FASTFORMER

Nguyễn Trần Đăng Dương

Fastformer

1. Những hạn chế của Transformer

- Cốt lõi chính của Transformer là cơ chế self-attention, cho phép mô hình hoá context bên trong input sequence,
  + self-attention tính dot product các biểu diễn đầu vào theo từng cặp vị trí trong sequence → do đó độ phức tạp tính toán sẽ là $O(N^2)$ với N: là độ dài của input sequence → xử lý không hiệu quả với long input sequence

- Có nhiều biến thể của Transformer để có thể xử lý với long input sequence (BigBird, Linformer, Longformer, Linear Transformer, ...) tuy nhiên vẫn chưa mô hình hoá đầy đủ được global context.

2. Transformer và cơ chế self-attention.

- Transformer được xây dựng chủ yếu multi-head self-attention để có thể mô hình hoá hiệu quả contexts bên trong sequence bằng cách capture tương tác giữa từng cặp vị trí trong input sequence. Một h-head self-attention có biểu thức sau:

$$\text{Multi-head}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h) \cdot W^O$$

Trong đó + $Q, K, V \in \mathbb{R}^{N \times d}$ là ma trận đầu vào query, key, value. với N là độ dài sequence, d là hidden dimension tại mỗi attention head
+ $W^O \in \mathbb{R}^{hd \times d}$ là learnable parameter, dùng để linear transformation.

- Mỗi attention head có biểu thức như sau:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \text{softmax}\left(\frac{QW_i^Q \cdot (KW_i^K)^T}{\sqrt{d}}\right) \cdot VW_i^V$$

Trong đó $+, W_i^Q, W_i^K, W_i^V \in R^{d \times d}$ là ma trận learnable parameter

→ Từ biểu thức trên ta có thể thấy độ phức tạp tính toán là $O(N^2)$, với N là độ dài của input sequence

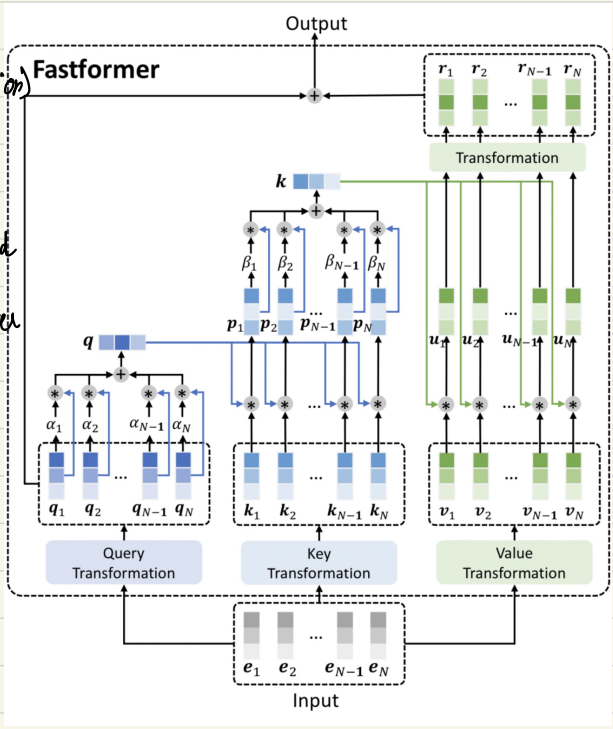
3. Fastformer

- Là 1 biến thể của Transformer, sử dụng cơ chế additive attention để mô hình hoá hiệu quả contexts sequence với độ phức tạp tuyến tính.

Kiến thức của fastformer

- Input matrix $E \in R^{N \times d}$ (với N là độ dài của input sequence, d là hidden dimension), các vector tương ứng sẽ là $[e_1, e_2 ... e_N]$
- Thực hiện phép linear transformation với input matrix:
  + $Q = E.W^Q$
  + $K = E.W^K$
  + $V = E.W^V$

  } Với $W^Q, W^K, W^V \in R^{d \times d}$ là learnable parameter matrix
- Khi đó $Q, K, V \in R^{N \times d}$
  + $Q = [q_1, q_2 ... q_N]$
  + $K = [k_1, k_2 ... k_N]$
  + $V = [v_1, v_2 ... v_N]$
- Mô hình hoá contexts của input sequence dựa trên tương tác của Query, Key, Value
  + một cách hiệu quả để giảm độ phức tạp tính toán đó là summarize attention matrix trước khi mô hình hoá tương tác giữa chúng. → Additive attention có thể summarize một cách hiệu quả các thông tin quan trọng trong thời gian tuyến tính



+) Sử dụng additive attention để summarize query matrix thành global query vector; vector này sẽ có đóng thông tin global context bên trong attention query. Attention weight α_i tại vector thứ i trong query matrix được tính bằng

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

Với $w_q \in R^d$ là learnable parameter vector

→ Khi đó global attention query vector $q \in R^d$ được tính bằng

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

α_1 [$q_{11}\ q_{12} \dots q_{1d}$]

\+

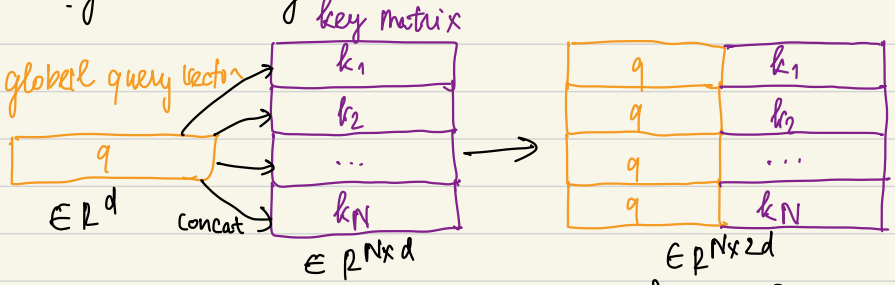
α_2 [$q_{21}\ q_{22} \dots q_{2d}$] → [q] $\in R^d$

