

CS 489: Introduction to Natural Language Processing
Midterm Practice Exam B

Solutions

Instructor: Freda Shi

Total: 100 points

Name: _____

Student ID: _____

1. Instructions

- **Coverage:** Lectures 1–9 (tokenization, embeddings, supervised learning, statistical LMs, neural networks, the Transformer, neural LMs, pretraining objectives).
- **Materials:** Closed book, closed notes; no electronic devices other than non-programmable calculators. Calculators are allowed but not necessary: all numerical constants you may need are given on the **reference sheet** (last page), which also provides the formulas needed for the calculations.
- **Conventions:** “log” means the natural logarithm (nats) unless a question explicitly says $\log_2$ (bits).
- **Part I** (single choice, 5 questions $\times$ 4 pts): exactly one option is correct. No justification required.
- **Part II** (true or false, 8 statements $\times$ 3 pts, 24 pts): mark **T** or **F** next to each statement in Questions 6 and 7. Each correctly classified statement earns 3 pts.
- **Part III** (short answer, 4 questions, 56 pts): show your work. Correct reasoning with an arithmetic slip earns most of the credit. Concise answers are preferred; no answer needs more than a few sentences plus the required math.

2. Part I: Single Choice (20 pts)

Each question is worth 4 points. Exactly one option is correct.

2.1 Question 1

Immediately before the first merge in BPE training, the most frequent adjacent pairs have these counts (all unlisted pairs have lower counts):

Pair	Count
(t, h)	12
(h, e)	9
(e, r)	9
(a, n)	7

A proposed first merge is (h, e). Which statement is correct?

- (A) The proposal follows BPE's rule, since merges may happen in any order.
- (B) The proposal violates BPE's greedy rule: (t, h) must be merged first because it has the highest count.
- (C) The proposal violates BPE's rule because BPE merges the *rarest* pairs first.
- (D) Nothing can be concluded without seeing the vocabulary.

Solution: B. BPE greedily merges the most frequent adjacent pair, so (t, h) with count 12 must precede (h, e) with count 9.

2.2 Question 2

A character-level language model over a 50-symbol alphabet is claimed to have lower test cross-entropy than a model that assigns probability $1/50$ to each symbol at every position. Its reported test cross-entropy is 6.0 bits per character, with exactly one scored token per character. Which conclusion follows?

- (A) The report and claim are inconsistent: uniform prediction gives $\log_2 50 \approx 5.64$ bits, so 6.0 bits is *worse* than uniform.
- (B) The claim is plausible, because cross-entropy measured in bits can never exceed 1.
- (C) The claim is plausible, because uniform guessing yields a loss of 0.
- (D) Nothing can be concluded, because perplexity must equal the vocabulary size.

Solution: A. Uniform prediction has cross-entropy $-\log_2(1/50) = \log_2 50 \approx 5.64$ bits per character, so the reported 6.0 bits is worse, not better.

2.3 Question 3

A sentence S has 10 scored tokens and joint log-likelihood $\log P(S) = -27.7$ (natural log) under a language model. The model's perplexity on this sentence is closest to:

- (A) 16
- (B) $e^{27.7} \approx 10^{12}$
- (C) 2.77
- (D) $10^{-27.7}$

Solution: A. The mean negative log-likelihood is $27.7/10 = 2.77$ nats, giving $\text{PPL} = e^{2.77} \approx 16$.

2.4 Question 4

A decoder-only language model receives $w_1, \dots, w_N$ during teacher-forced training, and position t predicts w_{t+1} . Why is a causal attention mask needed?

- (A) Without it, a position $t < N$ could access its target w_{t+1} among the input tokens, so training loss would not reliably measure next-token prediction from the prefix.
- (B) Without it, the gradients must vanish within a few hundred steps.
- (C) Without it, the model must ignore positional encodings.
- (D) Without it, sequences of different lengths cannot be batched together.

Solution: A. Without a causal mask, position $t < N$ can access its target w_{t+1} in the input, allowing target copying instead of prediction from the prefix.

2.5 Question 5

LSTMs can mitigate the vanishing-gradient problem of plain RNNs primarily because the cell state is updated:

- (A) with a forget gate that controls the norm of gradients along the direct cell-state path, helping mitigate exploding and vanishing gradients.
- (B) by a ReLU activation, which has constant gradient 1.
- (C) in a feed-forward manner, since LSTMs remove recurrence entirely.
- (D) with the forget gate fixed at zero, so the previous cell state is discarded at every time step.

Solution: A. Along the direct cell-state path, the forget gate scales the propagated gradient elementwise. Since forget-gate values lie between 0 and 1, this scaling cannot increase the gradient's Euclidean norm; values close to 1 reduce attenuation. This norm control can help stabilize gradient flow, but does not guarantee that the full LSTM gradient never explodes or vanishes. Setting the forget gate to zero instead discards the previous cell state's direct contribution and blocks that gradient path.

3. Part II: True or False (24 pts)

Each question has four statements. Mark **T** or **F** next to *each* statement (3 pts per correctly classified statement).

3.1 Question 6 (12 pts)

Consider word-embedding methods (PMI matrices, Word2Vec).

- (a) $\text{PMI}(w, c)$ is large for word–context pairs that co-occur far more often than their marginal frequencies predict.
- (b) Multiplying a nonzero word vector by a positive scalar changes its cosine similarity with another fixed nonzero word vector.
- (c) Negative sampling replaces the full-vocabulary softmax objective with binary classification between observed (word, context) pairs and sampled “noise” pairs.
- (d) In the CBOW architecture, the center word is predicted from the sum (or average) of the surrounding context embeddings.

Solution:

- (a) **T.** $\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$ is large when the joint probability greatly exceeds the product of the marginals.
- (b) **F.** For $a > 0$, the scale factor cancels: $\cos(a\mathbf{u}, \mathbf{v}) = \frac{a\mathbf{u}^\top \mathbf{v}}{a\|\mathbf{u}\|\|\mathbf{v}\|} = \cos(\mathbf{u}, \mathbf{v})$.
- (c) **T.** Negative sampling trains a binary classifier to distinguish observed pairs from sampled noise pairs rather than computing a full-vocabulary softmax objective.
- (d) **T.** CBOW predicts the center word from a sum or average of its surrounding context embeddings.

3.2 Question 7 (12 pts)

Consider BERT-style masked language modeling and autoregressive pretraining of decoder-only Transformers.

- (a) BERT can use both left and right context in the corrupted input to predict a selected token.
- (b) BERT’s masked-language-model loss includes every input position, including positions not selected for prediction.
- (c) A position selected for BERT’s masked-language-model objective still contributes to the loss if its input token is left unchanged.
- (d) Even with teacher forcing, a decoder-only Transformer must compute next-token predictions one position at a time rather than in parallel.

Solution:

- (a) **T.** BERT’s bidirectional attention can use both left and right context in the corrupted input.
- (b) **F.** The masked-language-model loss includes only positions selected for prediction.
- (c) **T.** Selection for prediction, not whether the input token is replaced, determines inclusion in the loss.
- (d) **F.** With teacher forcing, input tokens at all positions are known, so next-token predictions can be computed in parallel while a causal mask blocks access to future tokens.

4. Part III: Short Answer (56 pts)

4.1 Question 8 (11 pts)

For this toy unigram-LM tokenization problem, only the following four pieces may be used to segment `unlockable`. Their learned probabilities are:

$$p(\text{un}) = 0.2, \quad p(\text{lock}) = 0.1, \quad p(\text{unlock}) = 0.015, \quad p(\text{able}) = 0.1.$$

The remaining probability mass belongs to vocabulary pieces not used in this toy problem.

- (a) (4 pts) Enumerate all valid segmentations of `unlockable` using only the four pieces above, and compute the unigram-model score $\prod_i p(\text{piece}_i)$ of each segmentation.

Solution: The only valid segmentations and their scores are

$$\begin{aligned} \text{un} \mid \text{lock} \mid \text{able} &: 0.2 \times 0.1 \times 0.1 = 0.002, \\ \text{unlock} \mid \text{able} &: 0.015 \times 0.1 = 0.0015. \end{aligned}$$

- (b) (2 pts) Which segmentation does deterministic highest-score encoding output, and why?

Solution: It outputs `un | lock | able`, since $0.002 > 0.0015$ and deterministic encoding maximizes the product of piece probabilities.

- (c) (2 pts) Suppose a BPE tokenizer has already applied earlier learned merges to obtain `un | lock | able`. Its next applicable rule is $(\text{un}, \text{lock}) \rightarrow \text{unlock}$, and no subsequent rule applies to the resulting sequence. What final segmentation does it produce?

Solution: Applying the specified merge gives `unlock | able`, which is final because no subsequent rule applies.

- (d) (3 pts) Explain why these tokenizers can disagree, referring to BPE's greedy training criterion and the unigram tokenizer's likelihood criterion. Does either criterion guarantee that every token boundary is a morpheme boundary?

Solution: BPE greedily selects the most frequent adjacent pair during training and encodes using the learned merge order, whereas the unigram tokenizer learns piece probabilities using a likelihood criterion and encodes by maximizing their product over segmentations. These different criteria can prefer different segmentations, and neither explicitly enforces or guarantees morpheme boundaries.

4.2 Question 9 (14 pts)

Consider cross-entropy and next-token prediction in the following independent settings.

- (a) (5 pts) For a two-token vocabulary $\{a, b\}$ and a fixed context h , suppose each next token has true probability $1/2$. What model probabilities minimize the *expected* cross-entropy for this context, and what is its minimum value in nats? Explain why this expected loss cannot be zero.

Solution: By $H(P, Q) \geq H(P)$ with equality at $Q = P$, the minimizing probabilities are $q(a | h) = q(b | h) = 1/2$, giving

$$\mathcal{L}_{\min} = -\frac{1}{2} \ln \frac{1}{2} - \frac{1}{2} \ln \frac{1}{2} = \ln 2 \approx 0.693 \text{ nats.}$$

Zero expected loss would require probability 1 for both possible outcomes in the same context, which violates normalization.

- (b) (6 pts) A decoder-only Transformer is trained with teacher forcing on the token sequence (w_1, w_2, w_3, w_4) , using inputs (w_1, w_2, w_3) . What is the target list for next-token prediction? Explain how the causal attention mask restricts attention and why it is needed for next-token prediction.

Solution: The target list is (w_2, w_3, w_4) : position t uses the prefix through w_t to predict w_{t+1} . The causal mask permits position t to attend only to input positions $j \leq t$ and blocks $j > t$. This prevents future-token leakage: without the mask, positions 1 and 2 could access their targets w_2 and w_3 in the input instead of predicting them from the available prefix.

- (c) (3 pts) A separate model assigns probability $1/V$ to every vocabulary token at every position, where $V > 1$. Compute its average cross-entropy in nats and its perplexity on any nonempty test sequence. Do these predictions effectively use the context?

Solution: For any nonempty sequence of N scored tokens,

$$\mathcal{L} = -\frac{1}{N} \sum_{t=1}^N \ln(1/V) = \ln V, \quad \text{PPL} = e^{\ln V} = V.$$

These predictions do not effectively use context: the same uniform distribution is output for every prefix.

4.3 Question 10 (13 pts)

A character-level bigram language model has vocabulary $V = \{a, b\}$ and the following training counts $C(x, y)$:

Previous character	Next: a	Next: b
a	1	1
b	0	2

For evaluation, the initial character a is given as context. Score only the two-character continuation ba. No start or end tokens are used. Briefly justify your answer.

- (a) (4 pts) Under unsmoothed maximum-likelihood estimation, compute $P(b | a)$ and $P(a | b)$. State the perplexity of the two scored characters and explain your answer.

Solution: The unsmoothed probabilities are

$$P(b | a) = \frac{1}{1+1} = \frac{1}{2}, \quad P(a | b) = \frac{0}{0+2} = 0.$$

The continuation has probability $(1/2) \cdot 0 = 0$, so its average negative log-probability and perplexity are both $+\infty$.

- (b) (6 pts) Now apply add-one smoothing:

$$P_{+1}(y | x) = \frac{C(x, y) + 1}{\sum_{z \in V} C(x, z) + |V|}.$$

Compute the two conditional probabilities again, then compute the average cross-entropy in nats for the two scored characters.

Solution: With $|V| = 2$, the smoothed probabilities are

$$P_{+1}(b | a) = \frac{1+1}{2+2} = \frac{1}{2}, \quad P_{+1}(a | b) = \frac{0+1}{2+2} = \frac{1}{4}.$$

Therefore,

$$\mathcal{L}_{+1} = -\frac{1}{2} \left(\ln \frac{1}{2} + \ln \frac{1}{4} \right) = \frac{3}{2} \ln 2 \text{ nats}.$$

- (c) (3 pts) Compare $P(b | b)$ before and after smoothing. Briefly explain the trade-off illustrated by this change.

Solution:

$$P(b | b) = \frac{2}{0+2} = 1, \quad P_{+1}(b | b) = \frac{2+1}{2+2} = \frac{3}{4}.$$

Smoothing gives the unseen $b \rightarrow a$ transition positive probability by reducing the probability of the observed $b \rightarrow b$ transition. This avoids zero probabilities at the cost of assigning less probability to that observed transition.

4.4 Question 11 (18 pts)

A Transformer uses the position-wise feed-forward network $\text{FFN}(\mathbf{x}) = W_2 \text{GELU}(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$. Consider the following separate cases.

- (a) (6 pts) Replace GELU by the identity and consider a stack containing *only these FFNs*, with compatible dimensions and no attention, normalization, or other operations. Rewrite one FFN as an affine map $\mathbf{x} \mapsto W' \mathbf{x} + \mathbf{b}'$, giving W' and $\mathbf{b}'$ explicitly. Show that two such FFNs compose to a single affine map, and state what type of function a stack of any depth computes.

Solution: Replacing GELU by the identity gives

$$\text{FFN}(\mathbf{x}) = W_2(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 = \underbrace{W_2 W_1}_{W'} \mathbf{x} + \underbrace{(W_2 \mathbf{b}_1 + \mathbf{b}_2)}_{\mathbf{b}'}.$$

For two such affine maps $F_1(\mathbf{x}) = A_1 \mathbf{x} + \mathbf{c}_1$ and $F_2(\mathbf{x}) = A_2 \mathbf{x} + \mathbf{c}_2$,

$$F_2(F_1(\mathbf{x})) = A_2(A_1 \mathbf{x} + \mathbf{c}_1) + \mathbf{c}_2 = (A_2 A_1) \mathbf{x} + (A_2 \mathbf{c}_1 + \mathbf{c}_2).$$

Repeated composition therefore leaves any FFN-only stack affine, of the form $\mathbf{x} \mapsto A \mathbf{x} + \mathbf{c}$ regardless of depth.

- (b) (6 pts) Consider a Transformer *encoder* with full bidirectional attention, no attention mask, and no positional information anywhere. It uses the same parameters at every position, and dropout is disabled. Treat each word as one token and compare “dog bites man” with “man bites dog”. Explain how their output token representations are related and why mean pooling gives the same sentence representation.

Solution: Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ be the input embeddings of *dog*, *bites*, and *man*, respectively. With no positional information, “man bites dog” has inputs $(\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1)$, the same vectors in a different order. Stack the input vectors as rows and let P swap positions 1 and 3:

$$X = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad X' = PX = \begin{bmatrix} \mathbf{x}_3^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_1^\top \end{bmatrix}.$$

For one attention head, project each input to a query, key, and value:

$$\mathbf{q}_i = W_Q \mathbf{x}_i, \quad \mathbf{k}_i = W_K \mathbf{x}_i, \quad \mathbf{v}_i = W_V \mathbf{x}_i, \quad i = 1, 2, 3.$$

