CS 224n: Assignment 5 Written Part

May 9, 2022

Question 1: Attention Exploration (21 points)

(a) **Copying in attention:** Describe (in one sentence) what properties of the inputs to the attention operation would result in the output c being approximately equal to v_j for some j $\in \{1, \ldots, n\}$. Specifically, what must be true about the query q, the values $\{v_1, \ldots, v_n\}$ and/or the keys $\{k_1, \ldots, k_n\}$?

Solution:
When a k_j is very similar to q, the dot product of $k_j^T q$ is very large, then the exponential of the dot product is very large, and thus α_j is close to 1. Therefore, the output c will be approximately equal to v_j in this circumstance.

(b) **An average of two:** Give an expression for a query vector q such that the output c is approximately equal to the average of v_a and v_b, that is, $\frac{1}{2}(v_a + v_b)$.

Solution:
$q = t(k_a + k_b), t \gg 0$.
When q is a vector on the platform formed by k_a and k_b, other k_i is perpendicular to q, and thus α_i (for $i \neq a, b$) $= 0$. And therefore, the output c is approximately equal to the average of v_a and v_b.

(c) **Drawbacks of single-headed attention:**

i. Assume that the covariance matrices are $\sigma_i = \alpha I$, for vanishingly small α. Design a query q in terms of the μ_i such that as before, $c \approx \frac{1}{2}(v_a + v_b)$, and provide a brief argument as to why it works.

Solution:
Same as above. $q = t(\mu_a + \mu_b), t \gg 0$.

ii. Though single-headed attention is resistant to small perturbations in the keys, some types of larger perturbations may pose a bigger issue. Specifically, in some cases, one key vector k_a may be larger or smaller in norm than the others, while still pointing in the same direction as π_a. When you sample $\{k_1, \ldots, k_n\}$ multiple times, and use the q vector that you defined in part i., what qualitatively do you expect the vector c will look like for different samples?

Solution:
Because α is vanishingly small, $k_a \approx \varepsilon_a \mu_a, \varepsilon_a \sim \mathbb{N}(1, \frac{1}{2})$.
When $q = t(\mu_a + \mu_b), t \gg 0$, $k_i^T q \approx 0$ for $i \notin \{a, b\}$, $k_a^T q \approx t\varepsilon_a, k_b^T q \approx t\varepsilon_b$.

Therefore:

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

Since $\varepsilon_a, \varepsilon_b \sim \mathbb{N}(1, \frac{1}{2})$, when $\varepsilon_a > \varepsilon_b$, the output c is closer to v_a, and when $\varepsilon_a < \varepsilon_b$, the output c is closer to v_b.

(d) **Benefits of multi-headed attention:** Now we'll see some of the power of multi-headed attention.

i. Assume that the covariance matrices are $\Sigma_i = \alpha I$, for vanishingly small α. Design q_1 and q_2 such that c is approximately equal to $\frac{1}{2}(v_a + v_b)$.

Solution:

$$q_1 = t_1\mu_a$$
$$q_2 = t_2\mu_b$$
$$t_1, t_2 \gg 0$$

ii. Assume that the covariance matrices are $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a\mu_a^T)$ for vanishingly small α, and $\Sigma_i = \alpha I$ for all $i \neq a$. Take the query vectors q_1 and q_2 that you designed in part i. What, qualitatively, do you expect the output c to look like across different samples of the key vectors? Please briefly explain why. You can ignore cases in which $q_i^T k_a < 0$.

