KOREA UNIVERSITY

DSBA Transformer survey paper study

A Survey of Transformers

#3: Attention 2

arXiv preprint

DSBA Data Science & Business Analytics

고려대학교 산업경영공학과

Data Science & Business Analytics Lab

이유경, 김명섭, 윤훈상, 김지나, 허재혁, 김수빈

발표자 : 김지나

03

발표 목차

Contents

03 Overview

1. Lineaerized Attention

- Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention (ICML 2020, 110회 인용)
- Masked Language Modeling for Proteins via Linearly Scalable Long-Context Transformers (arXiv 2020, 17회 인용)
- Random Feature Attention (ICLR 2021, 21회 인용)
- Rethinking Attention with Performers. (ICLR 2021, 117회 인용)
- Linear Transformers Are Secretly Fast Weight Memory Systems (arXiv 2021, 5회 인용)

2. Prototype and Memory Compression

- Fast Transformers with Clustered Attention (arXiv 2020, 20회 인용)
- Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting (AAAI 2021, 17회 인용)
- Generating Wikipedia by Summarizing Long Sequences(ICLR 2018, 379회 인용)
- Set Transformer (In Proceedings of ICML, 2019, 229회 인용), Luna (arXive 2021, 1회 인용)
- Luna: Linear Unified Nested Attention. (arXiv 2021, 1회 인용)

03 Linearized Attention
개념

T: sequence length

Linearization을 통해 attention의 computational complexity $\boldsymbol{O(T^2)} \rightarrow \boldsymbol{O(T)}$줄임

- 기존 Attention: $Q, K, V \in R^{TXD}$에 대한 attention matrix를 위한 $softmax(QK^T)V$ 연산
  - QK^T연산은 Quadratic, computational complexity $\boldsymbol{O(T^2)}$

- Linearized Attention: $softmax(QK^T)$ 연산을 위해 QK^T를 $Q'K'^T$로 disentangle
  - Computational complexity $\boldsymbol{O(T)}$
  - $K'^T V$ 연산 먼저 수행 후, Q'와 연산
  - $Q'K'^T V \Rightarrow Q'(K'^T V)$

03 **Linearized Attention**
개념

Linearized Attention

- Un-normalized attention matrix

$$\hat{\mathbf{A}} = \exp(\mathbf{Q}\mathbf{K}^\top)$$

  - $\exp(\cdot)$ is applied element-wise
  - 기존 softmax 취한 score에 따른 attention matrix에서 normalization을 위한 denominator 생략
- Regular Attention

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V} = \mathbf{A}\mathbf{V}$$

$$\mathbf{Z} = \mathbf{D}^{-1}\hat{\mathbf{A}}\mathbf{V} \quad \text{where} \quad \mathbf{D} = \text{diag}(\hat{\mathbf{A}}\mathbf{1}_T^\top)$$

$\mathbf{1}_T^\mathrm{T}$: the all-ones column vector of length T

Linearized Attention

- Approximate or replace the unnormalized attention matrix $\exp(QK^{T})$ with $\phi\,(Q)\phi\,(K)^{T}$

$$\hat{\mathbf{A}} = \exp(\mathbf{Q}\mathbf{K}^{\top}) \Rightarrow \phi(\mathbf{Q})\phi(\mathbf{K})^{\top}$$

  - ϕ: is a feature map that is applied in row-wise manner

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})}\mathbf{v}_j, \Rightarrow \mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^{\top}}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^{\top}}\mathbf{v}_j = \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^{\top}},$$

