

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

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

A shared BPE tokenizer with a fixed vocabulary budget is trained on a corpus that is 99% English text and 1% text in a low-resource language L, without rebalancing the data. Assume L shares few character sequences with English. Which consequence is most likely?

- (A) English frequencies dominate merge selection, so L tends to get fewer useful merges and more tokens per word.
- (B) L gets especially short segmentations because BPE prioritizes the least frequent language.
- (C) Both languages necessarily get the same number of tokens per word because they share a tokenizer.
- (D) BPE cannot be trained unless each language contributes the same amount of text.

Solution: A: English dominates the pair counts, so a fixed vocabulary budget tends to leave L with fewer useful merges and more tokens per word.

2.2 Question 2

On a fixed evaluation set, a language model's mean token log-probability is -0.2 nats/token. Which pair gives its average cross-entropy loss and perplexity?

- (A) Loss = -0.2 nats/token; perplexity = $e^{-0.2}$.
- (B) Loss = 0.2 nats/token; perplexity = $e^{0.2}$.
- (C) Loss = 0.2 nats/token; perplexity = $e^{-0.2}$.
- (D) Loss = $e^{0.2}$ nats/token; perplexity = 0.2 .

Solution: B: Cross-entropy negates the mean log-probability, giving 0.2 nats/token and perplexity $e^{0.2}$.

2.3 Question 3

For a fixed tokenizer and scoring convention, which statement about perplexity and sequence likelihood is **true**?

- (A) Two sequences of different lengths with equal perplexity must have equal joint likelihood.
- (B) Two sequences with equal joint likelihood must have equal perplexity, regardless of length.
- (C) Perplexity is a length-normalized transform of the joint likelihood: $\text{PPL}(S) = P(S)^{-1/N}$ for a sequence with $N > 0$ scored tokens.
- (D) Perplexity is the arithmetic mean of the per-token probabilities.

Solution: C: By the chain rule, $\text{PPL}(S) = \exp(-\ln P(S)/N) = P(S)^{-1/N}$, the reciprocal geometric mean of the gold-token probabilities.

2.4 Question 4

In count-based distributional semantics, pointwise mutual information is preferred over raw co-occurrence counts mainly because raw counts:

- (A) are often dominated by very frequent words (e.g., *the*, *of*), whereas PMI corrects for marginal frequency.
- (B) cannot be stored in a matrix of feasible size.
- (C) are always negative and therefore unusable directly.
- (D) require gradient descent to estimate.

Solution: A: PMI uses $\log \frac{P(w,c)}{P(w)P(c)}$ to discount co-occurrences explained by high marginal frequencies rather than a strong association.

2.5 Question 5

An encoder-decoder Transformer and a decoder-only Transformer are trained to generate a target sequence $y_1, \dots, y_n$ from a source sequence x . The decoder-only model receives x as a prefix. Both use causal attention over target positions and teacher forcing with correct next-token alignment, and loss is computed only on target tokens. Which statement is correct?

- (A) Only the encoder-decoder model can learn a conditional distribution $p(y \mid x)$.
- (B) The encoder-decoder model must predict target tokens independently of one another.
- (C) The decoder-only model must also include source-token prediction losses because source tokens appear in its input.
- (D) Both models can minimize the following loss despite using different architectures:

$$-\sum_{t=1}^n \log p_{\theta}(y_t \mid x, y_{<t}).$$

Solution: D: Both architectures can model the autoregressive conditional distribution $p_{\theta}(y \mid x) = \prod_{t=1}^n p_{\theta}(y_t \mid x, y_{<t})$ and minimize its negative log-likelihood. Source tokens can provide context without contributing prediction losses, and each target prediction can depend on previous target tokens.

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)

Let p and q be true and model distributions over a finite next-token vocabulary, with $q(w) > 0$ for every token.

- (a) At a fixed context, two models with the same unique most-probable token must have equal expected cross-entropy under the same p .
- (b) Adding a common finite constant to all logits leaves the softmax cross-entropy loss unchanged.
- (c) For a single labeled example, the cross-entropy loss depends only on the model's probability for the correct token, not on how the remaining probability is distributed among incorrect tokens.
- (d) Different true distributions can yield equal expected cross-entropies but different KL divergences $D_{\text{KL}}(p \parallel q)$.

