

CS 5090 · Midterm 1

Software Engineering with Generative AI · Michigan Technological University · Friday, October 9, 2026 · 50 minutes

Name: _____ MTU username: _____

- Closed book, closed notes. No phones, laptops, or AI tools.
- No calculator needed. The formulas and values you need are in the box below: you are tested on using them, not on memorizing them.
- Multiple choice and true/false: circle **one** answer. Other questions: answer in the space provided; show your work for partial credit.
- Short answers are graded on the idea, not the length.
- The Bonus questions are the hardest on the exam. Do them last.

Formula sheet

- **Softmax:** $\text{softmax}(z)_i = e^{z_i} / \sum_j e^{z_j}$ (outputs are positive and sum to 1)
- **Attention:** $\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k}) V$. The "scaled scores" are the entries $z_i = q \cdot k_i / \sqrt{d_k}$ that go into the softmax; a causal mask removes the scores of future positions before the softmax
- **Cross-entropy loss** for one token: $L = -\ln p(\text{correct token})$, where $p(\text{correct token})$ is the probability the model's softmax gives to the token that actually comes next
- **Top-p sampling:** sample only from the smallest set of most likely tokens whose total probability is $\geq p$
- **Chinchilla rule of thumb:** training tokens $D \approx 20 \times \text{parameters } N$
- **pass@k:** generate k samples per problem; a problem counts as solved if at least one of the k passes the unit tests; $\text{pass}@k = (\text{problems solved}) / (\text{total problems})$
- **Values:** $e^0 = 1 \cdot e^1 \approx 2.7 \cdot e^2 \approx 7.4 \cdot \ln 2 \approx 0.69 \cdot \ln(a^b) = b \ln a \cdot 16,384 = 2^{14}$, so $\ln 16,384 = 14 \times \ln 2 \approx 9.7$

Part A	Part B	Part C	Part D	Bonus	Total
/ 30	/ 10	/ 30	/ 30	/ 10	/ 100

Part A · Multiple choice

15 questions × 2 points = 30

A1. In a decoder-only LLM, the linear layer after the last Transformer block maps each position's vector to a vector whose size is:

- (a) d_{model} (b) the number of heads (c) the vocabulary size $|V|$ (d) the context length

A2. Why do code LLMs use subword tokenization (such as BPE)?

- (a) Subword sequences are always shorter than word sequences.
(b) A fixed-size vocabulary can still represent identifiers never seen in training, such as `parse_user_config_v2`.
(c) It removes the need for an embedding layer.
(d) It lets the model see individual characters.

A3. Under a causal mask, in a 5-token sequence the token at position 3 can attend to:

- (a) all 5 positions (b) positions 1, 2, 3 (c) positions 1, 2 only (d) positions 3, 4, 5

A4. Which statement about multi-head attention is TRUE?

- (a) All heads share one set of Q/K/V projections, and W_O averages them.
(b) Each head has its own Q/K/V projections, and W_O learns to combine the heads' outputs.
(c) Each head processes a different subset of the tokens.
(d) Adding heads increases the context length.

A5. During generation, the KV cache stores:

- (a) previous answers, to reuse for other users
(b) the keys and values of tokens already processed, so the prefix is not recomputed
(c) the gradients from the last training step
(d) the vocabulary

A6. Next-token probabilities: return 0.45, if 0.25, for 0.15, x 0.10, pass 0.05. With **top-p sampling**, $p = 0.8$, which tokens can be sampled?

- (a) {return} (b) {return, if} (c) {return, if, for} (d) all five

A7. Beam search with beam width 3:

- (a) samples randomly among the 3 most likely tokens
- (b) keeps the 3 highest-scoring partial sequences at each step instead of only the single best token
- (c) generates 3 answers and returns the majority
- (d) forces the output to follow a grammar

A8. By the Chinchilla rule of thumb, a 50M-parameter model should be trained on about:

- (a) 50M tokens
- (b) 250M tokens
- (c) 1B tokens
- (d) 50B tokens

A9. In LoRA fine-tuning:

- (a) all weights are updated with a smaller learning rate
- (b) the pretrained weights are frozen and only small low-rank matrices are trained
- (c) only the output layer is trained
- (d) the model is retrained from scratch

A10. Compared with pretraining, instruction tuning changes:

- (a) the loss function, but not the data
- (b) the data (instruction–response pairs), but the loss is still next-token prediction
- (c) the architecture of the model
- (d) the tokenizer

A11. In InstructGPT (RLHF), the reward model outputs:

- (a) a probability distribution over the vocabulary
- (b) a single scalar score for a response
- (c) a corrected version of the response
- (d) the ideal response written by a human

A12. What does DPO do?

- (a) trains a reward model, then runs PPO against it
- (b) optimizes the preference data directly, without a reward model or PPO
- (c) pretrains on more data
- (d) samples more answers at inference time

A13. A main reason DeepSeek-R1-Zero used GRPO instead of PPO:

- (a) GRPO eliminates the critic model, reducing the cost of RL training
- (b) GRPO needs human preference rankings
- (c) GRPO does not need any reward
- (d) GRPO trains the tokenizer

A14. Which of these is **NOT** test-time scaling?

