multi filters in CNNs, we can use multi-head attention with k heads:

$$\text{MultiHead}(q, H) = (a_1 \oplus \cdots \oplus a_k)W^O, \tag{3.2}$$

$$a_i = \text{Attention}(qW_i^Q, HW_i^K, HW_i^V), \forall i \in [1, k], \tag{3.3}$$

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

Update

Let $s^t \in \mathbb{R}^{1 \times d}$ and $H^t \in \mathbb{R}^{n \times d}$ denote the states for the realy node and all the n satellite nodes at step t. We start from the embedding of the input $E = [e_1, \ldots, e_n]$, where $e_i \in \mathbb{R}^{1 \times d}$ is the embedding of the i-th token.
We initialize the state with $H^0 = E$ and $s^0 = average(E)$.

1. the state of each satellite node h_i are updated:

$$C_i^t = [h_{i-1}^{t-1}; h_i^{t-1}; h_{i+1}^{t-1}; e^i, s^{t-1}], \tag{3.4}$$

$$h_i^t = MultiAtt(h_i^{t-1}, C_i^t). \tag{3.5}$$

where C_i^t denotes the context information for the i-th satellite node. After the infomration exchange, a layer normalization is applied:

$$h_i^t = \text{LayerNorm}(\text{ReLU}(h_i^t)), \forall i \in [1, n]. \tag{3.6}$$

2. the state of the relay node s is updated:

$$s^t = MultiAtt(s^{t-1}, [s^{t-1}; H^t]), \tag{3.7}$$

$$s^t = \text{LayerNorm}(\text{ReLU}(s^t)). \tag{3.8}$$