Solution: (a) **F**: For $p = (1, 0)$, models $(0.75, 0.25)$ and $(0.9, 0.1)$ both predict the first token but have different expected losses, $-\ln 0.75$ and $-\ln 0.9$.

(b) **T**: A common logit shift multiplies every softmax numerator and its denominator by the same factor, leaving all probabilities and the loss unchanged.

(c) **T**: The loss $-\log q(y)$ depends only on the probability of the correct token y .

(d) **T**: For uniform $q = (1/2, 1/2)$, both $p_1 = (1, 0)$ and $p_2 = (1/2, 1/2)$ give cross-entropy $\ln 2$, but their KL divergences from q are $\ln 2$ and 0, respectively.

3.2 Question 7 (12 pts)

Consider standard BPE and the unigram LM tokenizer (as implemented in SentencePiece).

- (a) BPE builds its vocabulary bottom-up by successively merging pairs; the unigram LM tokenizer prunes an initially large vocabulary top-down.
- (b) Given a fixed vocabulary and fixed unigram token probabilities, Viterbi decoding finds a valid segmentation that maximizes the product of its token probabilities.
- (c) Standard BPE training selects each merge by maximizing corpus likelihood under a unigram probabilistic model, rather than by pair frequency.
- (d) Character-level BPE initialized only with characters observed in training can always encode a new, unseen character without an unknown-token replacement or fallback.

Solution: (a) **T**: BPE adds merged symbols bottom-up, while the unigram LM tokenizer prunes pieces from a large seed vocabulary.

(b) **T**: Viterbi decoding uses dynamic programming to maximize $\prod_i P(x_i)$ over all valid segmentations, equivalently maximizing $\sum_i \log P(x_i)$.

(c) **F**: Standard BPE selects the most frequent adjacent pair, not the merge maximizing unigram likelihood.

(d) **F**: Merging known characters cannot create an unseen character, so an unknown-token replacement or fallback is needed.

4. Part III: Short Answer (56 pts)

4.1 Question 8 (12 pts)

You are training a BPE tokenizer on the following corpus (word: count), starting from individual characters and merging only within words. Weight each pair occurrence by the word's count. **No** end-of-word marker is used in this question. If several pairs tie for the maximum count, merge the one whose *merged symbol* comes first alphabetically.

Word	Count
hug	4
hugging	6
hugged	5
huge	2

- (a) (6 pts) Perform the first **three** merges. For each, state the merged pair, its count, and the resulting representation of hugging.

Solution: Initially (h,u) and (u,g) both have count $4 + 6 + 5 + 2 = 17$; the alphabetical tie-break selects hu.

Merge	Rule	Count	hugging becomes
1	(h,u) $\rightarrow$ hu	17	hu g g i n g
2	(hu,g) $\rightarrow$ hug	17	hug g i n g
3	(hug,g) $\rightarrow$ hugg	11	hugg i n g

The second pair occurs in all 17 word occurrences; the third occurs in hugging and hugged, giving $6 + 5 = 11$.

- (b) (3 pts) Using this corpus, explain in one or two sentences why frequently occurring stems or affixes can become BPE pieces.

Solution: Frequently occurring stems or affixes contribute frequent character sequences across word forms, so repeated pair merges can build reusable pieces. Here the repeated spelling hug becomes a piece after two merges, without BPE needing any knowledge of meaning.

- (c) (3 pts) Treat *huge* as a single morpheme unrelated to *hug*. Give its segmentation after two merges and use it to explain why BPE pieces need not correspond to morphemes.

Solution: After two merges, huge is segmented as hug e. Under the stated assumption, this splits a single morpheme into nonmorphemic pieces: frequent shared spelling does not imply shared morphological structure or meaning.

4.2 Question 9 (16 pts)

Consider character-level language models with vocabulary size $V = 50$. Score only the character positions specified below, with no start or end tokens.

- (a) (4 pts) At every scored position, Model U assigns the same finite logit to all 50 characters. Use softmax to find its output distribution, then compute its average cross-entropy loss in nats and its perplexity. Explain why these scores do not depend on which characters occur in the text.

Solution: If all logits equal a finite constant c , softmax gives

$$q_U(w) = \frac{e^c}{50e^c} = \frac{1}{50}, \quad \mathcal{L}_U = -\ln \frac{1}{50} = \ln 50 \approx 3.9120 \text{ nats/token}, \quad \text{PPL}_U = e^{\mathcal{L}_U} = 50.$$

Every gold character receives probability $1/50$, so every scored position has the same loss regardless of the text.

- (b) (6 pts) On two scored positions, Model R assigns probabilities $1/25$ and $1/100$ to the respective gold characters. At each position, it divides the remaining probability equally among the other 49 characters. Compute its average cross-entropy loss in nats and its perplexity.

Solution:

$$\mathcal{L}_R = -\frac{1}{2} \left(\ln \frac{1}{25} + \ln \frac{1}{100} \right) = \frac{1}{2} \ln(25 \cdot 100) = \ln 50 \approx 3.9120 \text{ nats/token},$$

$$\text{PPL}_R = e^{\mathcal{L}_R} = 50.$$

- (c) (6 pts) Consider arbitrary discrete data distributions p and model distributions q over these 50 characters. State separately (i) the smallest cross-entropy attainable when both p and q may vary, and (ii) the minimum when p is fixed and only q may vary. Give both answers in nats and bits, and state whether a finite upper bound exists in either setting.

Solution: (i) When both p and q may vary, the minimum is 0 nats or 0 bits, attained by a deterministic data distribution and a matching probability-one prediction. (ii) For fixed p , the minimum is $H_{\text{nats}}(p)$ nats or $H_{\text{bits}}(p) = H_{\text{nats}}(p) / \ln 2$ bits, attained at $q = p$. There is no finite upper bound in either setting or unit. If $p(w) > 0$ and $q(w) \rightarrow 0$, then $-p(w) \log q(w) \rightarrow +\infty$; setting $q(w) = 0$ makes cross-entropy infinite. The uniform baselines $\ln 50$ nats and $\log_2 50$ bits are not upper bounds.

4.3 Question 10 (10 pts)

- (a) (4 pts) Let $\text{BPT}(S) = -\frac{1}{N} \sum_{i=1}^N \log_2 p(w_i | w_{<i})$ be the average bits-per-token of a sequence with $N > 0$ scored tokens. Starting from the chain rule in log base 2, derive

$$\text{PPL}(S) = 2^{\text{BPT}(S)},$$

and state the general identity relating perplexity to the average loss $\bar{H}$ measured with logarithms of base $b > 1$.

Solution: Taking base-2 logarithms of the chain rule gives

$$\log_2 P(S) = \log_2 \prod_{i=1}^N p(w_i | w_{<i}) = \sum_{i=1}^N \log_2 p(w_i | w_{<i}) = -N \text{BPT}(S).$$

Therefore, by change of base,

$$\text{PPL}(S) = \exp\left(-\frac{\ln P(S)}{N}\right) = \exp((\ln 2)\text{BPT}(S)) = 2^{\text{BPT}(S)}.$$

In general, $\bar{H} = -\frac{1}{N} \sum_i \log_b p(w_i | w_{<i})$ implies $\text{PPL} = \exp((\ln b)\bar{H}) = b^{\bar{H}}$ for $b > 1$.

- (b) (6 pts) On the same held-out tokens with the same tokenizer and scoring convention, Model A has BPT = 2.0 bits/token and Model B has BPT = 2.5 bits/token. Compute each model's perplexity to two decimal places and the difference in their average per-token log-likelihoods in nats (A minus B). Explain why a fixed reduction in bits/token gives a fixed multiplicative improvement in perplexity.