Solution:

$$k_a = \varepsilon_a\mu_a, \quad k_b = \varepsilon_b\mu_b, \quad \varepsilon_a, \varepsilon_b \sim \mathbb{N}(1, \frac{1}{2})$$
$$q_1 = t_1\mu_a, \quad q_2 = t_2\mu_b$$

Therefore,

$$c_1 = v_a \cdot \frac{\exp(\varepsilon_a t_1)}{\exp(\varepsilon_a t_1)+\exp(\varepsilon_b t_1 \mu_b \mu_a)} \approx v_a$$
$$c_2 = v_b \cdot \frac{\exp(\varepsilon_b t_2)}{\exp(\varepsilon_b t_2)+\exp(\varepsilon_a t_2 \mu_b \mu_a)} \approx v_b$$
$$c = \frac{1}{2}(c_1 + c_2) \approx \frac{1}{2}(v_a + v_b)$$

)

(e) **Key-Query-Value self-attention in neural networks:** In this question, we'll show how key-value-query attention like this allows the network to use different aspects of the input vectors xi in how it defines keys, queries, and values. Intuitively, this allows networks to choose different aspects of x_i to be the "content" (value vector) versus what it uses to determine "where to look" for content (keys and queries.)

i. First, consider if we didn't have key-query-value attention. If we perform self-attention with these vectors, what vector does c_2 approximate? Would it be possible for c_2 to approximate u_b by adding either u_d or u_c to x_2? Explain why or why not (either math or English is fine).

Solution:

$$q_2 = u_a$$

Because u_a, u_b, u_c, u_d are mutually orthogonal vectors in $\mathbb{R}^d$,

$$\exp(k_1^T q_2) = \exp(k_3^T q_2) \approx 0$$

$$\exp(k_2^T q_2) \approx 1$$

Therefore, $c_2 \approx u_a$.

ii. Now consider using key-query-value attention as we've defined it originally. Using the same definitions of x_1, x_2 and x_3 as in part i, specify matrices K,Q,V such that $c_2 \approx u_b$, and $c_1 \approx u_b - u_c$. There are many solutions to this problem, so it will be easier for you, if you first find V such that $v_1 = u_b$ and $v3 = u_b - u_c$, then work on Q and K. Some outer product properties may be helpful (as summarized in this footnote).

Solution:

$$V = (U_b^T U_b - U_c^T U_c) \cdot \frac{1}{\beta^2}$$

$$K = I$$

$$Q = (U_d U_a^T + U_c U_d^T) \cdot \frac{1}{\beta^2}$$

Proof:

$$v_1 = u_b, \quad v_2 = 0, \quad v_3 = u_b - u_c$$

$$k_1 = u_d + u_b, \quad k_2 = u_a, \quad k_3 = u_c + u_b$$

$$q_1 = u_c, \quad q_2 = u_d, \quad q_3 = 0$$

Therefore,

$$c_1 \approx v_3 = u_b - u_c$$

$$c_2 \approx v_1 = u_b$$

1 Question2: Pretrained Transformer models and knowledge access (35 points)

(a) None.

(b) None.

(c) None.

(d) Report your model's accuracy on the dev set (as printed by the second command above). As a reference point, we want to also calculate the accuracy the model would have achieved if it had just predicted "London" as the birth place for everyone in the dev set. Fill in london_baseline.py to calculate the accuracy of that approach and report your result in your write-up.

Solution:
The model's accuracy on the dev set is 0.8%.
London baseline accuracy is 5.0%.

(e) None.

(f) Report the accuracy on the dev set (printed by the third command above).

Solution:
The model's accuracy on the dev set is 21.2%.

(g) Report the accuracy of your synthesizer attention model on birth-place prediction on birth_dev.tsv after pretraining and fine-tuning.

i. **Solution:**
The accuracy is 10.78%.

ii. **Solution:**
The single layer synthesizer cannot capture the contextual information as the key-query-value self attention can.

2 Question3: Considerations in pretrained knowledge (5 points)

(a) Succinctly explain why the pretrained (vanilla) model was able to achieve an accuracy of above 10%, whereas the non-pretrained model was not.

Solution:
The pretrained model contains extra knowledge learned from the pretrained text.

(b) Come up with two reasons why this indeterminacy of model behavior may cause concern for such applications.

1. It may produce misleading information, e.g. create incorrect birth place that looks real.
2. It shows bias and stereotype.

(c) Concisely describe a strategy your model might take for predicting a birth place for that person's name, and one reason why this should cause concern for the use of such applications.

Solution:
The model will relate the unseen name to some closely related seen name, and output the birth place of the seen name. However, the similarity of names has no relationship with the birth places.