TWO-STAGE PROMPT OPTIMIZATION: FROM REASONING-GUIDED SEARCH TO GRADIENT-GUIDED REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing automatic prompt optimization methods often focus on either broad semantic search or fine-grained local refinement, limiting their ability to benefit from both. We propose a two-stage framework that combines complementary reasoning-based and gradient-based prompt optimization, and introduce GradPO as its second-stage refiner. The first stage can use any general reasoning-based optimizer to make broad prompt improvements based on feedback. In the second stage, GradPO uses target-model gradients to select multiple high-impact spans and combines short, context-aware replacements through beam search, without requiring a separate optimizer model. Experiments on FS-TACRED, FS-FewRel, and MATH-500, spanning relation extraction and mathematical reasoning, show that these two forms of optimization complement each other: local refinement usually further improves prompts found by the first stage. Across the relation extraction tasks, GradPO provides the most consistent improvements, achieving state-of-the-art performance on FS-TACRED with Qwen3-4B while remaining competitive on FS-FewRel. On MATH-500, our GradPO variants improve first-stage prompts more frequently than the alternative refiners, further showing that our approach is effective across both reasoning and non-reasoning tasks.

1 INTRODUCTION

Large language models (LLMs) are commonly guided for downstream tasks through natural-language instructions and demonstrations. However, their performance can be highly sensitive to small changes in prompt wording or formatting (Sclar et al., 2024), while manually finding an effective prompt is time-consuming and does not scale well across tasks or models. Automatic prompt optimization (APO) addresses this problem by using task feedback or evaluation scores to iteratively search for better natural-language prompts while keeping the model parameters fixed (Pryzant et al., 2023; Yang et al., 2024; Wang et al., 2024; Guo et al., 2025). It therefore offers a practical way to improve LLM task performance without task-specific parameter training.

Natural-language prompt optimization has mainly followed two directions: reasoning-based and gradient-based optimization. Reasoning-based methods use feedback to rewrite prompts in natural language (Wang et al., 2024; Yuksekgonul et al., 2024), enabling broad semantic and logical improvements within a few iterations. Gradient-based methods, in contrast, do not directly learn from semantic or logical feedback; instead, they use loss and gradient signals to identify impactful tokens or spans for precise local edits (Wen et al., 2023; Das et al., 2025). Local refinements are important because LLM behavior can be highly sensitive to small token-level choices (Sclar et al., 2024). The two directions are therefore *complementary*: reasoning-based optimization supports global prompt search from feedback, while gradient-based optimization provides fine-grained refinement. However, most prior work studies them in isolation, leaving their combination underexplored.

In this paper, we study automatic prompt optimization. We propose a two-stage framework that combines reasoning-based global search with gradient-guided local refinement. To the best of our knowledge, this is the first work to systematically study this combination for natural-language prompt optimization. In the first stage, a reasoning-based optimizer uses LLM-based feedback to make broad natural-language improvements. In the second stage, we introduce GradPO, which uses

