CS 224N: Assignment 5: Self-Attention, Transformers, and Pretraining

1. Attention exploration (21 points)

Multi-headed self-attention is the core modeling component of Transformers. In this question, we'll get some practice working with the self-attention equations, and motivate why multi-headed self-attention can be preferable to single-headed self-attention.

(a) (2 points) **Copying in attention:** Recall that attention can be viewed as an operation on a query $q \in \mathbb{R}^d$, a set of value vectors $\{v_1, ..., v_n\}$, $v_i \in \mathbb{R}^d$, and a set of key words $\{k_1, ..., k_n\}$, $k_i \in \mathbb{R}^d$, specified as follows:

$$c = \sum_{i=1}^{n} v_i \alpha_i$$

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

where α_i are frequently called the "attention weights", and the output $c \in \mathbb{R}^d$ is a correspondingly weighted average over the value vectors.

We'll first show that it's particularly simple for attention to "copy" a value vector to the output c. 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 $j \in \{1, ..., n\}$. Specifically, what must be true about the query q, the values $\{v_1, ..., v_n\}$ and/or the keys $\{k_1, ..., k_n\}$?

💡 **Answer:**

For $c \approx v_j$, α_j should be ≈ 1, while $\alpha_{i \neq j}$ should be ≈ 0. In other words, the dot product of q with k_j must be larger than $\sum_{i \neq j} \exp(k_i^T q)$, so that the probability mass is concentrated on j, and thus,

$$c = \sum_{i=1}^{n} \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp\left(k_j^T q\right)} v_i \approx v_j$$

Now,

$$\alpha_j = \frac{\exp(k_j^T q)}{\sum_{i=1}^{n} \exp\left(k_i^T q\right)}$$

Since $\alpha_j \approx 1$,

$$\sum_{i=1}^{n} \exp(k_i^T q) \approx \exp(k_j^T q)$$

$$\exp(k_i^T q) \approx 0 \text{ for } i \neq j$$

$$k_i^T q \approx -\infty$$

(b) (4 points) **An average of two:** Consider a set of key vectors $\{k_1, ..., k_n\}$ where all key vectors are perpendicular, that is $k_i \perp k_j$ for all $i \neq j$. Let $\|k_i\| = 1$ for all i. Let $\{v_1, ..., v_n\}$ be a set of arbitrary value vectors. Let $v_a, v_b \in \{v_1, ..., v_n\}$ be two of the value vectors. 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, $\frac{1}{2}(v_a + v_b)$. Note that you can reference the corresponding key vector of v_a and v_b as k_a and k_b.

Answer:

$c = \frac{v_a + v_b}{2} = \frac{1}{2}v_a + \frac{1}{2}v_b$. Comparing this with $c = \sum_{i=1}^{n} v_i \alpha_i$, we get,

$$\alpha_a = \frac{1}{2};\ \alpha_b = \frac{1}{2}$$

$$\exp(k_a^T q) = \exp(k_b^T q)$$

$$k_a^T q = k_b^T q$$

Let q be $\beta(k_a + k_b)$ where β is a large number acting as a scalar multiple.

Now, $\alpha_i = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)}$. Consider the numerator in this expression, $\exp(k_a^T q)$ where $i = a$. This can be written as,

$$\exp(k_a^T q) = \exp(k_a^T(\beta(k_a + k_b)))$$

$$= \exp(k_a^T k_a \beta + k_a^T k_b \beta)$$

Since $k_a^T k_a = 1$,

$$= \exp(\beta + k_a^T k_b \beta)$$

$$= \exp(\beta)$$
$$(\text{ since } \beta \text{ is any large number })$$

Similarly, $\exp(k_b^T q) = \exp(\beta)$.

Thus for all $j \in \{a, b\}$,

$$\exp(k_a^T q) = \exp(k_b^T q) = \exp(\beta)$$

and for all $j \notin \{a, b\}$, since the dot product similarity between the key and query vectors is simply average of v_a and v_b, as such, we have $\exp(k_j^T q) = \exp(0) = 1$. Thus,

$$\alpha_a = \alpha_b = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)} = \frac{\exp(\beta)}{(n-2) + 2\exp(\beta)}$$

which approaches $\frac{1}{2}$ as β grows and n is fixed.

