Homework 4: Reasoning, Vision-Language Models, and Agents — Solutions

CS 388: Natural Language Processing • Fall 2026

---

1. Question 1: RLVR and GRPO (35 points)

Students may find the two mistakes in either order; grade (a) with (b) and (c) with (d) as pairs. Each "effect" answer should assume the other mistake has been fixed.

(a) **(5 pts)** First mistake.

Solution. Line 12. The verifier checks whether the answer string appears *anywhere* in the completion, instead of checking the completion's final answer. The fix is to use `extract_answer`:

```
rewards = [float(extract_answer(c) == answer) for c in completions]
```

Grading: 3 pts for identifying that line 12 rewards the answer appearing anywhere rather than checking the final answer; 2 pts for the fix.

(b) **(7 pts)** Effect of the first mistake.

Solution. The reward no longer measures what we want (a correct final answer), so the policy learns to exploit it, which is reward hacking against a buggy verifier. A completion gets reward 1 whenever the string `"6"` appears anywhere in it, for example:

- **Hedging / listing many candidates.** *"The answer could be 4, 5, 6, or 8 . . . Final answer: 5."* This is rewarded even though the final answer is wrong. Since mentioning more numbers makes a hit more likely, the model learns to enumerate candidates and to produce longer, number-heavy reasoning.
- **Correct intermediate steps, wrong final answer.** If the right number shows up anywhere in the reasoning, the completion is rewarded, so the model is never pushed to get the final step right.
- **Accidental substring matches.** `"6"` is a substring of `"16"`, `"60"`, `"2026"`, etc., so wrong answers that contain the digit are rewarded.

Training reward would go up while accuracy on a correct evaluation (which checks the final answer) stays flat or goes down.

Grading: 3 pts for saying that the reward no longer measures final-answer correctness, so the model is rewarded for the wrong thing (reward hacking). 4 pts for a concrete description of what the model learns (listing many candidates, long number-heavy outputs, right intermediate steps but wrong final answers, substring matches); one well-explained behavior is full credit.

(c) **(5 pts)** Second mistake.

Solution. Line 20. The loss uses the log-probability of the *reference answer* given the question, not of the *sampled completion*. GRPO increases or decreases the probability of the policy's own samples according to their advantage. The fix:

```
logp = policy.log_prob(c, given=question)
```

Grading: 3 pts for identifying that line 20 scores `answer` instead of the completion `c`; 2 pts for the fix.

(d) **(7 pts)** Effect of the second mistake.

Solution. For a given question, `logp` is the same number $\ell = \log \pi_\theta(\text{answer} \mid \text{question})$ for all G completions, because it does not depend on `c`. The group-normalized advantages always sum to zero:

$$\sum_{i=1}^{G} \frac{r_i - \bar{r}}{s + \epsilon} \;=\; \frac{1}{s + \epsilon}\Big(\sum_i r_i - G\bar{r}\Big) \;=\; 0.$$

So the loss for each question is

$$-\sum_i A_i\,\ell \;=\; -\ell \sum_i A_i \;=\; 0,$$

and its gradient is exactly 0. The model never learns anything, no matter what the rewards are.

Even if the advantages did not sum to zero, this would at best be supervised training on the bare reference answer (`"6"`) with no reasoning, which is not what RLVR is for: the point is to reinforce the model's own reasoning traces that led to correct answers.

Grading: 5 pts for showing that the loss for each question is 0 (the log-prob is the same for every completion and the advantages sum to 0), so there is no learning. 3 pts for "the model does not learn from its own samples / trains on the answer only" without the zero-sum argument. Add 2 pts for explaining why training on the answer alone (no reasoning) is not what we want, or for connecting it to the on-policy nature of GRPO.

(e) **(6 pts)** All wrong / all right.

Solution. If all 8 completions are wrong, every reward is 0, so $\bar{r} = 0$ and $s = 0$, and every advantage is $(0-0)/(0+\epsilon) = 0$. The question contributes nothing to the gradient. The same holds if all 8 are correct: every reward is 1, $\bar{r} = 1$, $s = 0$, and every advantage is 0.

GRPO only learns from questions where the group has a *mix* of correct and incorrect completions, since the signal comes from comparing completions within a group.

Grading: 2 pts for each case (advantages are 0, so no gradient); 2 pts for the general point that only mixed groups give signal.

(f) **(5 pts)** Starting from 0% accuracy.

Solution. By (e), every group is all wrong, so every advantage is 0 and the gradient is always 0: training makes no progress at all. RLVR can only sharpen behavior the model already produces at least occasionally. Possible fixes (any one is enough):

- **Curriculum**: start with easier questions the model sometimes solves, and add harder ones as it improves;
- **SFT warm start**: first fine-tune on worked solutions (or distill from a stronger model) so the model gets some questions right;

- **Larger groups or higher temperature**, so a rare correct completion has a chance to appear;
- **Denser reward**, e.g. partial credit for the right format or correct intermediate steps (carefully, given part (b)).

Grading: 3 pts for "no learning, because every advantage is 0"; 2 pts for a reasonable fix.

2. Question 2: Vision-Language Models (30 points)

(a) **(12 pts)** Designing a VLM application.

Solution. Answers will vary; below is one good answer.

(i) **Input / output.** Input: the resident's photo, plus a prompt such as *"Does this photo show a pothole? If so, rate its severity as minor, moderate, or severe, and describe it in one sentence for a repair crew, including where it is on the road."* Output: a structured format the city's system can parse, e.g. JSON `{"is_pothole": true, "severity": "severe", "description": "..."}`. Optionally include metadata (GPS location, time) in the prompt.

(ii) **Zero-shot vs. fine-tuning.** Zero-shot is a reasonable starting point for (1) and (3), since general VLMs recognize potholes and write good descriptions. But "severity" is a city-specific convention, so I would fine-tune (or at least give few-shot examples) on a few thousand past 311 photos labeled by city inspectors with *is_pothole*, severity, and a crew description, including negatives (cracks, puddles, manhole covers, irrelevant photos).

(iii) **Evaluation.** Hold out a labeled test set from different neighborhoods and seasons than the training data. Measure precision and recall for pothole detection, accuracy and a confusion matrix for severity (especially severe predicted as minor, which is the costly error), and have crews or inspectors rate a sample of descriptions. Then run in shadow mode next to the current human triage before letting it act on its own.

(iv) **Failure mode.** For example: severity is hard to judge from a single photo with no sense of scale, so a small pothole in a close-up looks severe. Mitigations: ask users to include a reference object or take a second photo; output a confidence and send uncertain or severe cases to a human. Other good answers: puddles or shadows mistaken for potholes (add hard negatives to the training data), hallucinated details in the description (constrain the description to what is visible, spot-check), night or low-quality photos (ask for a retake), and adversarial or irrelevant submissions.

Grading: 3 pts per item. Full credit requires being specific to this task. A generic answer ("fine-tune it on data and evaluate accuracy") gets at most 1 pt per item.

(b) **(18 pts)** Reviewing Engineer B's architecture.

Solution. **(i)** At 448×448 with 64×64 patches there are $448/64 = 7$ patches per side, so the LLM receives $7 \times 7 = \mathbf{49}$ visual tokens per photo.

The photo is downscaled by $2048/448 \approx 4.57$, so one 64-pixel patch covers $64 \times 4.57 \approx \mathbf{293}$ pixels of the original photo along each side. A pill is 40 pixels across in the original (about $40/4.57 \approx 8.8$ pixels after resizing), so about $293/40 \approx 7$ pills fit along each side of a patch, i.e. up to roughly $7^2 \approx 50$ pills in a single patch when they are packed tightly. With 200 pills spread over 49 tokens, each token has to summarize several pills on average.

(ii) No. Counting small, touching objects requires fine-grained spatial detail: the model has to tell individual pills apart. In this design, each pill is only about 9 pixels across after resizing, and many pills are squeezed into each patch embedding, so the information needed to separate and count them is largely lost before the LLM ever sees the image. No amount of LLM capacity or fine-tuning data can recover it. The other hyperparameters (a 7B LLM,

a 24-layer encoder, AdamW at 2×10^{-5}) are reasonable and are distractors; the bottleneck is the image resolution and patch size. (A further practical issue: standard CLIP-pretrained encoders use patches of 14 or 16 pixels, so a 64-pixel-patch encoder would likely have to be pretrained from scratch.)

(iii) Keep more resolution and use smaller patches, so that a pill spans at least about one patch. For example:

- resize to 1024×1024 (a pill is then about 20 pixels across) and use a standard pretrained encoder with 16×16 patches: $(1024/16)^2 = 64^2 = 4096$ patches;
- merge each 2×2 block of neighboring patch embeddings in the projector (as many current VLMs do), giving $4096/4 = \mathbf{1024}$ visual tokens for the LLM.

Cost: the vision encoder now attends over 4096 patches instead of 49; attention cost grows quadratically, roughly $(4096/49)^2 \approx 7000\times$ more for the attention layers (and about $84\times$ more for the per-token layers). The 1024 visual tokens fit in the 4096-token context with plenty of room for the question and answer, but they make each LLM forward pass about $20\times$ longer than with 49 tokens. Without the 2×2 merging, the 4096 visual tokens alone would fill the context. Tiling the full-resolution image into crops ("AnyRes") is another good answer, as long as the token count is worked out. Also acceptable as an addition: training data where the model must point to or enumerate pills, or a detection-based counter that passes its count to the LLM.

