Fastformer: Additive Attention Can Be All You Need

(Summary)

Abstract

Transformer is a powerful model for text understanding. However, it is inefficient due to its quadratic complexity to input sequence length (pair-wise interactions). In this paper, we propose Fastformer with linear complexity, which is an efficient Transformer model based on additive attention.

Keywords: Fastformer, additive attention

1 Introduction

...

2 Related Work

2.1 Transformer and Self-Attention

An h-head self-attention mechanism can be formulated as follows:

$$MultiHead(Q, K, V) = concat(head_1, head_2, ..., head_h)W^O$$

where $Q, K, V \in R^{N \times d}$ are input query, key and value matrices, N is sequence length, d is the hidden dimension in each attention head, and $W^O \in R^{hd \times d}$ is a linear transformation parameter matrix.

The representation learned by each attention head is formulated as follows:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) = softmax(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}})VW_i^V$$

where $W_i^Q, W_i^K, W_i^V \in R^{d \times d}$ are learnable parameters.

From this formula, we can see that the computational complexity is quadratic to the sequence length N.

2.2 Efficient Transformer

...

3 Fastformer

3.1 Architecture

A potential way to reduce the quadratic complexity is to summarize the attention matrices before modeling their interactions. Additive attention is a form of attention mechanism that can efficiently summarize important information within a sequence in linear complexity.

$$E \in R^{N \times d} \xrightarrow{\text{linear transformation}} Q, K, V \in R^{N \times d}$$

Global query vector q

- The attention weight α_i of the i-th query vector is computed as follows:

$$\alpha_i = \frac{\exp(w_q^T q_i/\sqrt{d})}{\sum_{j=1}^{N} \exp(w_q^T q_j/\sqrt{d})}$$

  $w_q \in R^d$ is a learnable parameter vector.

- The global query vector $q \in R^d$ is computed as follows:

$$q = \sum_{i=1}^{N} \alpha_i q_i$$

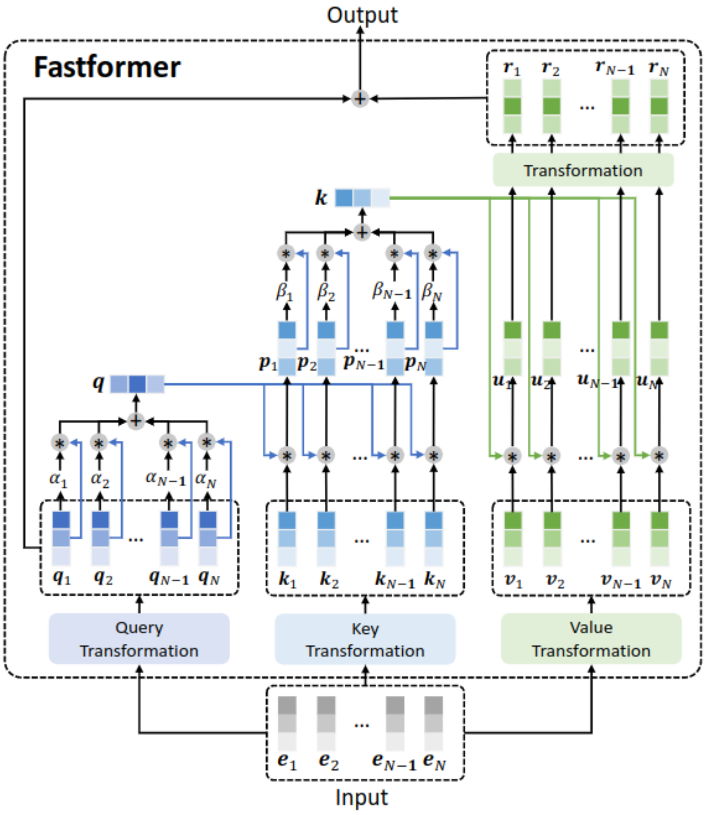
Output
Fastformer
r_1 r_2 r_{N-1} r_N
Transformation
k
β_1 β_2 β_{N-1} β_N
p_1 p_2 p_{N-1} p_N
u_1 u_2 u_{N-1} u_N
q
α_1 α_2 α_{N-1} α_N
q_1 q_2 q_{N-1} q_N
k_1 k_2 k_{N-1} k_N
v_1 v_2 v_{N-1} v_N
Query Transformation
Key Transformation
Value Transformation
e_1 e_2 e_{N-1} e_N
Input

Figure 1: The architecture of *Fastformer*.

Global key vector k

- We denote i-th vector in global context-aware key matrix as p_i, which is formulated as (the symbol $*$ means element-wise product):

$$p_i = q * k_i$$

- The attention weight β_i of the i-th key vector is computed as follows:

$$\beta_i = \frac{\exp(w_k^T p_i/\sqrt{d})}{\sum_{j=1}^{N} \exp(w_k^T p_j/\sqrt{d})}$$

  $w_k \in R^d$ is a learnable parameter vector.

- The global key vector $k \in R^d$ is computed as follows:

$$k = \sum_{i=1}^{N} \beta_i p_i$$

Global context-aware value matrix

- We denote i-th vector in key-value interaction matrix as u_i, which is formulated as:

$$u_i = k * v_i$$

- We apply a linear transformation layer to each key-value interactions vector u_i to learn its representation. The output matrix is denoted as $R = [r_1, r_2, ..., r_N] \in R^{N \times d}$.

$$U \in R^{N \times d} \xrightarrow{\text{linear transformation}} R \in R^{N \times d}$$

Output The global context-aware value matrix is further added together with the query matrix to form the final output of Fastformer.

$$R \in R^{N \times d} + Q \in R^{N \times d} \rightarrow O \in R^{N \times d}$$

By stacking multiple Fastformer layers, we can fully model contextual information. Motivated by the weight sharing techniques, we share the value and query transformation parameters to reduce the memory cost. In addition, we share the parameters across different Fastformer layers to further reduce the parameter size and mitigate the risk of overfitting.

3.2 Complexity Analysis

- Total complexity: $O(N \cdot d)$
- Total parameter: $3hd^2 + 2hd$

4 Results and discussion

We conduct extensive experiments on five benchmark datasets for different tasks: Amazon, IMDB, MIND, CNN/DM, PubMed.
We use Glove embeddings to initialize token embedding matrix. To obtain the embeddings in the classification and news recommendation tasks, we apply an additive attention network to convert the matrix output by *Fastformer* into an embedding.
Very goood!

5 Conclusion and Future Work

Chu Dinh Duc
11/01/2023