CS224N Assignment 4: Attention Exploration

1 Attention Exploration (14 points)

Multi-head 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.

Recall that attention can be viewed as an operation on a query vector $q \in \mathbb{R}^d$, a set of value vectors $\{v_1, \ldots, v_n\}, v_i \in \mathbb{R}^d$, and a set of key vectors $\{k_1, \ldots, k_n\}, k_i \in \mathbb{R}^d$, specified as follows:

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

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)} \tag{2}$$

with $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ termed the "attention weights". Observe that the output $c \in \mathbb{R}^d$ is an average over the value vectors weighted with respect to α.

1.1 (a) Copying in attention (3 points)

One advantage of attention is that it's particularly easy to "copy" a value vector to the output c. In this problem, we'll motivate why this is the case.

(i) **(2 points)** The distribution α is typically relatively "diffuse"; the probability mass is spread out between many different α_i. However, this is not always the case. Describe (in one sentence) under what conditions the categorical distribution α puts almost all of its weight on some α_j, where $j \in \{1, \ldots, n\}$ (i.e. $\alpha_j \gg \sum_{i \neq j} \alpha_i$). What must be true about the query q and/or the keys $\{k_1, \ldots, k_n\}$?

Answer: The categorical distribution α puts almost all its weight on some α_j when the query q has a much higher dot product with key k_j compared to all other keys (i.e., $k_j^\top q \gg k_i^\top q$ for all $i \neq j$), which occurs when q is strongly aligned with k_j in direction and/or magnitude.

(ii) **(1 point)** Under the conditions you gave in (i), describe the output c.

Answer: Under these conditions, since $\alpha_j \approx 1$ and $\alpha_i \approx 0$ for all $i \neq j$, the output $c \approx v_j$, effectively "copying" the value vector v_j to the output.

1.2 (b) An average of two (2 points)

Instead of focusing on just one vector v_j, a Transformer model might want to incorporate information from multiple source vectors. Consider the case where we instead want to incorporate

information from two vectors v_a and v_b, with corresponding key vectors k_a and k_b. Assume that (1) all key vectors are orthogonal, so $k_i^\top k_j = 0$ for all $i \neq j$; and (2) all key vectors have norm 1. Find an expression for a query vector q such that $c \approx \frac{1}{2}(v_a + v_b)$, and justify your answer.

Answer:

We can set $q = k_a + k_b$ (or any positive scalar multiple of it).

Justification: To achieve $c \approx \frac{1}{2}(v_a + v_b)$, we need $\alpha_a \approx \alpha_b \approx \frac{1}{2}$ and $\alpha_i \approx 0$ for all $i \neq a, b$. This requires that:

- $k_a^\top q = k_b^\top q$ (so both keys contribute equally)
- $k_a^\top q \gg k_i^\top q$ for all $i \neq a, b$ (so other attention weights are negligible)

With $q = k_a + k_b$:

$$\begin{aligned} k_a^\top q &= k_a^\top (k_a + k_b) = k_a^\top k_a + k_a^\top k_b = 1 + 0 = 1 \\ k_b^\top q &= k_b^\top (k_a + k_b) = k_b^\top k_a + k_b^\top k_b = 0 + 1 = 1 \\ k_i^\top q &= k_i^\top (k_a + k_b) = k_i^\top k_a + k_i^\top k_b = 0 + 0 = 0 \quad \text{for } i \neq a, b \end{aligned}$$

This gives us:

$$\alpha_a = \alpha_b = \frac{\exp(1)}{\exp(1) + \exp(1) + \sum_{i \neq a,b} \exp(0)} = \frac{e}{2e + (n-2)} \approx \frac{1}{2}$$

where the approximation holds because $e \approx 2.718 \gg 1$, so the denominator is dominated by $2e$ when n is not too large. Therefore, $c = \sum_i v_i \alpha_i \approx v_a \cdot \frac{1}{2} + v_b \cdot \frac{1}{2} = \frac{1}{2}(v_a + v_b)$.

1.3 (c) Drawbacks of single-headed attention (5 points)

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 $\{k_1, \ldots, k_n\}$ that are now randomly sampled, $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means $\mu_i \in \mathbb{R}^d$ are known to you, but the covariances Σ_i are unknown (unless specified otherwise in the question). Further, assume that the means μ_i are all perpendicular; $\mu_i^\top \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

(i) **(2 points)** Assume that the covariance matrices are $\Sigma_i = \alpha I, \forall i \in \{1, 2, \ldots, n\}$, 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: We can set $q = \mu_a + \mu_b$.

Justification: Since α is vanishingly small, the keys will be concentrated tightly around their means: $k_i \approx \mu_i$. Given that the means μ_i are orthogonal and have unit norm, this setup is essentially identical to part (b). Therefore:

$$\begin{aligned} k_a^\top q &\approx \mu_a^\top (\mu_a + \mu_b) = \mu_a^\top \mu_a + \mu_a^\top \mu_b = 1 + 0 = 1 \\ k_b^\top q &\approx \mu_b^\top (\mu_a + \mu_b) = \mu_b^\top \mu_a + \mu_b^\top \mu_b = 0 + 1 = 1 \\ k_i^\top q &\approx \mu_i^\top (\mu_a + \mu_b) = \mu_i^\top \mu_a + \mu_i^\top \mu_b = 0 + 0 = 0 \quad \text{for } i \neq a, b \end{aligned}$$

This gives approximately equal attention weights $\alpha_a \approx \alpha_b \approx \frac{1}{2}$ and negligible weights for other positions, yielding $c \approx \frac{1}{2}(v_a + v_b)$.

(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. 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 μ_a. As an example, let us consider a covariance for item a as $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ for vanishingly small α. This causes k_a to point in roughly the same direction as μ_a, but with large variances in magnitude. Further, let $\Sigma_i = \alpha I$ for all $i \neq a$.

When you sample $\{k_1, \ldots, k_n\}$ multiple times, and use the q vector that you defined in part (i), what do you expect the vector c will look like qualitatively for different samples? Think about how it differs from part (i) and how c's variance would be affected.

Answer: The vector c will exhibit high variance across different samples, and will not reliably approximate $\frac{1}{2}(v_a + v_b)$.

Explanation: With $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$, the key k_a has the form $k_a \approx \beta \mu_a$ for some scalar β that varies significantly across samples (due to the large variance $\frac{1}{2}$ along the μ_a direction), while $k_b \approx \mu_b$ remains tightly concentrated.

Using $q = \mu_a + \mu_b$ from part (i):

$$k_a^\top q \approx (\beta \mu_a)^\top (\mu_a + \mu_b) = \beta \mu_a^\top \mu_a + \beta \mu_a^\top \mu_b = \beta$$
$$k_b^\top q \approx \mu_b^\top (\mu_a + \mu_b) = 1$$

The attention weights depend on the ratio of $\exp(\beta)$ to $\exp(1)$:

- When $\beta \gg 1$ (large $\|k_a\|$): $\alpha_a \approx 1$ and $\alpha_b \approx 0$, so $c \approx v_a$
- When $\beta \ll 1$ (small $\|k_a\|$): $\alpha_a \approx 0$ and $\alpha_b \approx 1$, so $c \approx v_b$
- When $\beta \approx 1$: $\alpha_a \approx \alpha_b \approx \frac{1}{2}$, so $c \approx \frac{1}{2}(v_a + v_b)$

Unlike part (i) where c was consistently close to $\frac{1}{2}(v_a + v_b)$, here c varies dramatically between approximately v_a, v_b, or their average, depending on the random magnitude of k_a. This high variance in the output makes single-headed attention unreliable when key magnitudes are uncertain, motivating the need for multi-headed attention to handle such variability.

1.4 (d) Benefits of multi-headed attention (3 points)

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, except two query vectors (q_1 and q_2) are defined, which leads to a pair of vectors (c_1 and c_2), 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 $\{k_1, \ldots, k_n\}$ that are 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. Also as before, assume that the means μ_i are mutually orthogonal; $\mu_i^\top \mu_j = 0$ 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 in terms of μ_i such that c is approximately equal to $\frac{1}{2}(v_a + v_b)$. Note that q_1 and q_2 should have different expressions.

Answer: Set $q_1 = \mu_a$ and $q_2 = \mu_b$.