Grading: (i) 2 pts each for the 49 tokens, the $\approx$ 293-pixel patch, and the $\approx$ 50 pills per patch (accept $\approx$ 7 per side). (ii) 2 pts for "no"; 4 pts for explaining that the resolution and patch size destroy the fine detail needed to separate small objects, so the problem cannot be fixed downstream. (iii) 3 pts for a specific change (higher resolution and/or smaller patches, with numbers); 3 pts for working out the token count and checking it against the 4096 context (and noting the compute cost). A change that produces more than 4096 visual tokens with no plan to reduce them loses those 3 pts.

3. Question 3: Agents (35 points)

(a) **(10 pts)** The task as a POMDP.

Solution.

Component	Description
States S	Everyone's true availability next week (including events Carol has not entered, and how flexible each person is), the contents of all four calendars, the emails sent and received so far, and whether an invitation has been sent.
Actions A	Read the user's, Alice's, or Bob's calendar for a time range; look up Carol's free/busy status for a time; email an attendee; wait for and read replies; send an invitation for a time; give up.
Observations O	Calendar contents for the user, Alice, and Bob; Carol's free/busy results; the text of replies. The agent never sees anyone's true availability directly.
Transitions $T(s' \mid s, a)$	Reading and lookups do not change the state; sending an email produces a reply later; sending an invitation ends the episode. Calendars also change on their own as other meetings are booked, so transitions are stochastic.
Reward R	$+1$ if the invitation is for a time when all four can attend, -1 if someone cannot, 0 if no invitation is sent; 0 for intermediate steps (or a small cost per email, as in part (c)).

Grading: 2 pts per component, if specific to this task. Look for: true availability in S but not in O; Carol's free/busy status and email replies as observations; calendars changing on their own as a source of stochastic transitions.

(b) **(6 pts)** Hidden state and information-gathering actions.

Solution. Any two of:

- **Carol's true availability.** The free/busy lookup gives *partial* information: it misses events she has not entered. Emailing Carol gives (nearly) *complete* information, at a cost.
- **Whether Alice's or Bob's calendar entries are firm.** A "busy" block might be movable, or a "free" slot might be needed for focus time. Reading their calendars is *partial*; emailing them is more complete.
- **Whether a slot will still be free when the invitation goes out.** Re-reading the calendars just before sending gives *partial* information, since things can change right after.
- **The user's own preferences** (e.g. preferring mornings). Asking the user gives *complete* information.

Grading: 1 pt for each hidden part of the state, 1 pt for each matching action, 1 pt for each correct "complete or partial" judgment.

(c) **(9 pts)** Invite now or email Carol first?

Solution. **(i)** Invite now: $\mathbb{E}[R] = 0.7 \cdot (+1) + 0.3 \cdot (-1) = 0.4$.

Email first: the agent pays $c = 0.1$, and gets $+1$ only if Carol is free *and* the slot survives the wait, which has probability $0.7 \times 0.8 = 0.56$; otherwise it gives up (0).

$$\mathbb{E}[R] = -0.1 + 0.56 \cdot (+1) = 0.46.$$

Since $0.46 > 0.4$, the agent should **email Carol first**.

(ii) Emailing is better when $0.56 - c > 0.4$, i.e. when $c < \mathbf{0.16}$. So the value of Carol's reply is 0.16: it is worth asking as long as asking costs less than that.

Grading: (i) 2 pts for each expected reward and 1 pt for the decision. (ii) 2 pts for setting up the inequality and 2 pts for $c < 0.16$. Carry forward arithmetic mistakes.

(d) **(10 pts)** Updating the belief after a free/busy lookup.

Solution. **(i)** By Bayes' rule, with F = "Carol is actually free" and "free" the lookup result:

$$P(F \mid \text{free}) = \frac{P(\text{free} \mid F)\,P(F)}{P(\text{free} \mid F)\,P(F) + P(\text{free} \mid \neg F)\,P(\neg F)} = \frac{1 \cdot 0.7}{1 \cdot 0.7 + 0.4 \cdot 0.3} = \frac{0.7}{0.82} \approx \mathbf{0.854}.$$

(ii) With $p \approx 0.854$:

$$\text{invite now: } 2p - 1 \approx 0.707, \qquad \text{email first: } 0.8p - 0.1 \approx 0.583.$$

So the agent should now **invite now**. The free lookup was cheap evidence that raised the agent's confidence above the point where paying for an email and risking the slot is worthwhile. (By the same algebra as (c), invite-now wins whenever $2p - 1 > 0.8p - 0.1$, i.e. $p > 0.75$.)

(iii) If the lookup shows *busy*, then $P(\text{busy} \mid F) = 0$, so $P(F \mid \text{busy}) = 0$: Carol is definitely busy, since the lookup only reports busy when there is a real event. The agent should not invite for Tuesday at 2 pm or email about it; it should look for another slot (e.g. check Carol's free/busy for other times that work for the user, Alice, and Bob).

Grading: (i) 2 pts for setting up Bayes' rule and 2 pts for ≈ 0.854. (ii) 1 pt for each expected reward and 1 pt for the decision. (iii) 1 pt for $P = 0$ and 2 pts for "look for another slot". Carry forward errors from (i).