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

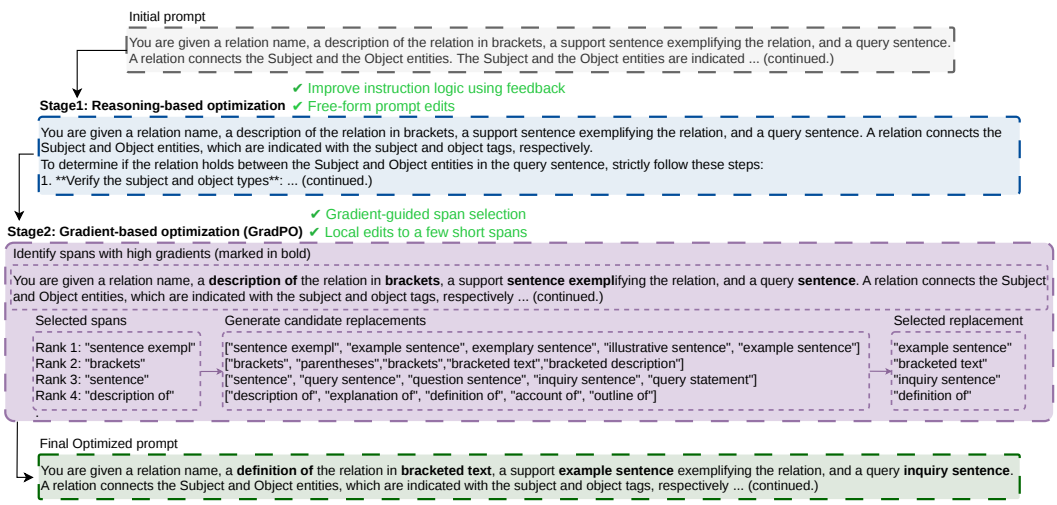


Figure 1: Overview of our two-stage framework with a real Qwen3-4B example on few-shot relation extraction. Stage 1 uses a reasoning-based optimizer to make broad semantic and logical revisions using feedback, producing an intermediate prompt. Stage 2 applies our GradPO to find high-impact local wording choices left unresolved by the coarser first-stage optimizer. Target-model's gradients rank the *selected spans* (shown in bold); GradPO then generates context-aware *candidate replacements* for each span and uses beam search to choose the combination of *selected replacements* with the lowest task and fluency loss. For example, "description of" becomes "definition of," and "brackets" becomes "bracketed text." These targeted edits preserve the prompt's overall structure while further refining an already optimized prompt. Overall, our results show that the reasoning-guided stage improves prompts broadly in a few iterations, while the gradient-guided stage provides targeted local refinement, with GradPO emerging as the most consistent second-stage refiner.

target-model gradients to select multiple high-impact spans and combines short, context-aware replacements through beam search without requiring a separate optimizer model. We consider both LLM-generated replacements and replacements based on the target model's token probabilities. Figure 1 illustrates the full pipeline with a real optimization example. Our experiments show that the two stages are complementary: global search improves prompts quickly, while local refinement makes precise edits that further improve performance (Section 5). Compared with alternative second-stage refiners, GradPO improves first-stage prompts more consistently across both relation extraction and mathematical reasoning.

Our primary contributions are as follows:

- We propose a two-stage framework for natural-language prompt optimization that combines reasoning-based global search with gradient-based local refinement, bringing together two optimization directions that prior work has largely studied separately.
- We introduce GradPO, a lightweight second-stage refiner that bridges the gap between GreaTer's narrow one-token search and LPO's broad rewrites through gradient-selected, context-aware span edits combined by beam search. It requires no separate optimizer model and can refine prompts produced by any reasoning-based optimizer.
- We evaluate the framework on three datasets spanning non-reasoning relation extraction (FS-TACRED and FS-FewRel) and mathematical reasoning (MATH-500). Across both task types, GradPO provides the most consistent second-stage refinement, and our framework achieves state-of-the-art performance on FS-TACRED with Qwen3-4B while remaining competitive on FS-FewRel. On MATH-500, our GradPO variants improve first-stage prompts more frequently than the alternative refiners.

2 Related Work

Reasoning-based prompt optimization. Many natural-language APO methods use LLMs to revise prompts based on task feedback, evaluation scores, or search history. Methods such as APO (Pryzant et al., 2023), PromptAgent (Wang et al., 2024), OPRO (Yang et al., 2024), TextGrad (Yuksekgonul et al., 2024), and EvoPrompt (Guo et al., 2025) explore broad prompt changes through feedback, planning, or search. LPO (Jain & Chowdhary, 2025) instead asks an optimizer LLM to focus its rewrites on selected local spans. These approaches can make semantic and logical improvements within a few iterations, making them suitable for broad first-stage search.