Justification: With $q_1 = \mu_a$, since keys are tightly concentrated around their means ($k_i \approx \mu_i$) and the means are orthogonal with unit norm:

$$k_a^\top q_1 \approx \mu_a^\top \mu_a = 1$$
$$k_i^\top q_1 \approx \mu_i^\top \mu_a = 0 \quad \text{for } i \neq a$$

This gives $c_1 \approx v_a$.

Similarly, with $q_2 = \mu_b$:

$$k_b^\top q_2 \approx \mu_b^\top \mu_b = 1$$
$$k_i^\top q_2 \approx \mu_i^\top \mu_b = 0 \quad \text{for } i \neq b$$

This gives $c_2 \approx v_b$.

Therefore, the final output is $c = \frac{1}{2}(c_1 + c_2) \approx \frac{1}{2}(v_a + v_b)$.

(ii) **(2 points)** Assume that the covariance matrices are $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ 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? Explain briefly in terms of variance in c_1 and c_2. You can ignore cases in which $k_a^\top q_i < 0$.

Answer: The output c will have much lower variance compared to the single-headed case in part (c)(ii), and will consistently approximate $\frac{1}{2}(v_a + v_b)$ across different samples.

Explanation: With $k_a \approx \beta \mu_a$ where β varies significantly, and using $q_1 = \mu_a$ and $q_2 = \mu_b$:

For head 1 ($q_1 = \mu_a$):

$$k_a^\top q_1 \approx (\beta \mu_a)^\top \mu_a = \beta$$
$$k_b^\top q_1 \approx \mu_b^\top \mu_a = 0$$

This head exhibits high variance: when β is large, $c_1 \approx v_a$; when β is small, c_1 is more diffuse but still weighted toward v_a.

For head 2 ($q_2 = \mu_b$):

$$k_a^\top q_2 \approx (\beta \mu_a)^\top \mu_b = 0$$
$$k_b^\top q_2 \approx \mu_b^\top \mu_b = 1$$

This head has low variance and consistently produces $c_2 \approx v_b$, regardless of β.

The final output $c = \frac{1}{2}(c_1 + c_2)$ benefits from the stability of head 2:

- When β is large: $c \approx \frac{1}{2}(v_a + v_b)$
- When β is small: $c \approx \frac{1}{2}(c_1 + v_b)$ where c_1 contributes partially, but v_b is reliably included

The key insight is that head 2 is *unaffected* by the variance in k_a because q_2 is orthogonal to μ_a. This stabilizes the overall output, dramatically reducing variance compared to single-headed attention.

1.5 (e) Summary (1 point)

Based on part (d), briefly summarize how multi-headed attention overcomes the drawbacks of single-headed attention that you identified in part (c).

Answer: Multi-headed attention overcomes the drawbacks of single-headed attention by using multiple specialized query vectors that focus on different aspects of the input. When one head's attention weights become unreliable due to variance in key magnitudes (as in part (c)(ii)), other heads remain stable because their queries are orthogonal to the problematic directions. By averaging the outputs from multiple heads, the model reduces overall variance and produces more consistent, reliable outputs. Each head can specialize in attending to specific positions or features, and their combined output is robust to perturbations that would destabilize a single-headed attention mechanism.

2 Position Embeddings Exploration (6 points)

Position embeddings are an important component of the Transformer architecture, allowing the model to differentiate between tokens based on their position in the sequence. In this question, we'll explore the need for positional embeddings in Transformers and how they can be designed.

Recall that the crucial components of the Transformer architecture are the self-attention layer and the feed-forward neural network layer. Given an input tensor $X \in \mathbb{R}^{T\times d}$, where T is the sequence length and d is the hidden dimension, the self-attention layer computes the following:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$
$$H = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$ are weight matrices, and $H \in \mathbb{R}^{T\times d}$ is the output.

Next, the feed-forward layer applies the following transformation:

$$Z = \text{ReLU}(HW_1 + \mathbf{1}\cdot b_1)W_2 + \mathbf{1}\cdot b_2$$

where $W_1, W_2 \in \mathbb{R}^{d\times d}$ and $b_1, b_2 \in \mathbb{R}^{1\times d}$ are weights and biases; $\mathbf{1} \in \mathbb{R}^{T\times 1}$ is a vector of ones; and $Z \in \mathbb{R}^{T\times d}$ is the final output.

2.1 (a) Permuting the input (4 points)

(i) **(3 points)** Suppose we permute the input sequence X such that the tokens are shuffled randomly. This can be represented as multiplication by a permutation matrix $P \in \mathbb{R}^{T\times T}$, i.e., $X_{\text{perm}} = PX$. Show that the output Z_{perm} for the permuted input X_{perm} will be $Z_{\text{perm}} = PZ$.

You are given that for any permutation matrix P and any matrix A, the following hold:

$$\text{softmax}(PAP^\top) = P\,\text{softmax}(A)\,P^\top \quad \text{and} \quad \text{ReLU}(PA) = P\,\text{ReLU}(A).$$

Proof:

Step 1: Self-attention with permuted input.

Given $X_{\text{perm}} = PX$, we compute:

$$\begin{aligned} Q_{\text{perm}} &= X_{\text{perm}} W_Q = (PX)W_Q = P(XW_Q) = PQ \\ K_{\text{perm}} &= X_{\text{perm}} W_K = (PX)W_K = P(XW_K) = PK \\ V_{\text{perm}} &= X_{\text{perm}} W_V = (PX)W_V = P(XW_V) = PV \end{aligned}$$

Next, we compute the attention scores:

$$Q_{\text{perm}} K_{\text{perm}}^\top = (PQ)(PK)^\top = PQ(K^\top P^\top) = P(QK^\top)P^\top$$

Applying softmax (using the given property):

$$\begin{aligned} \text{softmax}\left(\frac{Q_{\text{perm}} K_{\text{perm}}^\top}{\sqrt{d}}\right) &= \text{softmax}\left(\frac{P(QK^\top)P^\top}{\sqrt{d}}\right) \\ &= P\,\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)P^\top \end{aligned}$$

Computing the output of self-attention:

$$\begin{aligned} H_{\text{perm}} &= \text{softmax}\left(\frac{Q_{\text{perm}} K_{\text{perm}}^\top}{\sqrt{d}}\right) V_{\text{perm}} \\ &= \left[P\,\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)P^\top\right](PV) \\ &= P\,\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)(P^\top P)V \end{aligned}$$

Since P is a permutation matrix, $P^\top P = I$, so:

$$H_{\text{perm}} = P\,\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V = PH$$

Step 2: Feed-forward layer with permuted attention output.

Given $H_{\text{perm}} = PH$, we compute the feed-forward transformation. Note that for a permutation matrix, $P\mathbf{1} = \mathbf{1}$ (permuting rows of ones gives ones):

$$\begin{aligned} H_{\text{perm}} W_1 + \mathbf{1} \cdot b_1 &= (PH)W_1 + \mathbf{1} \cdot b_1 \\ &= P(HW_1) + (P\mathbf{1}) \cdot b_1 \\ &= P(HW_1 + \mathbf{1} \cdot b_1) \end{aligned}$$

Applying ReLU (using the given property):

$$\begin{aligned} \text{ReLU}(H_{\text{perm}} W_1 + \mathbf{1} \cdot b_1) &= \text{ReLU}(P(HW_1 + \mathbf{1} \cdot b_1)) \\ &= P\,\text{ReLU}(HW_1 + \mathbf{1} \cdot b_1) \end{aligned}$$

Finally:

$$\begin{aligned} Z_{\text{perm}} &= \text{ReLU}(H_{\text{perm}}W_1 + \mathbf{1} \cdot b_1)W_2 + \mathbf{1} \cdot b_2 \\ &= [P\,\text{ReLU}(HW_1 + \mathbf{1} \cdot b_1)]W_2 + \mathbf{1} \cdot b_2 \\ &= P[\text{ReLU}(HW_1 + \mathbf{1} \cdot b_1)W_2] + (P\mathbf{1}) \cdot b_2 \\ &= P[\text{ReLU}(HW_1 + \mathbf{1} \cdot b_1)W_2 + \mathbf{1} \cdot b_2] \\ &= PZ \end{aligned}$$

Therefore, $Z_{\text{perm}} = PZ$, as required.