- (a) letting the model think longer, e.g. appending "Wait" (budget forcing)
- (b) sampling several solutions and checking them
- (c) searching over partial solutions
- (d) training with RLVR

A15. In HumanEval, a generated solution is counted as correct when:

- (a) it looks similar to the reference solution
- (b) it passes the problem's hidden unit tests
- (c) a human grader likes it
- (d) it compiles

Part B · True or false

10 statements × 1 point = 10

B1. A language model assigns probabilities to sequences of tokens.	T F
B2. The Transformer removed recurrence, so a whole sequence can be processed in parallel during training.	T F
B3. In masked (causal) self-attention, a token can attend to tokens that come after it.	T F
B4. Pretraining requires human-labeled data.	T F
B5. Guided (constrained) decoding can force the output to follow a grammar, such as valid JSON.	T F
B6. LoRA updates all the weights of the base model.	T F
B7. In RLHF, PPO uses the reward model's score rather than the human preference pairs directly.	T F
B8. PPO tries to increase reward while limiting how much the policy changes at each step.	T F
B9. Fluent reasoning is always correct reasoning.	T F
B10. For the same samples, pass@5 is always at least as high as pass@1.	T F

Part C · Work it out

3 questions × 10 points = 30

C1. Causal attention by hand

10 pts

A sequence has 4 tokens t1, t2, t3, t4. For the **query at t3**, the scaled scores $z_i = q_3 \cdot k_i / \sqrt{d_k}$ against the keys of t1, t2, t3, t4 are

[0, 2, 0, 1]. These are the numbers that go into the softmax.

(a) (2 pts) Which score does the causal mask remove, and why?

(b) (6 pts) Compute the attention weights of t3 over the positions it can see (softmax over the remaining scores). To keep the arithmetic simple, use $e^0 = 1$ and $e^2 \approx 7$. Give each weight as a fraction and as a decimal (two places); show your work.

(c) (2 pts) Which token does t3 attend to most, and what do the weights add up to?

C2. Budgeting a small code model

10 pts

You train a code model like your Project 1 model: **40M parameters**, a vocabulary of **16,384** tokens, freshly initialized weights.

(a) (3 pts) Using the Chinchilla rule of thumb, how many training tokens should you budget?

(b) (4 pts) The training loss is the cross-entropy $L = -\ln p(\text{correct token})$. Before any training, the model's softmax is close to uniform over the 16,384 tokens. What is $p(\text{correct token})$ approximately, and therefore what **first logged training loss** do you predict? (Use the values on page 1.)

(c) (3 pts) Your team only has **100M tokens** of code. By the same rule, is 40M parameters too big or too small for this data?

C3. pass@k and verifiable rewards

10 pts

A code model generates 4 samples for each of 5 problems (✓ = passes the unit tests, ✗ = fails). Reminder: **pass@k** counts a problem as solved if at least one of its first k samples passes, and reports the fraction of problems solved.

Problem	sample 1	sample 2	sample 3	sample 4
P1	✓	✓	✗	✓
P2	✗	✗	✗	✗
P3	✗	✗	✓	✗
P4	✓	✗	✗	✗
P5	✗	✓	✓	✗

(a) (3 pts) pass@1, counting only sample 1: _____ (b) (3 pts) pass@4: _____

(c) (2 pts) Why does the difference between (a) and (b) count as test-time (inference-time) scaling?

(d) (2 pts) These samples are used for RLVR training with reward 1 if the tests pass, else 0. Write the rewards of P3's four samples: [__ , __ , __ , __]

Part D • Short answers

3 questions × 10 points = 30

D1. Post-training methods

10 pts

For each method, write the letter of its **training signal** from the box, and briefly describe its **training data**. Each letter is used once.

(P) next-token loss on demonstrations · (Q) score from a reward model trained on human rankings · (R) direct loss that prefers the chosen response over the rejected one · (S) 1 if the answer is verified correct (e.g. tests pass), else 0

Method	Signal	Training data
SFT / instruction tuning		
RLHF with PPO		
DPO		
RLVR with GRPO		

D2. Fine-tuning a model to ask (ClarifyCoder)

10 pts

Code LLMs rarely ask a clarifying question when a coding problem is ambiguous.

(a) (4 pts) Why doesn't more pretraining on GitHub code fix this?

(b) (6 pts) How did ClarifyCoder build its training data, and how did it train the model?

D3. Layer normalization

10 pts

What does layer normalization do in a Transformer block, and why do language models use layer norm rather than batch norm?

Bonus

1 question · 10 points · hardest on the exam

Bonus. Why the Transformer replaced the RNN

10 pts

(a) (3 pts) In an RNN, how many sequential steps are needed before the representation of the **last** token in an n-token sequence can depend on the **first** token? In one Transformer self-attention layer, how many?

(b) (4 pts) Give two reasons a Transformer trains much faster than an RNN on a GPU.

(c) (3 pts) Name one thing the RNN gets for free that the Transformer had to add back, and say how it was added.

End of exam. Check that your name is on the first page.