Regular Attention의 $sim(\cdot,\cdot)$

: *the exponential of inner product* $exp(\langle\cdot,\cdot\rangle)$

$sim(\cdot,\cdot)$: *a kernel function* $K\,(x,y) = \phi\,(x)\phi\,(y)^{T}$

$\otimes$: *outer product*

T: sequence length

Linearized Attention

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j,$$

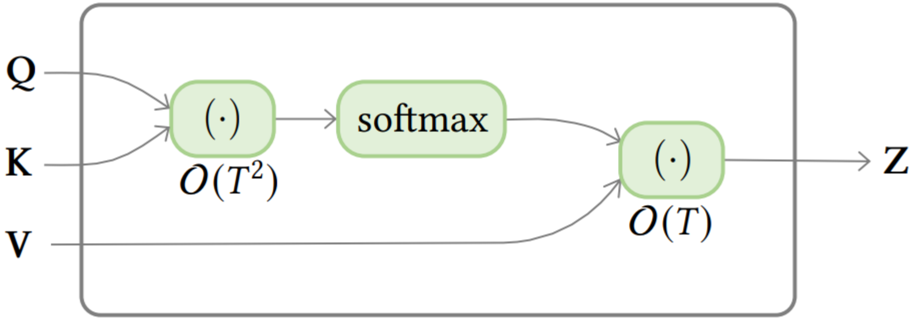
Regular Attention의 $sim(\cdot,\cdot)$

: *the exponential of inner product* $exp(\langle \cdot,\cdot \rangle)$

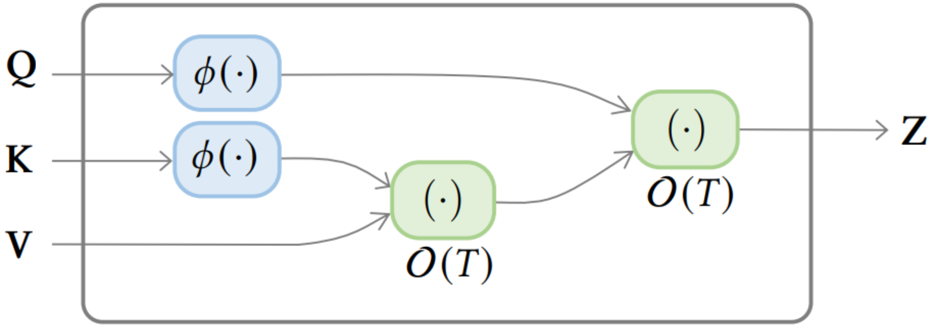
$$\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j = \frac{\phi(\mathbf{q}_i) \sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i) \sum_{j'} \phi(\mathbf{k}_{j'})^\top},$$

$sim(\cdot,\cdot)$: *a kernel function* $K(x,y) = \phi(x)\phi(y)^{\mathrm{T}}$

$\otimes$: *outer product*



(a) standard self-attention



(b) linearized self-attention

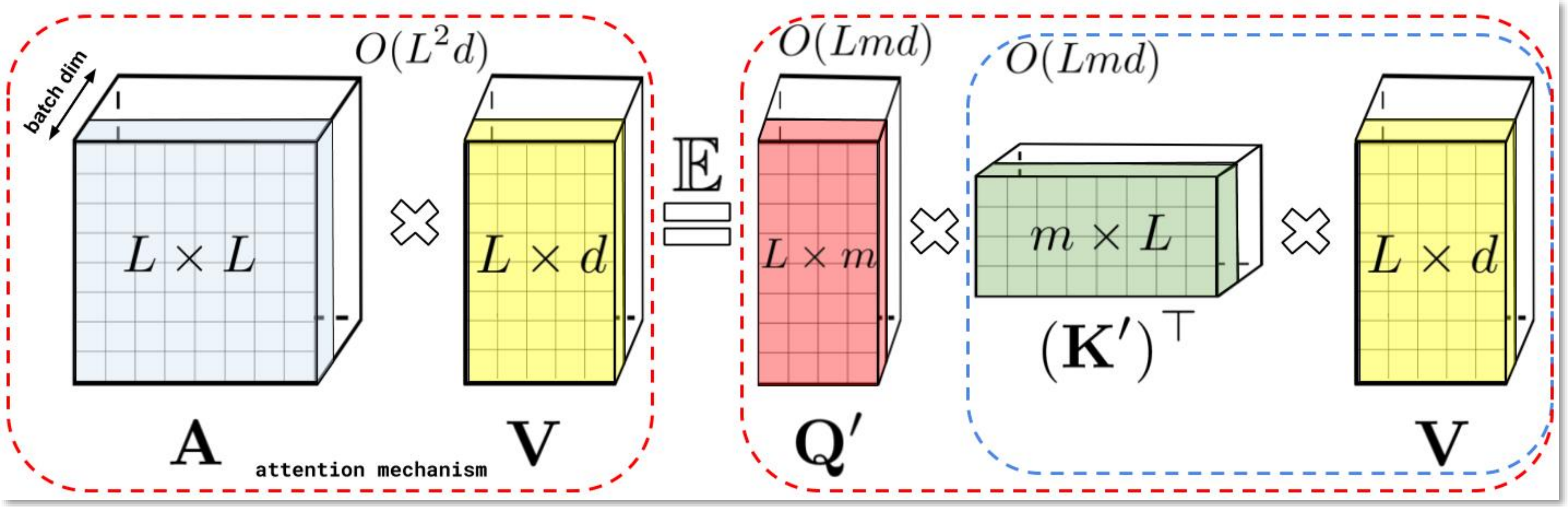
출처: https://ai.googleblog.com/2020/10/rethinking-attention-with-performers.html