(ii) **(1 point)** Think about the implications of the result you derived in part (i). Explain why this property of the Transformer model could be problematic when processing text.

Answer: The result shows that the Transformer architecture (without position embeddings) is *permutation-equivariant*: if we permute the input tokens, the output tokens are permuted in exactly the same way. This means the model treats the input as an unordered set rather than an ordered sequence. This is highly problematic for text processing because word order is crucial for meaning in natural language. For example, "dog bites man" and "man bites dog" have completely different meanings despite containing the same words, but a permutation-equivariant model would produce outputs that differ only by the same permutation, failing to capture the semantic difference. Position embeddings are therefore essential to break this symmetry and allow the model to distinguish between different orderings of tokens.

2.2 (b) Position embeddings design (2 points)

Position embeddings are vectors that encode the position of each token in the sequence. They are added to the input word embeddings before feeding them into the Transformer. One approach is to generate position embeddings using a fixed function of the position and the dimension of the embedding. If the input word embeddings are $X \in \mathbb{R}^{T \times d}$, the position embeddings $\Phi \in \mathbb{R}^{T \times d}$ are generated as follows:

$$\begin{aligned} \Phi(t, 2i) &= \sin\left(\frac{t}{10000^{2i/d}}\right) \\ \Phi(t, 2i+1) &= \cos\left(\frac{t}{10000^{2i/d}}\right) \end{aligned}$$

where $t \in \{0, 1, \ldots, T-1\}$ and $i \in \{0, 1, \ldots, d/2 - 1\}$.
Specifically, the position embeddings are added to the input word embeddings:

$$X_{\text{pos}} = X + \Phi$$

(i) **(1 point)** Do you think the position embeddings will help the issue you identified in part (a)? If yes, explain how and if not, explain why not.

Answer: Yes, position embeddings will help resolve the permutation-equivariance issue.

Explanation: When we add position embeddings to the input, we get $X_{\text{pos}} = X + \Phi$. If we now permute the input tokens with permutation matrix P, the word embeddings get permuted as PX, but the position embeddings Φ remain fixed in their original positions

because they are defined as functions of absolute positions $t \in \{0, 1, \ldots, T-1\}$, not tied to the tokens themselves.

Therefore, after permutation, the model receives:

$$X_{\text{perm, pos}} = PX + \Phi \neq P(X + \Phi) = PX_{\text{pos}}$$

This breaks the permutation-equivariance property. The position embeddings act as fixed positional "anchors" that tell the model where each token appears in the sequence, regardless of which token occupies that position. Thus, different orderings of the same tokens will produce genuinely different representations and outputs, allowing the model to distinguish "dog bites man" from "man bites dog."

(ii) **(1 point)** Can the position embeddings for two different tokens in the input sequence be the same? If yes, provide an example. If not, explain why not.

Answer: Theoretically yes, but it is extremely unlikely for practical sequence lengths.

Explanation: For two different positions $t_1 \neq t_2$ to have identical position embeddings, we would need:

$$\sin\left(\frac{t_1}{10000^{2i/d}}\right) = \sin\left(\frac{t_2}{10000^{2i/d}}\right) \quad \forall i \in \{0, \ldots, d/2-1\}$$
$$\cos\left(\frac{t_1}{10000^{2i/d}}\right) = \cos\left(\frac{t_2}{10000^{2i/d}}\right) \quad \forall i \in \{0, \ldots, d/2-1\}$$

This requires $\frac{t_1}{10000^{2i/d}} - \frac{t_2}{10000^{2i/d}} = 2\pi k_i$ for some integer k_i and for all i simultaneously, due to the periodicity of sine and cosine functions.

The different frequency components (different values of i) have vastly different periods:

- Low frequencies (small i): very long periods, rarely repeat
- High frequencies (large i): shorter periods, may repeat for large t

For all dimensions to align simultaneously would require $|t_1 - t_2|$ to be on the order of tens of thousands or more, far exceeding typical sequence lengths used in practice (usually $T < 10000$). Therefore, while theoretically possible due to periodicity, position embeddings are effectively unique for all practical purposes in standard Transformer applications.