

# Quiz 4

## 601.471/671 NLP: Self-Supervised Models

Johns Hopkins University Fall 2026

Name: \_\_\_\_\_

Student version

Time: 20 minutes. Total: 100 points + 5 possible bonus points.

Write the requested answers in the boxes. Explain Q1(c), Q2(d), and each bug in Q3; for all other parts, the requested pair, numbers, vectors, or short labels suffice. Each required part of Q1 and Q2(a)–(c) is worth 10 points; Q2(d) and Q3 are worth 20 points each. The WordPiece part of Q1(a) is optional: leaving it blank does not reduce your score; earned bonus points are added beyond 100. The 20-minute limit includes bonus work. No calculator is needed. Sequence positions are numbered from 0;  $\log$  denotes the natural logarithm. Unless stated otherwise, dropout is disabled.

### Q1. Tokenization and Neural Language Models

30 points

- (a) At one tokenizer-training merge step, consider only the two candidate adjacent token pairs below. Counts are measured before this merge:

| Pair   | Pair frequency | First token frequency | Second token frequency |
|--------|----------------|-----------------------|------------------------|
| (x, y) | 12             | 40                    | 40                     |
| (z, w) | 8              | 8                     | 8                      |

**Required BPE (10 points).** Which of the two pairs does frequency-based BPE merge next? Write the ordered pair of tokens.

Your answer:

**Optional WordPiece (5 bonus points; extra reading).** Using the same table, apply the pair-scoring rule from the extra reading and select the higher-scoring candidate. Write the ordered pair of tokens.

Your answer:

- (b) The input embedding table and output layer use separate parameters. The table is used only for lookup. One backward pass looks up IDs  $[2, 5, 2]$ ; the loss gradients with respect to the three returned vectors are  $[2, 1]$ ,  $[0, 3]$ , and  $[-1, 4]$ , in that order. These are the only uses of the table, and there is no weight decay or special padding behavior. Write the two-component data-loss gradient for table row 2.

Your answer:

- (c) In Handout 10's Alice/Bob example, two prefixes differ in an earlier name but have the same final four tokens, the key belonged to. A deterministic fixed-window LM sees only those four tokens. With the same model parameters used for both prefixes, can adding hidden layers alone make their next-token distributions differ? Answer yes/no, explain why in one sentence, and name one input change that would expose the distinction.

Your answer:

## Q2. Attention, Masking, and Cost

50 points

- (a) For one attention head at query position 1, scores for keys 0, 1, 2 are  $[\log 3, 0, \log 6]$ , already divided by  $\sqrt{d_k}$ . The corresponding value vectors are  $[4, 0]$ ,  $[0, 4]$ , and  $[8, 8]$ ; there is no padding. Before softmax, mask keys at positions later than the query; earlier positions and the query's own position are permitted. Write (i) the three attention weights in key order 0, 1, 2, including any zero weight, and (ii) their weighted sum of values, before output projection or residual addition.

Your answer:

- (b) An encoder-decoder has source length  $S = 5$ , target length  $U = 3$ , batch size  $B = 2$ , and  $H = 4$  heads. There is no padding. For standard decoder cross-attention, write four answers: the sequence of representations supplying Q; the sequence supplying K/V; the numerical weight tensor shape with axes ordered as batch, head, query position, key position; and the softmax axis, identified by its role and length.

Your answer:

- (c) In a dense full-sequence forward pass, replace length  $T$  by  $2T$ , holding batch size, model width, head count, FFN width, and dtype fixed. Using leading arithmetic/storage terms, write the multiplicative growth factors for (i) Q/K/V projections, (ii) both  $QK^\top$  and weighted-value products, and (iii) storage of one explicitly materialized attention-weight tensor. Give the three factors in order; do not estimate wall-clock runtime.

Your answer:

- (d) **20 points.** Consider Handout 11's proposed replacement of softmax attention by hard selection of the single highest-scoring key. Hold values fixed and make a small query/key change that leaves the unique winning key unchanged. Consider only gradients through this attention operation. Does the small query/key change alter the output? What does this imply about the gradient through the selection operation to Q/K? Separately, can the selected value receive a nonzero gradient from a loss depending on the output? Use ordinary derivatives, and explain your answers in one or two sentences.

Your answer:

### Q3. Debug a Decoder Model

20 points

The snippet below is intended to train a decoder model to predict the next input token at each position. Each attention query may use its own position and earlier positions, but may not use later positions. There are **exactly three independent bugs** in the attention computation or loss calculation.

**For each bug, give its line number, a corrected expression or line, and a one-sentence reason.** Give one bug per answer box; any order is acceptable. Analyze each bug assuming the other two have been corrected. Do not count a consequence of a bug, such as an invalid numerical output, as a separate bug. You do not need to rewrite the full program.

Use these assumptions:

- `math`, `torch`, and `torch.nn.functional` as `F` are imported. All tensors and model parameters are on the same device and have compatible dtypes.
- $B$  is batch size,  $H$  is head count,  $T \geq 3$  is sequence length, and  $d$  is head width. These and `vocab_size` are positive integers. `token_ids` has shape  $[B, T]$ , dtype `torch.long`, and IDs in  $[0, \text{vocab\_size} - 1]$ . There is no padding or dropout.
- $q$ ,  $k$ , and  $v$  each have shape  $[B, H, T, d]$ . They are correctly computed from each position's token and position embeddings separately; no earlier operation has mixed positions. In `scores`, the final two axes are query position, then key position.
- `lm_head` correctly maps the last feature axis of width  $Hd$  to `vocab_size` raw logits. The intended loss averages next-token cross-entropy over all  $B(T - 1)$  input positions with a following token in the same sequence. All code outside this snippet is correct.

```
1 scores = q @ k.transpose(-2, -1) / math.sqrt(d)
2 allowed = torch.ones(T, T, device=q.device).tril().bool()
3 scores = scores.masked_fill(allowed, float("-inf"))
4 weights = scores.softmax(dim=-2)
5 context = weights @ v
6 context = context.transpose(1, 2).reshape(B, T, H * d)
7 logits = lm_head(context)
8 loss = F.cross_entropy(
9     logits[:, :-1].reshape(-1, vocab_size),
10    token_ids[:, :-1].reshape(-1),
11 )
```

**Bug 1: line number, correction, and reason**

Your answer:

**Bug 2: line number, correction, and reason**

Your answer:

**Bug 3: line number, correction, and reason**

Your answer: