Homework 2: Self-Attention and Transformers

CS 388: Natural Language Processing • Fall 2026

Instructions. This assignment can be done entirely by hand – no coding required (Question 4 asks you to read a short PyTorch file and describe a fix, but you never have to run anything); however, please transcribe your answers/work into the provided LaTeXfile and submit the PDF, rather than a handwritten response. **Please use the `\B{}` command to wrap your solutions, making them blue (this will help us when grading).** Show your work. When you are asked to construct a matrix, write out every entry, and give each entry as a number. Submit a single PDF to Gradescope.

Setup and Notation

Our alphabet is $\Sigma = \{\texttt{A}, \texttt{B}, \texttt{C}\}$, embedded as one-hot vectors in $\mathbb{R}^3$:

$$e(\texttt{A}) = [1, 0, 0], \qquad e(\texttt{B}) = [0, 1, 0], \qquad e(\texttt{C}) = [0, 0, 1].$$

All vectors are row vectors. A sequence is a string $x = x_1 x_2 \cdots x_n$ with each $x_i \in \Sigma$; we write $X \in \mathbb{R}^{n \times 3}$ for the matrix whose i-th row is $e(x_i)$. Throughout this assignment every sequence has length at most $L = 5$.

Model. We will use a single layer of scaled dot-product self-attention, with parameter matrices W_Q, W_K, and W_V, each in $\mathbb{R}^{3\times 3}$. For each position i we form a query, a key, and a value,

$$q_i = e(x_i)W_Q, \qquad k_i = e(x_i)W_K, \qquad v_i = e(x_i)W_V,$$

and the attention output at position i is $\text{softmax}\big(q_i K^\top\big)\, V$. Real implementations divide the scores by $\sqrt{d}$ to stop them from growing with the embedding dimension; here d is tiny and we are choosing the weights by hand, so we drop that constant — it would only rescale W_Q.

We only ever care about the output at the *final* position, which is the position that has read the whole string. So write $q = q_n = e(x_n)W_Q$, let

$$s_i = q\, k_i^\top, \qquad \alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)}, \qquad o \;=\; \sum_{i=1}^{n} \alpha_i v_i \;\in\; \mathbb{R}^3,$$

so s_i is the score of position i, α is the attention distribution, and o is the output vector. Note that position n attends to itself: the sum runs over all of $1, \ldots, n$, including $i = n$.

Inference. We read the three coordinates of o off as scores for the three symbols, and the model's prediction is

$$\hat{y}(x) \;=\; \arg\max_{\ell \in \{\texttt{A},\texttt{B},\texttt{C}\}} \; o_\ell.$$

Initially, we will not have any output projection or MLP (until Question 2, where we add an output projection). A tie for the argmax counts as a wrong answer. A rule is *captured* by a choice of parameters if $\hat{y}(x)$ is correct for *every* sequence x of length $n \le L$ to which the rule applies.

Simplification. In Questions 1 and 2 we will fix $W_K = W_V = I_3$, so that $k_i = v_i = e(x_i)$, leaving W_Q as the only parameter for you to design. Under this choice, if x_n is the a-th symbol of the alphabet and x_i is the b-th, then

$$s_i \;=\; e(x_n)\, W_Q\, e(x_i)^\top \;=\; (W_Q)_{ab},$$

i.e. W_Q is nothing more than a 3×3 lookup table of scores: the entry *is* the score, indexed by (final symbol, attended symbol). Feel free to use this fact without re-deriving it.

Output. Because $W_V = I_3$, the values are the embeddings themselves, so $o = \sum_i \alpha_i e(x_i)$, and the ℓ-th coordinate of o is just the total attention mass placed on the positions holding ℓ:

$$o_\ell \;=\; \sum_{i\,:\,x_i=\ell} \alpha_i.$$

In particular the coordinates of o are nonnegative and sum to 1, so o is a probability distribution over Σ. You may use both of these facts without proof.

1. Question 1 (25 points)

Consider sequences over $\{\texttt{A}, \texttt{B}\}$ only, and the following pair of decision rules (which we saw in class):

R1	if the sequence is all `A`'s, predict `A`
R2	if the sequence contains at least one `B`, predict `B`

Together these cover every sequence over $\{\texttt{A}, \texttt{B}\}$.