Gradient-based prompt optimization. A separate line of work uses model gradients to identify influential parts of a prompt. Some methods optimize continuous prompt representations or use gradient-aligned soft-prompt updates rather than directly editing natural language (Wang et al., 2025; Zhu et al., 2023), while discrete methods search over natural-language prompt tokens (Wen et al., 2023). Among them, GreaTer (Das et al., 2025) is closest to our work: it uses target-model gradients to select and replace one token at a time using only left context. In contrast, GradPO edits multiple high-impact spans, uses prompting-based replacement generation with both left and right context, and combines candidate edits through beam search, enabling richer local search (Section 3).

Prompting and prompt optimization for relation extraction. Recent work improves relation extraction with LLMs through prompting, reasoning, and demonstration design (Ma et al., 2023; Xu et al., 2023; Wan et al., 2023; Mo et al., 2024; Chakma et al., 2026), but does not optimize the task instruction itself. Most prompt-optimization work for RE instead learns soft prompts for pretrained encoders (Chen et al., 2022; Lv et al., 2022). A recent exception optimizes natural-language prompts for relation triple extraction (Mihindukulasooriya et al., 2025); however, it does not study episodic few-shot relation extraction (FSRE). This setting is particularly challenging because each episode presents a changing set of candidate relations, and the model must predict the query relation from only a few demonstrations. We study natural-language prompt optimization in this episodic setting on FS-TACRED and FS-FewRel in addition to math reasoning.

Prompt optimization for mathematical reasoning. Mathematical reasoning is also an important setting for APO. OPRO (Yang et al., 2024) uses an LLM to optimize instructions from task scores, while LPO (Jain & Chowdhary, 2025) reports gains from localized rewrites on mathematical reasoning benchmarks. These studies show that reasoning prompts benefit from optimization, but do not combine broad reasoning-based search with gradient-guided span refinement. We evaluate this combination on MATH-500 (Hendrycks et al., 2021) to study whether the same two-stage framework is effective for a reasoning-intensive task.

3 Two-stage Prompt Optimization

3.1 Prompt Optimization Task

Prompt optimization improves an LLM's task performance by revising its natural-language prompt while keeping the model parameters fixed. Given training and validation data, it uses training signals to generate candidate prompts and validation performance to select the strongest one. More formally, let f_θ be a fixed target LLM and P_0 an initial natural-language task prompt. For each task example (x, y), we write the model output as $(r, \hat{y}) = f_\theta(P, x)$, where r is an optional reasoning trace and $\hat{y}$ is the final prediction, and compared with the ground truth y. For a non-reasoning task, $r = \emptyset$ and the model produces only $\hat{y}$. The training signals can be observed errors for reasoning-based optimization or loss and gradient information for gradient-based optimization; the final prompt P^* is selected from the candidate prompts $\{P_i\}$ based on the highest validation score.

We apply this formulation to two task types. For episodic few-shot relation extraction (FSRE), x contains few candidate relations and support examples, together with a query sentence and entity pair; the task uses no reasoning trace ($r = \emptyset$), and $\hat{y}$ is one candidate relation or `no_relation`[1].

[1]Here, `no_relation` means that none of the candidate relations in the episode holds between the entity pair, not that the entities have no relation.

For MATH-500, x is a mathematical problem, r is the generated step-by-step reasoning, and $\hat{y}$ is the final answer compared with y. Only the task prompt, optimization data, and evaluation metric change across tasks; the two-stage procedure remains the same. Appendix B.2 provides the detailed formulation of each task.

3.2 Overview of Our Algorithm

Our algorithm has two stages, illustrated through a concrete end-to-end prompt transformation in Figure 1. Starting from P_0, the first stage uses a reasoning-based optimizer to make broad, free-form revisions and returns the strongest prompt based on validation score, denoted P_R. This stage any reasoning-based optimization strategies. To our knowledge, we are the first to use gradient-based prompt optimization as a second-stage refiner after reasoning-based search. This stage initializes $P^{(0)} = P_R$, performs Q local-refinement iterations, and returns the final prompt $P^* = P^{(Q)}$. Complete prompt examples are provided in Appendix E.

3.3 Local Refinement via Gradient-Based Prompt Optimization (GradPO)

Starting from $P^{(0)} = P_R$, we further refine the first-stage prompt with GradPO, our proposed gradient-guided local prompt optimizer. Unlike the prior one-token editing method GreaTer (Das et al., 2025), GradPO identifies and edits multiple high-impact spans using short replacements informed by both left and right context. Its prompting-based replacements help preserve local grammar, while beam search combines span edits for richer local search without changing the overall prompt structure. Algorithm 1 gives the complete execution, while Figure 1 traces these steps on a real prompt.

At iteration q, GradPO samples a training subset $\mathcal{D}_{\text{grad}}$ and computes gradients for each token in the current prompt $P^{(q-1)}$ from the average loss on $\mathcal{D}_{\text{grad}}$. The task loss and gradients are computed from the final prediction $\hat{y}$, not from the reasoning trace r even if present (e.g. math task). Rather than selecting a single token for editing, it groups nearby high-gradient tokens into editable spans. Starting from the strongest gradient peaks, it expands each span to the left or right as long as neighboring tokens have sufficiently large gradient values under a threshold. The resulting spans are ranked by their maximum gradient magnitude, and the top spans are selected for refinement. In Figure 1, this is the *Selected spans* step: the gradients identify phrases such as "description of" and "brackets" that remain after Stage 1 (the gradient and span-selection steps in Algorithm 1).

After selecting spans, GradPO generates candidate replacements for each span using LLM prompting. Specifically, we mark the selected spans in the prompt and use the same target LLM to propose several natural-language replacements. We further study another variant, GradPO-Prob, where replacements are selected based on the target model's token probabilities. For GradPO-Prob, we use the left context of each span to sample high-probability first-token candidates, and then let the target model continue generation to complete the remaining tokens in the span. This corresponds to *Generate candidate replacements* in Figure 1; for example, "description of" receives candidates including "definition of" and "explanation of" (the candidate-generation step in Algorithm 1).

GradPO uses beam search to combine replacements across multiple spans. Here, the beam is the set of candidate prompts retained after replacing each span, beginning with the current prompt $P^{(q-1)}$. For each selected span, every candidate replacement is applied to every retained prompt. The target LLM performs the replacement while keeping the remaining text unchanged, which helps preserve local grammar. The resulting prompts are scored, and only the highest-scoring prompts, up to the fixed beam width, are retained before moving to the next span. Because the original text is also a candidate for each span, the search considers combinations ranging from no span changes to edits of all selected spans, while retaining only the highest-scoring combinations. We separately study the replacement source through LLM-generated candidates in GradPO and probability-based candidates in GradPO-Prob. This is the *Selected replacement* step in Figure 1, where the beam retains edits such as "definition of" and "bracketed text" (the beam-search step in Algorithm 1).

Following GreaTer, we score prompts using the following objective:

$$\mathcal{L}(P, \mathcal{D}_{\text{grad}}) = \mathcal{L}_{\text{task}}(P, \mathcal{D}_{\text{grad}}) + \lambda \, \text{PPL}(P), \tag{1}$$

where $\mathcal{L}_{\text{task}}$ is the task loss on $\mathcal{D}_{\text{grad}}$, PPL is a fluency penalty based on perplexity, and λ controls the penalty strength. After all spans have been processed, the retained prompts are evaluated on $\mathcal{D}_{\text{val}}$,

Table 1: Comparison of second-stage refiners: GreaTer makes narrow gradient-guided edits, while LPO makes broader edits without gradients. GradPO combines gradient guidance with multiple short, context-aware edits and beam search while relying only on the target model.