Linearized Attention

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j, \quad \Rightarrow \quad \mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j$$

Regular Attention의 $sim(\cdot,\cdot)$

: $the\ exponential\ of\ inner\ product\ \ exp(\langle\cdot,\cdot\rangle)$



출처: https://ai.googleblog.com/2020/10/rethinking-attention-with-performers.html

Linearized Attention

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j,$$

Regular Attention의 $sim(\cdot,\cdot)$

: *the exponential of inner product* $exp(\langle\cdot,\cdot\rangle)$

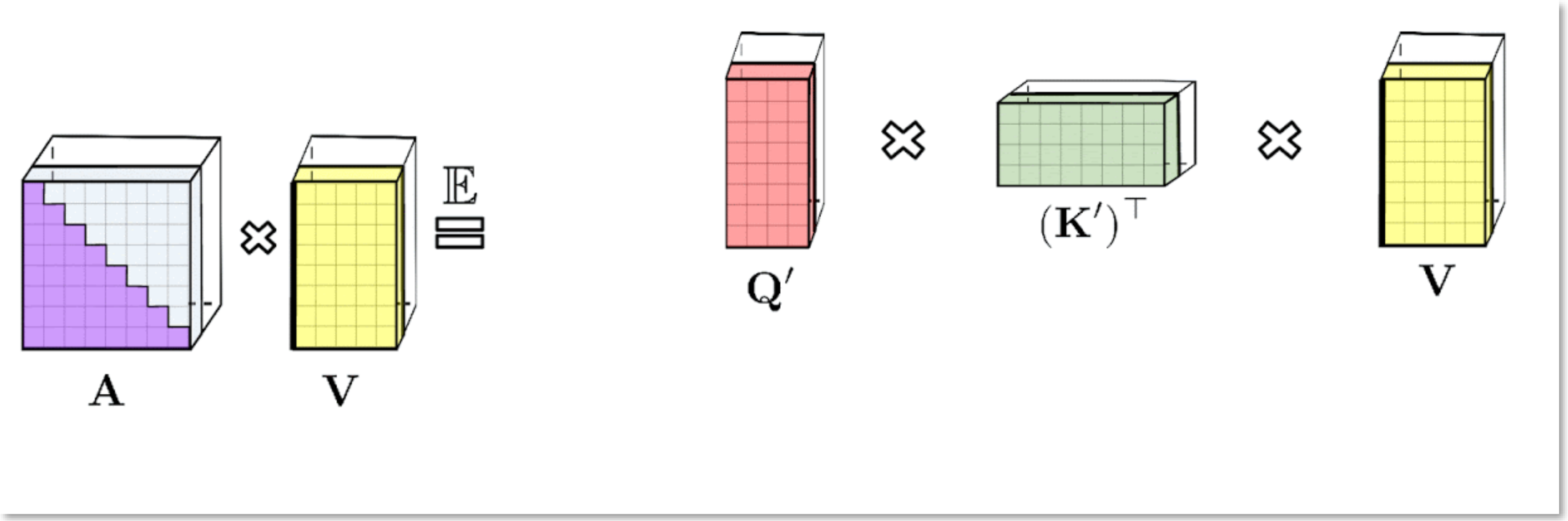
$sim(\cdot,\cdot)$: *a kernel function* $K(x,y) = \phi(x)\phi(y)^{T}$

$\otimes$: *outer product*

$$\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j$$

Vector 연산의 summation

$$= \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top},$$



03 Linearized Attention
개념

Linearized Attention

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j,$$

Regular Attention의 $sim(\cdot,\cdot)$

: *the exponential of inner product* $exp(\langle\cdot,\cdot\rangle)$

$$\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j$$

Vector 연산의 summation

$$= \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top},$$

$sim(\cdot,\cdot)$: *a kernel function* $K(x,y) = \phi(x)\phi(y)^{\mathrm{T}}$

$\otimes$: *outer product*

Attention can be linearized by first computing the highlighted terms

연산량 매우 줄어듦

03 Linearized Attention
개념

Linearized Attention

$$
\begin{aligned}
\mathbf{z}_i &= \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j \\
&= \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top},
\end{aligned}
$$

- Memory matrix

$$\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j$$

  - retrieve a value by multiplying the memory matrix with feature mapped query with proper normalization.

$$\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j$$

  - maintains a memory matrix by aggregating associations represented by outer products of (feature mapped) keys and values

(1) Feature map $\boldsymbol{\phi}(\cdot)$ (2) Aggregation rule

출처: https://kjhov195.github.io/2020-01-07-activation_function_2/

03 Linearized Attention

(1) Feature Maps

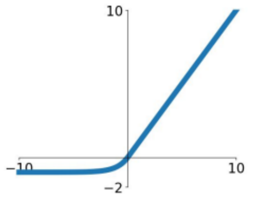
Feature Maps

- Linear Transformer (ICML 2020, 110회 인용)
  - Simple feature map

$$\boldsymbol{\phi}_{\mathbf{i}}(\mathbf{x}) = \mathrm{elu}(\boldsymbol{x}_i) + 1$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



기존의 dot product attention을 approximate하는 것을 목표로 하지 않고, 비슷한 수준의 성능을 내는 것을 목표로 하여, standard transformer에 준하는 성능 달성

Method	Validation PER	Time/epoch (s)
Bi-LSTM	10.94	1047
Softmax	5.12	2711
LSH-4	9.33	2250
Linear (ours)	8.08	**824**

- Speech recognition 실험 결과, linear transformer를 사용하였을 때, PER을 8까지 낮춰, 다른 모델에 비해 softmax와 가장 성능이 유사하며, 소요 시은 softmax의 3배 이상 감소했다.

Feature Maps

- Performer – first version (arXiv 2020, 17회 인용)

  기존의 dot product attention을 approximate하는 것을 목표로 함

  - Random feature map

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}[f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

$$f_1, \cdots, f_l : \mathbb{R} \to \mathbb{R} \text{ and } h : \mathbb{R}^D \to \mathbb{R}.$$

where $\omega_1, \cdots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ are drawn from some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^D)$

Softmax를 approximate하기 위해 아래와 같은 kernel 함수를 사용

$$h(\mathbf{x}) = \exp(\frac{\|\mathbf{x}\|^2}{2}), l = 2, f_1 = \sin, f_2 = \cos.$$

Feature Maps

- Performer – first version (arXiv 2020, 17회 인용)

  기존의 dot product attention을 approximate하는 것을 목표로 함

  - Random feature map

Theorem 1 (Rahimi & Recht, 2007). *Let $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{2D}$ be a nonlinear transformation:*

$$\phi(\mathbf{x}) = \sqrt{1/D}\left[\sin(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \sin(\mathbf{w}_D \cdot \mathbf{x}), \cos(\mathbf{w}_1 \cdot \mathbf{x}), \ldots, \cos(\mathbf{w}_D \cdot \mathbf{x})\right]^\top.$$

When d-dimensional random vectors $\mathbf{w}_i$ are independently sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$,

$$\mathbb{E}_{\mathbf{w}_i}[\phi(\mathbf{x}) \cdot \phi(\mathbf{y})] = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2\right).$$

03 Linearized Attention
(1) Feature Maps

Feature Maps

- Random Feature Attention (ICLR 2021, 21회 인용)

  Performer(ver. 1)와 유사

  - Random feature map

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}[f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

$$f_1, \cdots, f_l : \mathbb{R} \to \mathbb{R} \text{ and } h : \mathbb{R}^D \to \mathbb{R}.$$

where $\omega_1, \cdots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ are drawn from some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^D)$

query, key를 feature space에 보내기 전 $\mathbf{l_2}$−normalization 하기 때문에, $\boldsymbol{h(x)} = \mathbf{1}$

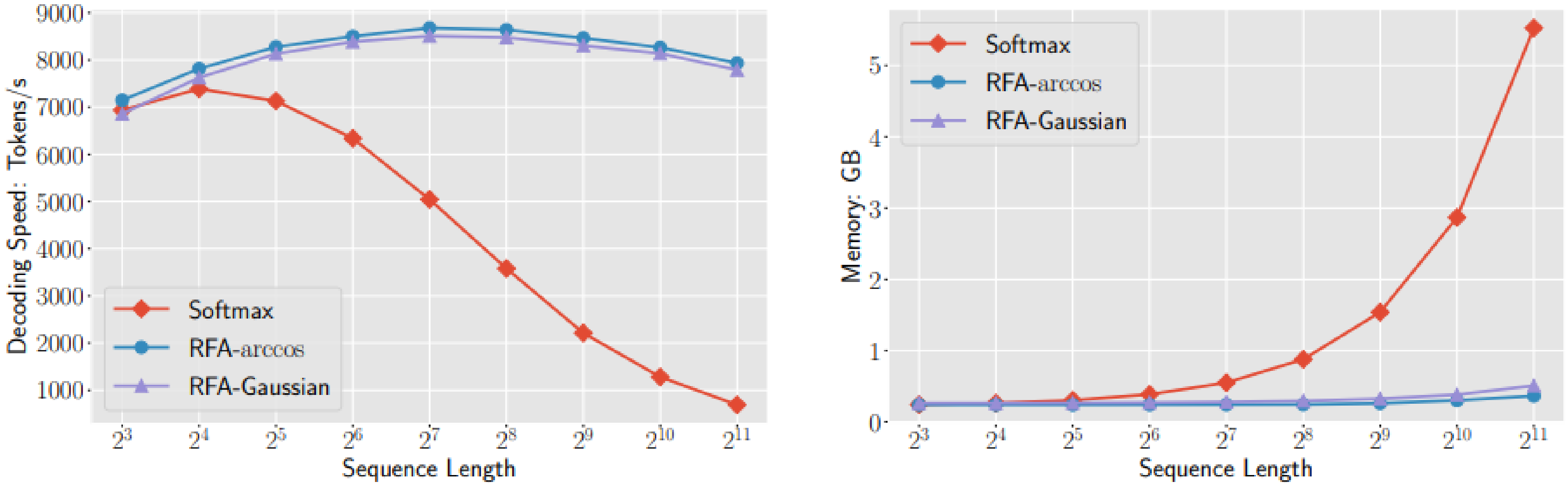
$$h(\mathbf{x}) = \exp(\frac{\|\mathbf{x}\|^2}{2}), l = 2, f_1 = \sin, f_2 = \cos.$$

Feature Maps

- Random Feature Attention (ICLR 2021, 21회 인용)

  Performer(ver. 1)와 유사

  - Random feature map



(a) Speed vs. lengths. (b) Memory vs. lengths.

Feature Maps

- Random Feature Attention (ICLR 2021, 21회 인용)

  Performer(ver. 1)와 유사

  - Random feature map

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}\left[f_1(\omega_1^\top \mathbf{x}), \cdots, f_1(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})\right],$$

$$f_1, \cdots, f_l : \mathbb{R} \to \mathbb{R} \text{ and } h : \mathbb{R}^D \to \mathbb{R}.$$

$$\text{where } \omega_1, \cdots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D} \text{ are drawn from some distribution } \mathcal{D} \in \mathcal{P}(\mathbb{R}^D)$$

query, key를 feature space에 보내기 전 $\mathbf{l_2}$-normalization 하기 때문에, $\boldsymbol{h(x) = 1}$

The trigonometric random feature map leads to an unbiased approximation,
it does not guarantee non-negative attention scores

$$h(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x}\|^2}{2}\right), l = 2, f_1 = \sin, f_2 = \cos.$$

Feature Maps

- Performer – second version (ICLR 2021, 117회 인용)
  - Positive random feature map

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}[f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

$$f_1, \cdots, f_l : \mathbb{R} \rightarrow \mathbb{R} \text{ and } h : \mathbb{R}^D \rightarrow \mathbb{R}.$$

where $\omega_1, \cdots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ are drawn from some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^D)$

$$h(\mathbf{x}) = \exp(-\frac{\|\mathbf{x}\|^2}{2}), l = 1, f_1 = \exp$$

Guarantees unbiased and non- negative
approximation of dot-product attention
⇒ Performer(ver. 1) 보다 stable하고,
더 좋은 approximation 결과 보임

Feature Maps

- Performer – second version (ICLR 2021, 117회 인용)
  - Positive random feature map

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}[f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

$$f_1, \cdots, f_l : \mathbb{R} \to \mathbb{R} \text{ and } h : \mathbb{R}^D \to \mathbb{R}.$$

where $\omega_1, \cdots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ are drawn from some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^D)$

$$h(\mathbf{x}) = 1, l = 1, f_1 = \text{ReLU}.$$

Effective in various tasks including machine translation and protein sequence modeling.

Feature Maps

- Linear Transformers Are Secretly Fast Weight Memory Systems (arXiv 2021, 5회 인용)

feature space 상에서의 orthogonality를 이용할 수 있는 feature map 제안

$$\phi_{i+2(j-1)D}(\mathbf{x}) = \mathrm{ReLU}([\mathbf{x}, -\mathbf{x}])_i \mathrm{ReLU}([\mathbf{x}, -\mathbf{x}])_{i+j}$$

for $i = 1, \cdots, 2D$, $j = 1, \cdots, \nu$.

$input\ x \in R^D$

$the\ feature\ map\ \phi : R^D \rightarrow R^{2\nu D}$

Aggregation Rule

The associations $\{\phi\,(k)_j\ \otimes\ \mathrm{v}_j\}$ are aggregated into the memory matrix by simple summation

⇒ 새로운 association을 memory network S에 추가할 때, 선택적으로 association을 drop하는 것이 효과적

Aggregation Rule

- Random Feature Attention (ICLR 2021, 21회 인용)
  - Gating mechanism

$$
\begin{aligned}
g_t &= \mathrm{sigmoid}(\mathbf{w}_g \cdot \mathbf{x}_t + b_g), \\
\mathbf{S}_t &= g_t\, \mathbf{S}_{t-1} + (1 - g_t)\, \boldsymbol{\phi}\,(\mathbf{k}_t) \otimes \mathbf{v}_t, \\
\mathbf{z}_t &= g_t\, \mathbf{z}_{t-1} + (1 - g_t)\, \boldsymbol{\phi}\,(\mathbf{k}_t)\,.
\end{aligned}
$$

  - $\mathrm{w_g}$ and $b_g$are learned parameters, and $\mathrm{x_t}$ is the input representation at timestep t.
  - By multiplying the learned scalar gates 0 〈 g_t 〈 1 against the hidden state (S_t, z_t), history is exponentially decayed, favoring more recent context.

Aggregation Rule

- Linear Transformers Are Secretly Fast Weight Memory Systems (arXiv 2021, 5회 인용)

  Assoociation의 단순 합으로 memory matrix를 update하는 것은, memory matrix의 capacity를 제한하는 것, 따라서 write-and-remove를 통해 capacity를 확장하는 방식을 제안

  - Write-and-remove update

$$
\begin{aligned}
\boldsymbol{k}^{(i)}, \boldsymbol{v}^{(i)}, \boldsymbol{q}^{(i)} &= \boldsymbol{W}_k \boldsymbol{x}^{(i)}, \boldsymbol{W}_v \boldsymbol{x}^{(i)}, \boldsymbol{W}_q \boldsymbol{x}^{(i)} \\
\bar{\boldsymbol{v}}^{(i)} &= \boldsymbol{W}^{(i-1)} \phi(\boldsymbol{k}^{(i)}) \\
\beta^{(i)} &= \sigma(\boldsymbol{W}_\beta \boldsymbol{x}^{(i)}) \\
\boldsymbol{v}_{\text{new}}^{(i)} &= \beta^{(i)} \boldsymbol{v}^{(i)} + (1 - \beta^{(i)}) \bar{\boldsymbol{v}}^{(i)}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{W}^{(i)} &= \boldsymbol{W}^{(i-1)} \underbrace{+ \boldsymbol{v}_{\text{new}}^{(i)} \otimes \phi(\boldsymbol{k}^{(i)})}_{\text{write}} \underbrace{- \bar{\boldsymbol{v}}^{(i)} \otimes \phi(\boldsymbol{k}^{(i)})}_{\text{remove}} \\
&= \boldsymbol{W}^{(i-1)} + \beta^{(i)} (\boldsymbol{v}^{(i)} - \bar{\boldsymbol{v}}^{(i)}) \otimes \phi(\boldsymbol{k}^{(i)})
\end{aligned}
$$