(a) **(15 pts)** Write down a matrix W_Q that captures R1 and R2, giving all nine entries.

(b) **(10 pts)** Verify your answer on the three sequences `AAA`, `AAB`, and `ABA` by computing s_i, α_i, and o for each.

2. Question 2 (40 points)

Now let all three symbols appear, and consider the following rule set, which extends the previous one:

	contains `B`?	contains `C`?	prediction
R1	no	no	`A`
R2	yes	no	`B`
R3	no	yes	`A`
R4	yes	yes	`C`

In other words, we predict `B` if a sequence contains a `B` and no `C`; we predict `C` if and only if it contains both a `B` and a `C`; and otherwise we predict `A`. The rules still cover every sequence, and we still fix $W_K = W_V = I_3$.

(a) **(12 pts)** Show that no single W_Q captures R1–R4. *Hint:* both `CA` and `BCA` end in the same symbol, so they use the same row of W_Q, and they contain the same number of `A`'s and the same number of `C`'s. You should be able to write out two incompatible inequalities that these rules require.

(b) **(6 pts)** Explain in two or three sentences *why* this happens, in terms of how the scores s_i are computed. Your explanation should make clear what property of the rule set a single head cannot represent, and should not depend on the particular counterexample from (a).

(c) **(16 pts)** Still assuming $W_K = W_V = I_3$, show that multi-head attention with two heads — that is, two separate query matrices — can capture the rules. Heads 1 and 2 should produce

outputs $o^{(1)}, o^{(2)} \in \mathbb{R}^3$. These are concatenated into $[o^{(1)}; o^{(2)}] \in \mathbb{R}^6$ and passed through a linear output layer

$$z \;=\; [o^{(1)}; o^{(2)}]\, W_O, \qquad W_O \in \mathbb{R}^{6\times 3}\,,$$

with the prediction now $\hat{y}(x) = \arg\max_\ell z_\ell$. Design $W_Q^{(1)}$, $W_Q^{(2)}$, and W_O so that some coordinate of $o^{(1)}$ behaves as an (approximate) indicator of "x contains `B`" and some coordinate of $o^{(2)}$ as an indicator of "x contains `C`". Give the matrices and identify the two coordinates. Remember that your construction has to be correct for every sequence of length at most $L = 5$.

(d) **(6 pts)** Why can the rules above be captured by a model with no non-linearity in the output layer?

3. Question 3 (20 points)

For this part, restrict the alphabet to $\{\texttt{A}, \texttt{C}\}$ and consider a single rule:

R5	if the first symbol is `C`, predict `C`; otherwise predict `A`

(a) **(5 pts)** Can single-head or multi-head attention, as we have defined it so far, capture this rule? Why or why not? (No proof needed here, 1–2 sentences of explanation is fine.)

(b) **(15 pts)** Fix the model by hand with an integer positional encoding. Embed position i by appending its index, so that token i is represented by the 4-dimensional vector

$$\tilde{e}_i \;=\; [\, e(x_i),\; i\,] \;\in\; \mathbb{R}^4,$$

take $W_Q, W_K, W_V \in \mathbb{R}^{4\times 4}$, and let the prediction be the argmax over the first three coordinates of o as before. Give W_Q, W_K, W_V explicitly (all sixteen entries of each) such that R5 is captured for all sequences of length at most $L = 5$.

4. Question 4 (15 points)

The file `mhsa.py` accompanying this assignment implements a single layer of multi-head self-attention in PyTorch. The file contains a mistake.

(a) **(5 pts)** Find the mistake. Point to the incorrect line, and say in one sentence what it does instead of what it was supposed to do.

(b) **(5 pts)** What happens if the mistake is not fixed? Describe what you would see if you ran the file as given, relating it to Question 2.

(c) **(5 pts)** Give a one-line fix: the single line of code that should replace the offending line.

What to turn in. A single PDF containing every matrix you were asked to construct, written out entry by entry; the worked verifications requested in Question 1(b); the impossibility argument in Question 2(a); and your written answers to the discussion questions. Do not submit code – write the one-line fix from Question 4(c) directly into your PDF.