This gives,

$$c = \alpha_a v_a + \alpha_b v_b = \frac{1}{2} v_a + \frac{1}{2} v_b = \frac{v_a + v_b}{2}$$

(c) (5 points) **Drawbacks of single-headed attention:** In the previous part, we saw how it was
possible for a single-headed attention to focus equally on two values. The same concept could easily be extended to any subset of values. In this question we'll see why it's not a *practical* solution. Consider a set of key vectors $\{k_1, ..., k_n\}$ that are now randomly sampled, $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means μ_i are known to you, but the covariances Σ_i are unknown. Further, assume that the means μ_i are all perpendicular; $\mu_i^T \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

i. (2 points) 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.

💡 **Answer:**

The setting is effectively identical to Question 1 (b). Let $q = \beta(k_a + k_b)$ for large
β. The variance being approximately 0 means $k_a \approx \mu_a$ and $k_b \approx \mu_b$, and the argument from Question 1 (b) holds.

ii. (3 points) 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 . As an example, let us consider a

covariance for item a as $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^T)$ for vanishingly small α (as shown in figure 1). Further, let $\Sigma_a = \alpha I$ for all $i \neq a$.

When you sample $\{k_1, ..., k_n\}$ 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?

Answer:

Upon sampling $\{k_1, ..., k_n\}$ multiple times, the key vectors would show large variance since one key vector may be larger or smaller in norm than the others. This would imply that some samples c would be weighted much more towards v_a and some towards v_b. This is because each sample k_a would be approximately some constant multiple of μ_a; some will be more like $\frac{1}{2}\mu_a$, some more like $\frac{3}{2}\mu_a$. This implies that the values of $\exp(k_a^T q)$ and $\exp(k_b^T q)$ corresponding to two keys k_a and k_b would not be equal; one or the other will be greater for each sample, and so c will be weighted more towards one or the other.

(d) (3 points) **Benefits of multi-headed attention:** Now we'll see some of the power of multi-headed attention. We'll consider a simple version of multi-headed attention which is identical to single-headed self-attention as we've presented it in this homework, except two query vectors (q_1 and q_2) are defined, which leads to a pair of vectors (c_1 and c_1), each the output of single-headed attention given its respective query vector. The final output of the multi-headed attention is their average, $\frac{1}{2}(c_1 + c_2)$. As in question 1(c), consider a set of key vectors $\{k_1, ..., k_n\}$ that are randomly sampled,

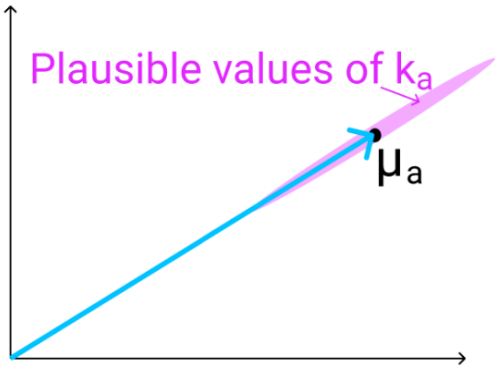

Figure 1: The vector μ_a (shown here in 2D as an example), with the range of possible values of k_a shown in red. As mentioned previously, k_a points in roughly the same direction as μ_a, but may have larger or smaller magnitude.

$k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means μ_i are known to you, but the covariances Σ_i are unknown. Also as before, assume that the means μ_i are mutually orthogonal; $\mu_i^T \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

i. (1 point) 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 $\frac{1}{2}(v_a + v_b)$.

> 💡 **Answer:**
>
> Let $q_1 = \beta \mu_a$ and let $q_2 = \beta \mu_b$, and the argument from single headed attention
> applies to each of the two cases.

ii. (2 points) 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 $q_i^T k_a < 0$.

💡 **Answer:**

For most samples, c_1 and c_2 will look like v_a and v_b, respectively, and so c will look
like $\frac{1}{2}(v_a + v_b)$, without much variation. This is because even though there's variation in the value of k_a, as long as $k_a^T q_1$ is positive, as β grows, the attention weight will be concentrated on a. Independently, there's no variation in k, and so c_2 will look like k without much variation.