$$
\boldsymbol{y}^{(i)} = \boldsymbol{W}^{(i)} \phi(\boldsymbol{q}^{(i)})
$$

$\beta(i)$ is the "write-strength. only depends on input $x(i)$

새로운 input key-value pair가 들어오면,

1) k^i와 직전 memory matrix W^{i-1}의 association $\bar{v}^i$ 을 구하고

2) 현재 v^i와 $\bar{v}^i$의 convex combination을 memory matrix에 update함

03 Query Prototyping and Memory Compression

개념

목적

Reduce the complexity of attention by

(1) Query prototyping: reducing the number of **queries**

(2) Memory compression: reducing the number of **key-value pairs**

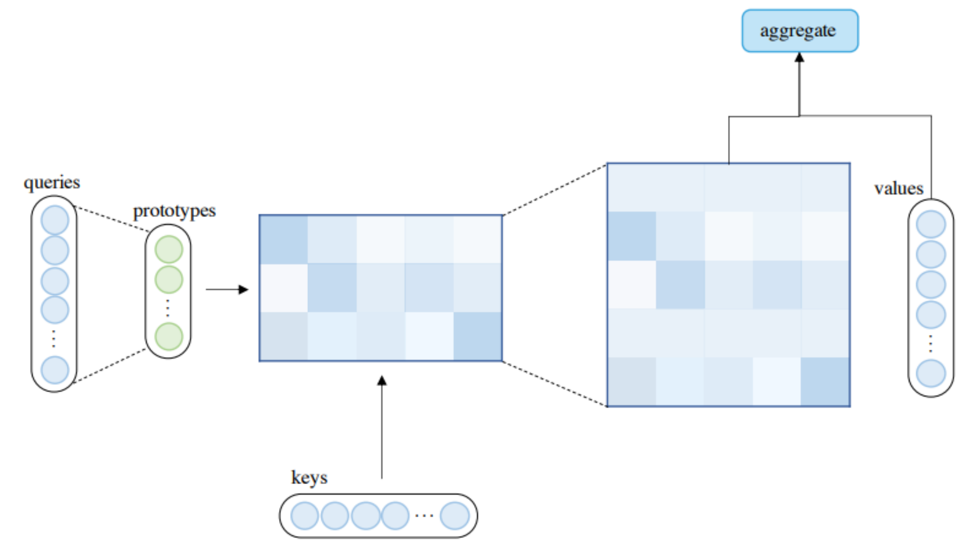
The key-value pairs are often referred to as a **key-value memory**

Attention with Prototype Queries

Several prototypes of queries serve as the main source to compute attention distributions

query representation 중,

- 특정 position의 query distribution을 copy
- discrete uniform distribution으로 채움



(a) Query prototyping

Attention with Prototype Queries

- Clustered Attention (arXiv 2020, 20회 인용)
  - groups queries into several clusters and then computes attention distributions for cluster centroids.

① Centroid 구하기

$$Q_j^c = \frac{\sum_{i=1}^{N} S_{ij} Q_i}{\sum_{i=1}^{N} S_{ij}}$$

$S_{ij} = 1$, if the i-th query Q_i belongs to the j-th cluster and 0 otherwise.

② Centroid에 대한 attention score 계산

$$A^c = \mathrm{softmax}\left(\frac{Q^c K^T}{\sqrt{D_k}}\right)$$

$Q^c \in \mathbb{R}^{C \times D_k}$ as the centroid matrix

③ Centroid에 의한 새로운 value 계산

$$\hat{V}^c = A^c V.$$

④ 가장 가까운 centroid에 대한 attention value 도출

$$\hat{V}_i = \sum_{j=1}^{C} S_{ij} \hat{V}_j^c.$$

Attention with Prototype Queries

- Clustered Attention (arXiv 2020, 20회 인용)
  - groups queries into several clusters and then computes attention distributions for cluster centroids.

	full	clustered-100	i-clustered-100
WER (%)	15.0	18.5	15.5
Time/Epoch (h)	3.84	1.91	2.57
Convergence Time (h)	228.05	132.13	127.44

03 Query Prototyping and Memory Compression

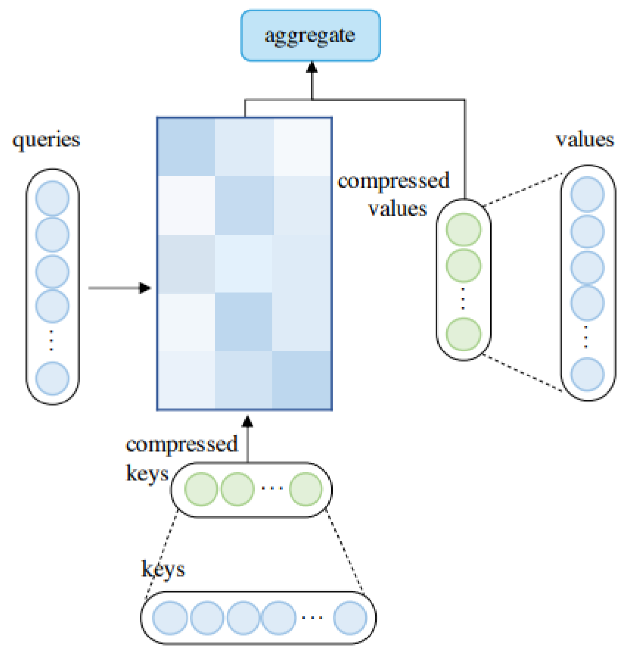
(1) Attention with Prototype Queries

Attention with Prototype Queries

- Informer (AAAI 2021, 17회 인용)
  - Query sparsity measurement를 제안하여, 상위 u개의 query만을 가지고 attention distribution 계산
    - 나머지 query에는 discrete uniform distribution 부여
  - Query sparsity measurement
    - Query의 attention distribution과 the discrete uniform distribution 사이의 Kullback-Leibler divergence 값을 기반으로 정의
    - Attention distribution이 the discrete uniform distribution과의 차이가 클수록 몇몇 key에 dominant한 attention을 주는 query라고 할 수 있기 때문에, KLD가 클수록 중요한 query로 봄

Attention with Compressed Key–Value Memory

reduce the complexity by reducing the number of the key–value pairs before applying the attention mechanism



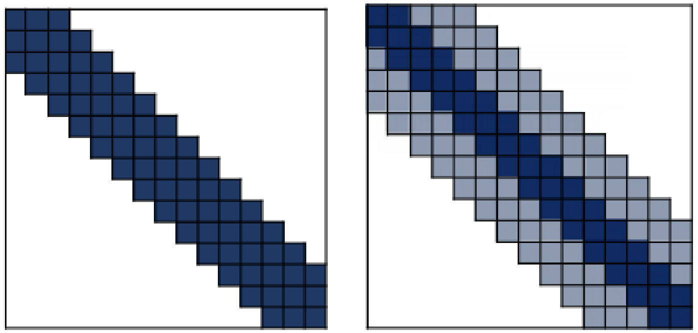
(b) Memory compression

Attention with Compressed Key-Value Memory

- Memory Compressed Attention(ICLR 2018, 379회 인용)
  - Strided convolution을 사용하여 key와 value의 개수를 줄임
  - Kernel size k에 따라 줄이고자 하는 key, value 개수를 조절하며, attention 연산량이 줄어들기 때문에, 같은 시간 안에 vanilla transformer보다 훨씬 긴 sequence를 처리할 수 있다.
- Set Transformer (In Proceedings of ICML 2019, 229회 인용),
- Luna (arXive 2021, 1회 인용)
  - Trainable global node를 정의하여, input으로부터의 정보를 취합하게 하며, 취합된 정보는 input이 attend할 memory로 사용된다.

Attention with Compressed Key-Value Memory

- Linformer(arXiv 2020, 132회 인용)
  - Key, value에 대해 linear projection을 적용하여 length n에서 더 짧은 n_k의 sequence로 사영
  - 단점: autoregressive attention에 사용할 수 없음
- Poolingformer (ICML 2021, 1회 인용)
  - 여러 개의 pooling operation을 사용하여, key와 value를 줄이면서도 receptive field를 키우고자 함



(a) Single-level local attention (b) Two-level pooling attention

03 Summary

1. Lineaerized Attention

- Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention (ICML 2020, 110회 인용)
- Masked Language Modeling for Proteins via Linearly Scalable Long-Context Transformers (arXiv 2020, 17회 인용)
- Random Feature Attention (ICLR 2021, 21회 인용)
- Rethinking Attention with Performers. (ICLR 2021, 117회 인용)
- Linear Transformers Are Secretly Fast Weight Memory Systems (arXiv 2021, 5회 인용)

2. Prototype and Memory Compression

- Fast Transformers with Clustered Attention (arXiv 2020, 20회 인용)
- Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting (AAAI 2021, 17회 인용)
- Generating Wikipedia by Summarizing Long Sequences(ICLR 2018, 379회 인용)
- Set Transformer (In Proceedings of ICML, 2019, 229회 인용), Luna (arXive 2021, 1회 인용)
- Luna: Linear Unified Nested Attention. (arXiv 2021, 1회 인용)

감사합니다