Solution:

$$\text{PPL}_A = 2^{2.0} = 4.00, \quad \text{PPL}_B = 2^{2.5} \approx 5.66.$$

The average per-token log-likelihoods are $-2 \ln 2$ and $-2.5 \ln 2$ nats, so their difference (A minus B) is

$$(-2 \ln 2) - (-2.5 \ln 2) = 0.5 \ln 2 \approx 0.3466 \text{ nats/token}.$$

Reducing loss by Δ bits/token divides perplexity by 2^Δ , independently of the baseline; here $\text{PPL}_B/\text{PPL}_A = 2^{0.5} \approx 1.4142$.

4.4 Question 11 (18 pts)

Consider the following questions about Transformers.

- (a) (6 pts) A Transformer **encoder** uses full bidirectional self-attention and mean pooling for classification, but has no positional encodings or other position-dependent features. Treat each word as a single token with a fixed embedding. All per-token operations share parameters across positions; there is no padding or dropout. Can it produce different pooled representations for dog bites man and man bites dog? Explain what happens to the token representations and their mean, and why input-dependent softmax attention does or does not change the conclusion. State one architectural change that would allow it to use word order.

Solution: No, the encoder is permutation equivariant: swapping dog and man permutes the output token representations in the same way, leaving their mean unchanged. Input-dependent softmax preserves this symmetry because permuting the inputs permutes both query rows and key columns, so the attention weights and weighted outputs are permuted consistently. Shared position-wise FFNs, layer normalization, and residual connections also preserve equivariance. Adding position-specific embeddings would allow the encoder to use word order, although the learned parameters must still exploit them.

- (b) (6 pts) A position-wise FFN is $f(\mathbf{x}) = W_2 g(W_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$. If g is replaced by the identity function, derive the equivalent single affine map, including its bias. Does this replacement guarantee that the *whole Transformer encoder* is affine? Explain using one remaining operation in the encoder.

Solution: With identity g , the FFN becomes

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

This makes the FFN affine, but does not guarantee that the whole encoder is affine. For example, self-attention weights $\text{softmax}(QK^\top / \sqrt{d_k})$ generally depend nonlinearly on the input through the input-dependent queries and keys.

- (c) (6 pts) A decoder-only language model is trained with input tokens [BOS, cats, sleep] and corresponding next-token targets [cats, sleep, EOS]. Positions are numbered 1 through 3. The causal mask below permits attention only to the current and earlier positions, with 0 allowing attention and $-\infty$ blocking it.

$$M = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}.$$

If the mask is removed, which input positions directly reveal the gold targets for predictions at positions 1 and 2? Explain why this leaks information that should be unavailable for next-token prediction.

Solution: Without the mask, input position 2 (cats) reveals the target for prediction at position 1, and input position 3 (sleep) reveals the target for prediction at position 2. This exposes future gold tokens rather than requiring prediction from the available prefix.

5. Reference Sheet

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

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

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

$$\text{PPL} = \exp(\mathcal{L}) \quad (\text{perplexity})$$

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

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

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

$$\mathbf{r} = \frac{1}{n} \sum_{i=1}^n \mathbf{h}_i \quad (\text{mean pooling})$$

$$H(p) = -\sum_w p(w) \log p(w)$$

$$H(p, q) = -\sum_w p(w) \log q(w) = H(p) + D_{\text{KL}}(p \parallel q)$$

$$D_{\text{KL}}(p \parallel q) = \sum_w p(w) \log \frac{p(w)}{q(w)} \geq 0$$

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

$$\log_b x = \frac{\ln x}{\ln b}, \quad \log_b(xy) = \log_b x + \log_b y$$

Attention uses row-wise softmax; V_{val} denotes the matrix of value vectors, not vocabulary size. For a causal mask, set disallowed scores to $-\infty$ before softmax ($e^{-\infty} = 0$). Terms with $p(w) = 0$ contribute zero to entropy, cross-entropy, and KL divergence.

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

$$\ln 2 \approx 0.6931, \quad \ln 50 \approx 3.9120, \quad \log_2 50 \approx 5.6439, \quad 2^{2.5} \approx 5.6569.$$

$$e^0 = 1, \quad e^{\ln 2} = 2, \quad \sqrt{2} \approx 1.4142.$$

BPE reminder: training repeatedly merges the most frequent adjacent pair of symbols; encoding applies the learned merge rules *in training order* to new text.