Quiz 5

601.471/671 NLP: Self-Supervised Models

Johns Hopkins University Fall 2026

Name: _____ **Student version**

Total: 100 required points + 15 optional bonus points.

Time: 20 minutes.

Coverage: normalization, KV caching, and programming with padded-batch losses. Write short answers and show the requested calculations. One or two sentences suffice for each explanation. No calculator is needed. Bonus parts are optional: leaving them blank costs no required points; bonus scores are added beyond 100. The time limit includes bonus work.

Q1. Normalization *40 + 5 bonus points*

(a) **20 required points + 5 bonus points.** Consider two sublayer layouts, where N is LayerNorm and S is a differentiable sublayer. Both map a token vector x to a vector of the same width:

$$\text{A: } y = N(x + S(x)), \qquad \text{B: } y = x + S(N(x)).$$

(i) **4 points.** Label A and B as pre-norm or post-norm.

Your answer:

(ii) **6 points.** For layout A, set $r = x + S(x)$, so $y = N(r)$. Assume the local Jacobians $J_S(u) = \partial S(u)/\partial u$ and $J_N(u) = \partial N(u)/\partial u$ are given; rows index outputs and columns index inputs. Use column gradients and leave these Jacobians symbolic. Given $g_y = \nabla_y \ell$ for a scalar loss ℓ, what gradient $g_r = \nabla_r \ell$ reaches r through N?

Your answer:

(iii) **6 points.** Still in A, use g_r from (ii) to write the two contributions to $g_x = \nabla_x \ell$: one through the direct x term in r, and one through $S(x)$. Add them to obtain g_x.

Your answer:

(iv) **Optional bonus: 5 points.** For layout B, set $n = N(x)$, so $y = x + S(n)$. First write $g_n = \nabla_n \ell$ in terms of g_y (2 bonus points). Then combine the direct residual contribution and the contribution through N to obtain g_x (3 bonus points). Indicate the Jacobians' evaluation inputs.

Your answer:

(v) **4 points.** Which layout has a direct residual path outside normalization, and what contribution does it make to g_x? Answer from the block formulas; the bonus derivation is not required.

Your answer:

Q1. Normalization (continued)

(b) **20 points.** Compare affine LayerNorm and bias-free RMSNorm on one token $x \in \mathbb{R}^D$. Normalize over its D features, using population variance and fixed ϵ.

(i) **8 points.** Write the formula for output coordinate j under each normalization (4 points each). Define any input statistics and learned parameters in your formulas.

Your answer:

(ii) **4 points.** Give each layer's learnable scalar count in terms of D (2 points each).

Your answer:

(iii) **4 points.** Now set $D = 2$, $x = [0, 2]$, and $\epsilon = 0$. Set all learned gains to 1 and all LayerNorm biases to 0. Compute this vector's feature mean, population variance, and mean of squared features.

Your answer:

(iv) **2 points.** Using the statistics from (iii), compute $\mathrm{LayerNorm}(x)$.

Your answer:

(v) **2 points.** Using the same settings, compute $\mathrm{RMSNorm}(x)$. Exact square roots are acceptable.

Your answer:

Q2. KV Caching *20 + 5 bonus points*

(a) **20 required points + 5 bonus points. KV caching.** A decoder generates one token at a time with fixed parameters and no dropout. In every layer, attention permits each position to read only itself and earlier positions; all other operations act separately on each position. Assume earlier token embeddings and position assignments do not change.

(i) **4 points.** Which two kinds of projected vectors from earlier tokens are stored in a KV cache?

Your answer:

(ii) **4 points.** When a new token is appended, can an earlier position attend to it under the stated attention restriction? Explain.

Your answer:

(iii) **Optional bonus: 5 points.** Using (ii) and the fixed-input/parameter assumptions, explain why earlier cached vectors can be reused at every layer.

Your answer:

(iv) **6 points.** One cached tensor has shape $[2, 2, 8, 4]$ (batch, head, cached position, head width). How many scalar entries does it contain? Show the product.

Your answer:

(v) **6 points.** Each of 3 layers caches two tensors of the shape in (iv). At 2 bytes per scalar, how many bytes do all these tensors occupy? Use your count from (iv); exclude all other memory.

Your answer:

Q3. Programming: Loss for a Padded Batch *40 + 5 bonus points*

The code below should compute the **mean next-token cross-entropy over non-PAD targets**. There are **two independent bugs**. Parts (i)–(iii) are required; (iv) is optional bonus work. No full model implementation is needed.

Use these assumptions:

- `F` is the imported alias for `torch.nn.functional`. `logits` contains raw floating-point scores of shape $[B, T, V]$; `targets` is an integer `torch.long` tensor of shape $[B, T]$. Axes are batch, position, and vocabulary class. Devices and dtypes are compatible; arithmetic is well behaved.
- $B = 2$, $T = 4$, $V = 3$, and `pad_id = 0`. The target rows are `[1, 2, 0, 0]` and `[2, 1, 2, 1]`; ID 0 marks padding. Logits and targets are already aligned for next-token prediction.
- All model computations and attention masks are correct.

API reference: `F.cross_entropy` (selected arguments)

- `input`: raw, unnormalized class scores (logits); for this question, shape $[N, V]$.
- `target`: integer class IDs, shape $[N]$.
- `ignore_index`: target ID excluded from the loss; its positions have zero loss in the unreduced output.
- `reduction="none"`: returns one loss per target position, without summing or averaging.

Here N is the number of flattened target positions and V the class count.
Adapted from PyTorch 2.14 documentation; class-index targets, default class weights and label smoothing.

```
1 probabilities = logits.softmax(dim=-1)
2 token_losses = F.cross_entropy(
3     probabilities.reshape(-1, V),
4     targets.reshape(-1),
5     ignore_index=pad_id, reduction="none",
6 ).reshape(B, T)
7 loss = token_losses.mean()
```

(i) **8 points. Count positions.** How many total target positions are in this batch, including PAD (4 points)? How many are non-PAD (4 points)?

Your answer:

(ii) **12 points. Check the averaging.** For this part only, use `token_losses` $= \begin{bmatrix} 1 & 3 & 0 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$ (values supplied). Compute (a) line 7's result (6 points) and (b) the mean over non-PAD targets (6 points). Show each denominator.

Your answer:

(iii) **20 points. Check the cross-entropy input.** Which line supplies the wrong kind of class scores? Give its line number, a correction, and a one-sentence reason. Assume the averaging bug is fixed.

Your answer:

(iv) **Optional bonus: 5 points. Fix the averaging.** Replace line 7 using `token_losses` and `targets` to average only non-PAD targets in any batch with at least one valid target. Explain why zero PAD losses do not make the original `mean()` correct. Assume the input bug is fixed.

Your answer: