Assignment 5

1. Attention exploration

a. Copying in attention .

i .

α can be interpreted as a categorical probability distribution because :

We can interpret the α_i as the probability that q ‘belongs to’ the class ‘i’ :

$P(L(q) = i) = \alpha_i, \sum \alpha_i = 1$

ii.

The categorical distribution α puts almost all of its weight on some α_j when $k_j^T.q >> k_i^T.q$ for all $i! = j$

iii.

When $\alpha_j >> \alpha_i$for all i != j $\Rightarrow$ α_j ~ 1 => c ~ v_j.

iv.

When k_j is similar to q, and other k_i virtually no corralates to q, c will ‘copy’ the value (v_j).

b. An average of two

i . c = $\frac{1}{2}(v_a + v_b)$. How to extract v_a from c ?

$s = v_a + v_b$

$M.s = v_a$

$M.(v_a + v_b) = v_a$

$M.v_a + M.v_b = v_a \qquad (*)$

M would solve (*) if

$M.v_a = v_a$, (1)

$M.v_b = 0.$ (2)

$v_a = (a_1 a_2 \dots a_m).(c_1 c_2 \dots c_m)^T$

(1) : $M.(a_1 a_2 \dots a_m).(c_1 c_2 \dots c_m)^T = v_a$

(2) : $M.(b_1 b_2 \dots b_m).(c_1 c_2 \dots c_m)^T = O_{mxm}$

Let $A = (a_1 a_2 \dots a_m)$, $B = (b_1 b_2 \dots b_m)$

$c = (c_1 c_2 \dots c_m)^T.$

We know that : $A^T.B = O_{mxm}.$

$A^T.A = I_{mxm}$

$v_a = A.c$

So $M = A.A^T$ would solve (*).

Showcase :

$M.s$

$= A.A^T.(v_a + v_b)$

$= A.A^T.v_a + A.A^T.v_b$

$= A.A^T.A.c + A.A^T.B.c$

$= A.c + A.O_{mxm}.c$

$= A.c$

$= v_a.$

1. Find an expression for a query vector q : c ½$(v_a + v_b)$

We need to have : $\alpha_a = \alpha_b =$ ½ and :$\alpha_i = 0$ for all $i != a, b.$ (*)

$\alpha_a = \alpha_b$

$= \dfrac{e^{k_a^T q}}{\sum \alpha_j + e^{k_a^T q} + e^{k_b^T q}}$

$\alpha_i = \dfrac{e^{k_i^T q}}{\sum \alpha_j + e^{k_a^T q} + e^{k_b^T q}}$

We can achieve (*) when $k_a^T q = k_b^T q >> k_i^T q$ for all $i! = a, b$.

We are given : all key vectors (k) are orthogonal and have norm 1, so we can choose $q = M(k_a + k_b)$, $M >> 0$.

c. Drawbacks of single-headed attention :

i.

$\alpha \to 0 \Rightarrow \sum_i \to 0$. As $k_i \sim N(u_i, \sum_i) \Rightarrow k_i \approx u_i \Rightarrow q = M(u_a + u_b)$

ii.

$k_a \approx [0.5u_a, 1, 5u_a]$. Assume $k_a = yu_a \Rightarrow y \sim N(1, 0.5)$

k_i ($i! = a$) $\approx u_i$

$q = M(u_a + u_b)$

$k_a^T q = yu_a^T q = yu_a^T.(u_a + u_b).M = y.M$

$k_b^T q = u_b^T.M.(u_a + u_b) = M$

$k_i^T q = u_i^T.M.(u_a + u_b) = 0$

$$\alpha_a = \frac{e^{k_a^T q}}{\sum e^{k_i^T q} + e^{k_a^T q} + e^{k_b^T q}} = \frac{1}{1 + e^{M(1-y)}}$$

$$\alpha_b = \frac{1}{1 + e^{M.(y-1)}}$$

$\alpha_i = 0$

We have : $c = \sum \alpha_i.v_i$

If $y \to 1.5 \Rightarrow \alpha_a \to 1$, $\alpha_b \to 0 \Rightarrow c = v_a$

If $y < 1 \Rightarrow \alpha_a \to 0$, $\alpha_b \to 1 \Rightarrow c = v_b$

In this case, c may put all weights in v_a or v_b

d. Benefits of multi-headed attention

i.

Design q_1, q_2 (two separate queries) : $c \approx \frac{1}{2}(v_a + v_b)$

Solution :

As $c = \frac{1}{2}(c_1 + c_2) \Rightarrow$ Find $q_1, q_2 : c_1 = v_a$ and $c_2 = v_b$ (**)

We can satisfy (**) if $\alpha_a - q_1 \rightarrow 1$, $\alpha_i - q_1 \rightarrow 0$ ($i! = a$) and $\alpha_b - q_2 \rightarrow 1$, $\alpha_i - q_2 \rightarrow 0$ ($i! = b$)

Thus, $q_1 = Mu_a$ and $q_2 = Mu_b$

ii.

When $\sum_a = \alpha I + \frac{1}{2}(u_a.u_a^T)$, if we have just one head, c may put all weights on v_a or v_b (When k_a varies a lot)

However, if we have two heads , even when there are some key values vary, we still got average through heads, and so not put all weights on any value vector.

2. Pretrained Transformer models and knowledge access

d. Make predictions (without pretraining)

Coding results :

Correct: 10.0 out of 500.0: 2.0%

f. Pretrain, finetune, and make predictions

Coding results :

Correct: 93.0 out of 500.0: 18.6%

g. Research ! Write and try out a more efficient variant of Attention

- Coding results : Using the Perceiver model

Correct: 25.0 out of 500.0: 5.0% (Had been improved : In UpProjectBlock, Scaling the embedding parameters * 6)

Scaling the Linear layers helps increase accuracy.

- Multi-headed attention :

Time complexity : $O(l^2d + dl^2)$

- Perceiver model :

Time complexity : $O(dm + Lm^2)$

$m << l$

3. Considerations in pretrained knowledge

1. The pretrained model achieved higher accuracy because it had prior knowledge through being trained on a large dataset. It learned features, patterns and then enabling it to generalize well on new examples. In contrast, the non-pretrained model takes longer time to learn the meaningful representations and scores worse.
2.

Concerns :

- Accuracy and Understandability : We can not distinguish correct information or fabricated one, model can produce results that we can not trust or rely on to make decisions
- Biases : Model can give biased answers when it does not learn the knowledge well, resulting biases when make decisions based on what it had learned.

3.

When we are given new names and query the model to predict their birthdates, the model just bases their predictions on pretrained knowledge. This can be problematic because it does not really rely on the fact but the data we gave and make predictions on the dataset.