\+

α_N [$\alpha_{N1}\ \alpha_{N2} \dots \alpha_{Nd}$]

- Mô hình hoá tương tác giữa global query vector với key matrix

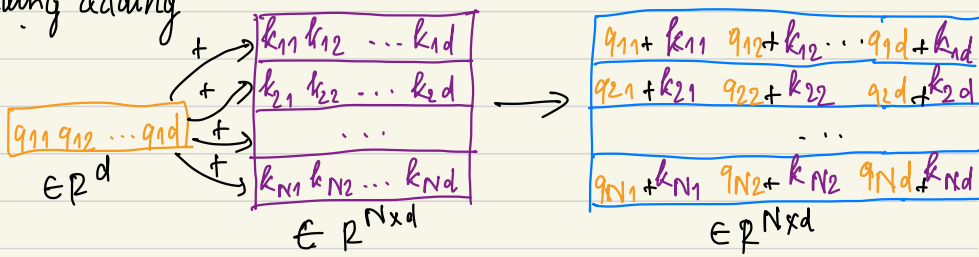
+) Sử dụng concatenating



→ Không mô hình hoá được context vì chỉ đơn giản là concat 2 vector

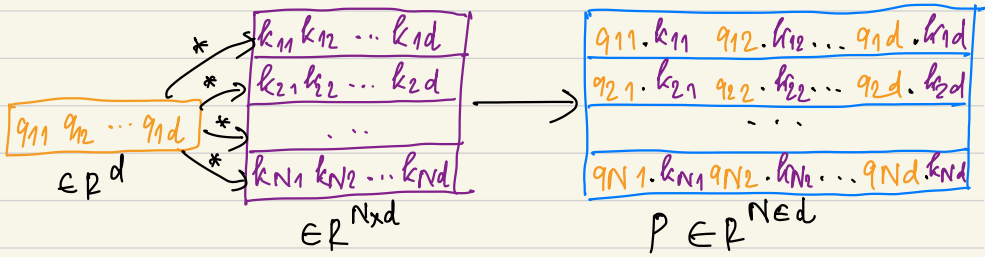
mà không xem xét tới tương tác giữa chúng

+, Sử dụng adding



→ không là phương pháp tối ưu vì chỉ có thể mô hình hoá tương tác tuyến tính giữa 2 vector, khiến có thể không học chính xác biểu diễn contexts

+, Sử dụng element-wise product $p_i = q * k_i \quad P \in \mathbb{R}^{N \times d}$



→ Mô hình hoá được mối quan hệ phi tuyến giữa 2 vector, giúp mô hình hoá được context phức tạp trong long sequence

– Tương tự sử dụng additive attention để summarize global context-aware key matrix P. Additive attention weigh β_i tại vector p_i được tính bởi

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

$w_k \in \mathbb{R}^d$ là learnable parameter vector

→ Khi đó global key vector $k \in R^d$ được tính bằng

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

- Tương tự để mô hình hoá tương tác giữa global key vector và attention value matrix, ta sử dụng element-wise product

$$u_i = k * v_i \quad . \quad u \in R^{N \times d}$$

- Sử dụng Linear transformation với ma trận tương tác key-value u để học những biểu diễn ẩn. Output matrix sẽ là $R = [r_1, r_2 \dots r_N] \in R^{N \times d}$
- Ma trận R sẽ được cộng với ma trận Query ban đầu → output của Fastformer

→ Mỗi global key và query vector sẽ tương tác với value và key vector để học biểu diễn contexts

→ Stacking nhiều Fastformer layers sẽ mô hình hoá đầy đủ thông tin contexts

→ Sử dụng kĩ thuật parameters sharing, chia sẻ transformation parameter trong value và query để giảm chi phí memory. Thêm vào đó sử dụng parameter sharing giữa các layer Fastformer khác nhau để giảm overfitting

⊛ Độ phức tạp tính toán

- Additive attention để học global query và global key sẽ có chi phí tính toán và chi phí bộ nhớ đều là $O(N.d)$ với N: là độ dài input sequence
  d: là hidden dimension
- Element-wise product cũng có chi phí tính toán và chi phí phí bộ nhớ là $O(N.d)$

→ Vậy độ phức tạp sẽ là $O(N.d)$, hiệu quả hơn rất nhiều so với Transformer với độ phức tạp $O(N^2.d)$

- Nếu kĩ thuật parameters sharing được sử dụng, tổng số lượng param tại mỗi layer Fastformer sẽ là $3hd^2 + 2hd$ << với ít nhất $4hd^2$ param trong Transformer

(*) Những bài toán mà Fastformer cho thấy performance gần như là tốt nhất so với các biến thể khác của Transformer
- News recommendation task ← text modeling, understanding user interest
- Text summarization task
- Sentiment and topic classification task

(*) Ảnh hưởng của parameter sharing
- Query - Value sharing: có performance tốt hơn một chút với không sharing
- Query - Value + Head wise sharing: performance bị tụt giảm do các attention-head có xu hướng capture những mẫu contexts khác nhau, phải sharing sẽ không có lợi trong việc mô hình hoá contexts
- Query - Value + Layer wise sharing: tăng performance vì giúp giảm thiểu overfitting

→ Fastformer sử dụng query-value + layer wise sharing sẽ làm tăng performance cũng như giảm parameters size