Practical No. 4

Jakub Krajewski
gim.jakubk@gmail.com

30 kwietnia 2022

2. Attention exploration

a)

We want the output to be approximately equal to v_j. Therefore we want α_j to be close to 1 (and the rest of α_i to be close to 0). This will happen if the smaller product $k_j^T q$ is a large number and the rest of scalar products $k_i^T q$ are small numbers. We don't have to assume anything about values v_i. The necessary properties of k_i and q vectors are described above. In particular, for the scalar product to be large, we need the vectors to point in similar direction and have big norm.

b)

We can take $q = C(k_a + k_b)$ for some large number C. This way we will have $k_a^T q = k_b^T q = C$ and $k_i^T q = 0$ for $i \neq a, b$. Therefore after applying softmax function we will get coeffitients α_i such that the output will be approximately equal to the average of v_a and v_b.

c)

1. We can write a similar expression as in part b) - namely $q = C(\mu_a + \mu_b)$ for some large constant C. This will work because with such covariance matrix vectors are likely to be close to corresponding μ_i and we can assume for them to be perpendicular and of unit length.

2. Such covariance matrix will cause k_a and k_b to point in approximately the same direction as μ_a and μ_b, but with varying norm (length) of the vector. We will be getting c approximately equal to the "weighted average" $a \cdot v_a + b \cdot v_b$, but with varying coeffitients a and b. Qualitatively, we shouldn't expect to get the regular average, but different "mixes" of the vectors.

d)

1. We can choose q_1 to be equal to Ck_a for some large constant C. This way we will get c_1 approximately equal to v_a. Similarly, we can take $q_2 = Ck_b$. As an effect, we will get the final

output as desired.

2. This time the solution will work similarly as in point 1. Again, such covariance matrix will cause k_a and k_b to point in approximately the same direction as μ_a and μ_b, but with varying norm (length) of the vector. However, the exact norms of vectors are this time not important, as we are only interested for the direction (thanks to the constant C we will get the same result as in point 1. This happens because there is one head "corresponding" to each of the vectors and therefore we don't need to worry about their respective norm.

e)

1. x_2 is perpendicular to both x_1 and x_3. All of the vectors have large norm. Therefore, c_2 will approximate $x_2 = u_a$. It is not possible for c_2 to approximate u_b by adding u_d or u_c to x_2, because this way x_2 will still be perpendicular to u_b (and we can't "extract" u_b from any expression due to the zero scalar product).

2. Vectors u_a, u_b, u_c and u_d are orthogonal, therefore it is easy to see that vectors x_1, x_2, x_3 are linearly independent. We are therefore guaranteed, that there exists linear map A s. t. $A(x_1) = p_1, A(x_2) = p_2, A(x_3) = p_3$ for any vectors p_1, p_2, p_3. Each linear map has its corresponding matrix. Therefore it is sufficient in this exercise to specify values of linear maps V, Q, K on vectors x_1, x_2, x_3. In our case, we can simply choose

$$\begin{aligned} V(x_1) = u_b, V(x_2) = u_d, V(x_3) = u_b - u_c \\ Q(x_1) = u_a, Q(x_2) = u_b, Q(x_3) = u_c \\ K(x_1) = u_a, K(x_2) = u_b, K(x_3) = u_c. \end{aligned}$$

3. Pretraining Transformer-based Generative Model

d)

My model's accuracy on the dev set is 6 out of 500 (1.2%). The "London" approach is significantly better: we get 25 out of 500 correct.

f)

This time the dev accuracy is 64 out of 500 correct (12.8%).

g)

1. Authors of the Linformer have proven theoretically, that attention can be well approximated by an expression linear in size of the data. In this model self-attention mechanism is approximated by a low-rank matrix. As an effect, time and space complexity is reduced to $O(n)$.

2. Intuitively, in our data we expect each token to be related to only a subset of other tokens. Based on this observation, different types of "sparse" attention approaches have been proposed. The BigBird model combines several ways of connecting tokens to get computationally efficient, yet expressive model.

3. The relationship between tokens is sparse in nature. We only have to find out the proper subset of related tokens for each token. We can trust BigBird to do it reliably, as it combines sevreal sensible, different ways of searching for connections.

4. Pretrained Models as Source of Knowledge

a)

The pretrained model was able to "save" some information about the world through pretraining, that was accessed later during finetuning. The non-pretrained model did't have any source to look for the information it was asked for.

b)

1. Some mistakes/wrong decisions can be made if people trust the model returning wrong information.

2. People won't trust such a model: even if it returns the true output almost all the time, the users won't trust it as incorrect outputs are hard to distinguish from the true ones.

3. The model could for example output the most popular birth place that it has seen during training. This should raise concern, because because we can't expect unknown person to be born there (we don't know anything about the person and also the training set may be imbalanced in this regard).