Stacking these vectors as rows gives $Q = XW_Q^\top$, $K = XW_K^\top$, and $V = XW_V^\top$; shared projection biases are omitted for clarity and preserve the same permutation relation. Scaled dot-product attention, with key dimension d_k and row-wise softmax, is

$$Z = \text{Attention}(Q, K, V) = \underbrace{\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)}_A V.$$

For the reordered inputs, $Q' = PQ$, $K' = PK$, and $V' = PV$, so

$$\frac{Q'K'^\top}{\sqrt{d_k}} = \frac{(PQ)(PK)^\top}{\sqrt{d_k}} = P \left(\frac{QK^\top}{\sqrt{d_k}} \right) P^\top.$$

Permuting the scores within a row only permutes its softmax weights, since the denominator sums the same exponentials; permuting rows only reorders the queries. Therefore,

$$A' = \text{softmax}\left(P \frac{QK^\top}{\sqrt{d_k}} P^\top\right) = PAP^\top, \quad Z' = A'V' = PAP^\top PV = PAV = PZ,$$

where $P^\top P = I$. Thus each head's outputs are reordered in exactly the same way as the inputs. Concatenating heads, the shared output projection, residual connections, per-token layer normalization, and shared position-wise FFNs all preserve this relation through every encoder layer. Hence, if the final outputs for “dog bites man” are $(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$, those for “man bites dog” are $(\mathbf{h}_3, \mathbf{h}_2, \mathbf{h}_1)$. The ordered outputs need not be identical at each position, but mean pooling is invariant to their permutation:

$$\frac{\mathbf{h}_1 + \mathbf{h}_2 + \mathbf{h}_3}{3} = \frac{\mathbf{h}_3 + \mathbf{h}_2 + \mathbf{h}_1}{3}.$$

- (c) (6 pts) Consider a single attention head on a sequence containing just one token $\mathbf{x}$, with linear query, key, and value projections without biases and no normalization. What is the softmax attention weight, and is the attention output linear or nonlinear in $\mathbf{x}$?

Solution: With just one score $s = (W_Q \mathbf{x})^\top (W_K \mathbf{x}) / \sqrt{d_k}$, the softmax weight is $e^s / e^s = 1$, so the head output is simply $W_V \mathbf{x}$. The output is linear in $\mathbf{x}$.

5. Reference Sheet

Formulas (you may use any of these without re-deriving them, unless a question explicitly asks you to derive one):

$$P(w_1, \dots, w_N) = \prod_{i=1}^N p(w_i \mid w_{<i}) \quad (\text{chain rule})$$

$$\mathcal{L} = -\frac{1}{N} \sum_{t=1}^N \log p_\theta(w_t \mid w_{<t}) \quad (\text{cross-entropy loss, nats})$$

$$\text{PPL} = \exp(\mathcal{L}) = 2^{\mathcal{L}_{\text{bits}}} \quad (\text{perplexity})$$

$$H(P, Q) = -\sum_x P(x) \log Q(x) \geq H(P) \quad (\text{equality iff } Q = P)$$

$$H(P) = -\sum_x P(x) \log P(x) \quad (\text{entropy, nats})$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$$\text{Attention}(Q, K, \mathbf{V}) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \mathbf{V}$$

$$\text{FFN}(\mathbf{x}) = W_2 \phi(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (\text{LSTM cell update})$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

Here $\mathcal{L}_{\text{bits}}$ uses $\log_2$ instead of $\log$ in the loss formula, $\odot$ denotes elementwise multiplication, and $\mathbf{V}$ is the matrix of attention values (not vocabulary size). Attention softmax is row-wise; a causal mask sets scores for disallowed positions to $-\infty$ before softmax. Mean pooling is $\mathbf{r} = \frac{1}{N} \sum_{i=1}^N \mathbf{h}_i$.

Numerical constants (approximate; feel free to use them):

$$\ln 2 \approx 0.693, \quad \log_2 50 \approx 5.64, \\ e^{2.77} \approx 16.0.$$

Unigram-LM tokenizer reminder: the score of a segmentation is the product of its piece probabilities (pieces assumed independent); deterministic encoding picks the highest-scoring segmentation.