Practical No. 4

Emilia Wiśnios
ew407219@students.mimuw.edu.pl

May 2, 2022

Part 2

a)

Since $c = \sum_{i=1}^{n} v_i \alpha_i$ we can get $c \approx v_j$ when

$$\alpha_i \approx \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j \end{cases}.$$

To achieve this following condition must be satisfied:

$$k_j^T q >> k_i^T q \text{ for } i \neq j$$

because

$$\alpha_i = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)}.$$

b)

To get $c \approx \frac{1}{2}(v_a + v_b)$, both α_a and α_b must be approximately equal to $\frac{1}{2}$ and α_i for $i \notin \{a, b\}$ must be approximately equal to 0. It follows $k_a^T q \approx k_b^T q >> k_i^T q$ for $i \notin \{a, b\}$. Let $q = w(k_a + k_b)$ for some $w >> 0$, then $k_a^T = k_b^T = w$ and $k_i^T = 0$ for $i \notin \{a, b\}$.

c)

1. Similarly to the question above, let $q = \beta(\mu_a + \mu_b)$ for $\beta >> 0$. Since α is vanishingly small, we can assume that $k_i \approx \mu_i$, because the probability density at $\mathcal{N}(x = \mu_i, \Sigma_i)$ us extremely high. We know that all μ_i's are orthogonal unit vectors, and the argument from the question above holds, so the final query is $q = w(\mu_a + \mu_b), w >> 0$.

2. We have

$$k_a \sim \mathcal{N}\left(\mu_a, \alpha I + \frac{1}{2}(\mu_a \mu_a^T)\right).$$

   For vanishingly small α we can assume $k_a \approx \varepsilon_a \mu_a$ for $\varepsilon_a \sim \mathcal{N}\left(1, \frac{1}{2}\right)$.

When $q = w(\mu_a + \mu_b), w >> 0$ we have

$$\begin{cases} k_i^T q \approx 0 & \text{for } i \notin \{a, b\} \\ k_a^T q \approx \varepsilon_a w & \text{for } i = a \\ k_b^T q \approx \varepsilon_b w & \text{for } i = b \end{cases} .$$

Then vector

$$c \approx \frac{\exp(\varepsilon_a w)}{\exp(\varepsilon_a w) + \exp(\varepsilon_b w)} v_a + \frac{\exp(\varepsilon_b w)}{\exp(\varepsilon_a w) + \exp(\varepsilon_b w)} v_b =$$

$$= \frac{1}{\exp((\varepsilon_b - \varepsilon_a)w) + 1} v_a + \frac{1}{\exp((\varepsilon_a - \varepsilon_b)w) + 1} v_b$$

Since $\varepsilon_a, \varepsilon_b \sim \mathcal{N}(1, \frac{1}{2})$, when $\varepsilon_a > \varepsilon_b$, c will be closer to v_a, in the opposite case, c will be closer to v_b. So the vector c will oscillate between v_a and v_b for different samples of $\{k_1, ..., k_n\}$.

d)

1. Similarly to the questions above let $q_1 = w\mu_a$ and $q_2 = w\mu_b, w >> 0$.
2. For most samples $c_1 \approx v_a$ and $c_2 \approx v_b$, and so c will look like

$$c = \frac{1}{2}(c_1 + c_2) \approx \frac{1}{2}(v_1 + v_2).$$

This is because even though there's variation in the value of k_a, for $k_a^T q_1 > 0$, as w grows, the attention will be concentrated on a. On the other hand, there's no variation in k_b and so c_2 will look like k_b without much variation.

e)

1. Vector c_2 approximates u_a. It's not possible for c_2 to approximate u_b by adding u_d or u_c to x_2. By doing so, we would only cause c_2 to incorporate either $u_d + u_b$ or $u_c + u_b$, but it's impossible to isolate u_b itself.
2. Let

$$V = \frac{u_b u_b^T - u_c u_c^T}{\beta^2}$$

then

$$v_1 = Vx_1 = \frac{(u_b u_b^T - u_c u_c^T)(u_d + u_b)}{\beta^2} = \frac{u_b u_b^T u_b}{\beta^2} = u_b$$

$$v_2 = Vx_2 = \frac{(u_b u_b^T - u_c u_c^T)u_a}{\beta^2} = 0$$

$$v_3 = Vx_3 = \frac{(u_b u_b^T - u_c u_c^T)(u_c + u_b)}{\beta^2} = \frac{u_b u_b^T u_b}{\beta^2} - \frac{u_c u_c^T u_c}{\beta^2} = u_b - u_c$$

To have $c_2 = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = \alpha_1 u_b + \alpha_3(u_b - u_c) \approx u_b$, we need $\alpha_1 \approx 1$ and $\alpha_3 \approx 0$. So $\exp(k_1^T q_2) >> \exp(k_3^T q_2)$. Similarly, to have $c_1 = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = \alpha_1 u_b + \alpha_3(u_b - u_c) \approx u_b - u_c$, we need $\alpha_1 \approx 0$ and $\alpha_3 \approx 1$. So $\exp(k_3^T q_1) >> \exp(k_1^T q_1)$. Let

$$K = u_a u_b^T + u_a u_d^T - u_c u_c^T$$

and

$$Q = u_a u_a^T - u_c u_b^T + u_c u_a^T$$

then

$$k_1^T q_2 = x_1^T K^T Q x_2 = (u_d + u_b)^T (u_b u_a^T + u_d u_a^T - u_c u_c^T)(u_a u_a^T - u_c u_b^T + u_c u_a^T) u_a =$$
$$= u_b^T u_b u_a^T u_a u_a^T u_a + u_d^T u_d u_a^T u_a u_a^T u_a = 2\beta^6$$
$$k_3^T q_2 = x_3^T K^T Q x_2 = (u_c + u_b)^T (u_b u_a^T + u_d u_a^T - u_c u_c^T)(u_a u_a^T - u_c u_b^T + u_c u_a^T) u_a =$$
$$= u_b^T u_b u_a^T u_a u_a^T u_a - u_c^T u_c u_c^T u_c u_a^T u_a = 0$$
$$k_3^T q_1 = x_3^T K^T Q x_1 = (u_c + u_b)^T (u_b u_a^T + u_d u_a^T - u_c u_c^T)(u_a u_a^T - u_c u_b^T + u_c u_a^T)(u_d + u_b) =$$
$$= u_c^T u_c u_c^T u_c u_b^T u_b = \beta^6$$
$$k_1^T q_1 = x_1^T K^T Q x_1 = (u_d + u_b)^T (u_a u_b^T + u_a u_d^T - u_c u_c^T)(u_a u_a^T - u_c u_b^T + u_c u_a^T)(u_d + u_b) = 0$$

So our both conditions are satisfied.

Part 3

d)

Obtained accuracy on the dev set: 0.8%
Obtained accuracy in london_baseline: 0.05%

f)

Obtained accuracy on the dev set: 6.2%

g)

1. Authors of Linformer architecture [2] show that the attention dot product matrix can be closely approximated by a matrix of much lower rank, which can be obtained by calculating the SVD of the key and value matrices and only using a fixed number of the largest singular values. Since SVD calculation also adds computational complexity, the authors replace it with a linear projection layer that serves to reduce the dimensionality.

2. The authors of the BigBird paper [3] propose sparse attention mechanism, that reduces quadratic dependency to linear. Sparse attention reduces computation time and the memory requirements of the attention mechanism by computing a limited selection of similarity scores from a sequence rather than all possible pairs, resulting in a sparse matrix rather than a full matrix. Thanks to that operation BigBird can process much longer sequences than popular BERT.

3. When sparse attention mechanisms are used in a standalone encoder (like BERT), they are Universal Approximators of sequence to sequence functions. Sparse encoder-decoder transformers are Turing Complete.

h)

1. Authors of the paper [1] introduce *Performers*, Transformer architectures which can estimate regular (softmax) full-rank-attention Transformers with provable accuracy, but using only linear (as opposed to quadratic) space and time complexity, without relying on any priors such as sparsity or low-rankness.

2. Applying random feature maps with potentially negative dimension-values (sin / cos) leads to unstable behaviours, especially when kernel scores close to 0 (which is the case for many entries of A corresponding to low relevance tokens) are approximated by estimators with large variance in such regions. This results in abnormal behaviours, e.g. negative-diagonal-values renormalizers D^{-1}, and consequently either completely prevents training or leads to sub-optimal models. [1]

3. Authors avoided the problem by changing trigonometric functions by exponential function (positive random feature map unbiased approximation). In the critical regions, where kernel values are small and need careful approximation, the new method outperforms naive approach.

4. Orthogonal random features maintain the marginal distributions of samples while enforcing that different samples are orthogonal. ORFs were introduced to reduce the variance of Monte Carlo estimators.

Part 4

a)

Pretrained model gained more information from corrupted span strategy.

b)

1. May cause bias or stereotype.

2. It gives misleading information (the answer looks valid even if it's incorrect).

c)

The model can generate answers by looking for people with similar name. This similarity has nothing to do with the real place of birth.

References

[1] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

[2] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[3] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. 2020.