(e) (7 points) **Key-Query-Value self-attention in neural networks:** So far, we've discussed attention as a function on a set of key vectors, a set of value vectors, and a query vector. In Transformers, we perform *self-attention*, which roughly means that we draw the keys, values, and queries from the same data. More precisely, let $\{x_1, ..., x_n\}$ be a sequence of vectors in $\mathbb{R}^d$. Think of each x_i as representing word i in a sentence. One form of self-attention defines keys, queries, and values as follows. Let $V, K, Q \in \mathbb{R}^{d \times d}$, be parameter matrices. Then

$$v_i = Vx_i \quad i \in \{1, ..., n\}$$

$$k_i = Kx_i \quad i \in \{1, ..., n\}$$

$$q_i = Qx_i \quad i \in \{1, ..., n\}$$

Then we get a context vector for each input i; we have $c_i = \sum_{j=1}^{n} \alpha_{ij} v_j$, where α_{ij} is defined as $\alpha_{ij} = \frac{\exp(k_j^T q_i)}{\sum_{\ell=1}^{n} \exp(k_\ell^T q_i)}$. Note that this is single-headed self-attention.

In this question, we'll show how key-value-query attention like this allows the network to use different aspects of the input vectors x_i 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. (3 points) First, consider if we didn't have key-query-value attention. For keys, queries, and values we'll just use x_i; that is, $v_i = q_i = k_i = x_i$. We'll consider a specific set of x_i. In particular, let u_a, u_b, u_c, u_d be mutually orthogonal vectors in $\mathbb{R}^d$, each with equal norm $\|u_a\| = \|u_b\| = \|u_c\| = \|u_d\| = \beta$, where β is very large. Now, let our x_i be:

$$x_1 = u_d + u_b$$

$$x_2 = u_a x_3 = u_c + u_b$$

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).

Answer:

c_2 approximates u_a. It is not possible for c_2 to approximate u_b just by adding u_d or u_c to x_2. Adding u_d or u_c to x_2 would cause c_2 to incorporate either $u_d + u_b$ or $u_c + u_b$, but it's impossible to isolate u_b itself.

ii. (4 points) 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 (and the graders),if you first find V such that $v_1 = u_b$ and $v_3 = u_b - u_c$, then work on Q and K. Some outer product properties may be helpful (as summarized in this footnote).

Answer:

Following the hint to start with $v_i = u_b$ and $v_3 = u_b - u_c$, we know that by the definition of key-query-value self-attention, $Vx_1 = v_1 = u_b$ and $Vx_3 = v_3 = u_b - u_c$. Using the properties of outer products (as given in the footnote), we can set $V = \beta^{-2} u_b u_b^T$ to make it true that $Vx_1 = u_b$. On an intuitive level, we must choose an outer product such that the first term is the vector we want (in v_1), while the second term is the vector we have (in x_1). Here is a more detailed explanation of why this works:

From eq. 3,

$$Vx_1 = v_1$$

Substituting,

$$\beta^{-2}(u_b u_b^T)(u_d + u_b) = u_b$$

Distributing terms,

$$\beta^{-2}\left[u_b(u_b^T u_d) + u_b(u_b^T u_b)\right] = u_b$$

Using the property of the dot product,

$$\beta^{-2}\left[0 + u_b\|u_b\|_2^2\right] = u_b$$

Using the definition of β,

$$\beta^{-2}\left[0 + u_b\beta^2\right] = u_b$$

Canceling β,

$$u_b = u_b$$

We can then add a $-u_c u_c^T$ term to V to cover the v_3 case as well. This yields $V = \beta^{-2}(u_b u_b^T - u_c u_c^T)$. This can be verified to work for v_3 as shown above.

Now that we have V, we can set about finding Q and K such that $c_1 \approx u_b - u_c$ and $c_2 \approx u_b$. To get $c_1 \approx v_3$, we must have $\alpha_{13} \approx 1$ and $\alpha_{1j\neq 3} \approx 0$. To accomplish this, $k_3^T q_1$ should be much larger than $k_{j\neq 1}^T q_2$. By similar reasoning, to get $c_2 \approx v_1$, $k_1^T q_2$ should be much larger than $k_{j\neq 1}^T q_2$. To make things simple, let us accomplish that by setting $k_1 = q_2$, $k_2 = 0$ and $k_3 = q_1$. We also know that $k_1 \neq k_3$. To satisfy these constraints, we'll need Q and K to pick out two unique terms from our x_1, x_2 and x_3. Looking at the structure of x_1, x_2 and x_3, we must use u_d and u_c. Let us pick $K = u_d u_d^T + u_c u_c^T$ and $Q = u_d u_a^T + u_c u_d^T$. It can be shown that these satisfy our constraints (though note that K and Q are interchangeable).

Thus we have $Q = u_d u_a^T + u_c u_d^T = \beta^{-2}(u_b u_b^T - u_c u_c^T), K = u_d u_d^T + u_c u_c^T$.

3. Considerations in pretrained knowledge (5 points)

(a) (1 point) Succinctly explain why the pretrained (vanilla) model was able to achieve an accuracy of above 10%, whereas the non-pretrained model was not.

Answer:

The pretrained model allows us to achieve better performance because it has learned both the low-level (semantic) and high-level (syntactic) features of language on a relatively larger dataset and then transfer learned the task at hand by fine-tuning on a smaller dataset. The non-pretrained model only has access to a small dataset and thus is limited in it's learning of language features beyond basic grammar/syntax/linguistic information, while the pretrained dataset contains specific world knowledge that the finetuning dataset lacks.

For instance, pretraining, with some probability, masks out the name of a person while providing the birth place, or masks out the birth place while providing the name - this teaches the model to associate the names with the birthplaces. At finetuning time, this information can be accessed, since it has been encoded in the parameters (which are initialized to the pretrained model's). Without pretraining, there's no way for the model to for the model to have any knowledge of the birth places of people that weren't in the finetuning training set, so it can't get above a simple heuristic baseline.

(b) (2 points) Take a look at some of the correct predictions of the pretrain+finetuned vanilla model, as well as some of the errors. We think you'll find that it's impossible to tell, just looking at the output, whether the model retrieved the correct birth place, or made up an incorrect birth place. Consider the implications of this for user-facing systems that involve pretrained NLP components. Come up with two reasons why this indeterminacy of model behavior may cause concern for such applications.

Answer:

There is a large space of possible reasons indeterminacy of model behavior may cause concern for user-facing systems that involve pretrained NLP components, such as:

1. Users will always get outputs that look valid (if the user doesn't know the real answer) and so won't be able to perform quality estimation themselves like one sometimes can when, e.g., a translation seems nonsensical. As such, system designers wouldn't have a way of filtering outputs for low-confidence predictions. This means we cannot even incorporate other techniques (say, another algorithm or simply, a human-in-the-loop) if the model's prediction is wrong since we cannot discern low-confidence predictions.
2. Models will not indicate that they simply do not know the birth place of a person (unlike a relational database or similar, which will return that the knowledge is not contained in it). This means the system cannot indicate a question is unanswerable. Since we cannot ascertain whether the model retrieved the correct birth place, or made up an incorrect birth place, we do not know when to trust and when not to trust the model. Once users realize the system can output plausible but incorrect answers, they may stop trusting the system, therefore making it useless.
3. Made up answers could be biased or offensive.
4. There is little avenue for recourse if users believe an answer is wrong, as it's impossible to determine the reasoning of the model is retrieving some gold standard knowledge (in which case the user's request to change the knowledge should be rejected), or just making up something (in which case the user's request to change the knowledge should be granted).
5. If the system makes downstream decisions on the basis of the model's output, the system can go wrong with its decisions

owing to a made up prediction that was incorrect. This would cause the system to do more harm than good.

(c) (2 points) If your model didn’t see a person’s name at pretraining time, and that person was not seen at fine-tuning time either, it is not possible for it to have “learned” where they lived. Yet, your model will produce something as a predicted birth place for that person’s name if asked. 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.

💡 **Answer:**

The model is probably doing a closest-similarity match in terms of features and is looking up the birth place for the target person using another person whose name bears the closest resemblance in terms of the said features. As such, the model could use character-level phonetic-like (sound-like information to make judgments about where a person was born based on how their name "sounds", likely leading to racist outputs. The model could also learn that certain names or types of names tend to be of people born in richer cities, leading to classist outputs that predict a birth place simply based on whether the names are like that of rich people or poorer people.

This would be a concern because this would lead to a form of "aliasing" where another person's birth place might be incorrectly assigned to the target person.