Feature	GreaTer	LPO	GradPO
Gradient guidance	✓	✗	✓
Multiple spans per iteration	✗	✓	✓
Bidirectional context	✗	✓	✓
Prompting helps local grammar	✗	✓	✓
Beam search over span edits	✗	✗	✓
Short edits on average	✓	✗	✓
Separate optimizer model not required	✓	✗	✓

and the best prompt is selected as the refined prompt. This produces the *Final Optimized prompt* at the bottom of Figure 1 (the validation-selection step in Algorithm 1); Full optimized prompts are provided in Appendix E.

Finally, for both GradPO and GradPO-Prob, we avoid using a separate optimizer model for candidate generation or span replacement, relying only on the target model itself. Moreover, we disable reasoning during these prompting steps to keep the refinement process lightweight.

3.4 Alternative Baselines for Local Refinement

We compare GradPO against two local refinement alternatives, GreaTer and LPO, using the same P_R for each method. None of the two baselines was originally designed specifically as a local refiner. We adapt them as strong baselines in the second step of our two-stage framework.

GreaTer. GreaTer (Das et al., 2025) is a gradient-based prompt optimization method. It was originally designed to iteratively optimize the full prompt by editing one token at a time in the order tokens present in the prompt, rather than to refine selected spans after a global optimizer. For a selected token position, GreaTer proposes candidate replacement tokens appear on language-model probabilities based on the context to the left and scores each resulting prompt using the objective in Equation 1 on a sampled training subset $\mathcal{D}_{\text{grad}}$. It then keeps the top token replacements with the lowest combined loss. Similar to GradPO, GreaTer does not use a separate optimizer model.

We also test a top-gradient variant that recomputes gradients at each step and edits the token with the largest gradient magnitude. We refer to this variant as GreaTer-TG. In both variants, the search space is highly local because each iteration changes only one token. This makes GreaTer a useful baseline for precise but narrow gradient-based refinement.

LPO. We also compare against LPO (Jain & Chowdhary, 2025) as a gradient-free local refinement baseline. At each iteration, LPO collects errors on sampled training examples and asks a separate optimizer model (usually larger than the target model) to mark few short spans for editing. It then asks the optimizer model to generate revised prompts by rewriting the marked spans while keeping the remaining text unchanged. The revised prompts are evaluated on the validation set, and the best-performing prompt among the generated candidates is selected for the next iteration.

Key differences among local refiners. Table 1 highlights several important differences between the second-stage refiners. GreaTer uses gradients and requires no separate optimizer model, but it edits only one token at a time without bidirectional replacement context, which can produce locally inconsistent edits. LPO edits multiple spans with both sides of the context, but it requires a separate optimizer model to reason over errors and generate rewrites; it also lacks gradient guidance and often makes broader changes. GradPO is distinct from both: it uses target-model gradients to select multiple high-impact spans, generates short prompting-based replacements with bidirectional context to help preserve local grammar, and uses beam search to select coherent edit combinations. It therefore provides richer local search than GreaTer and more targeted refinement than LPO, while relying only on the target model and preserving the first-stage prompt structure. We analyze these differences further in Section 6 and provide representative edit examples in Appendix C.

Additional implementation details and hyperparameters are provided in Appendix B.1; the input prompt format is described in Appendix D.

4 EXPERIMENTAL SETUP

LLMs. We use Qwen3-4B and Gemma3-4B as the target models in all main experiments. We focus on 4B target models, following prior prompt-optimization work in which target models are often smaller than the models used to optimize their prompts (Wang et al., 2024; Guo et al., 2025; Jain & Chowdhary, 2025). Our goal is to further improve their already optimized prompts through second-stage refinement. For methods that require a separate optimizer model, including the first-stage optimizers and LPO, we use the corresponding larger models (Qwen3-14B and Gemma3-12B).

First-stage Optimizers. We leverage three diverse reasoning-based optimizers. RPO performs iterative, feedback-driven population search: an optimizer LLM analyzes representative successes and failures and generates a revised prompt (Pryzant et al., 2023; Yang et al., 2024; Wang et al., 2024). EvoPrompt-DE performs non-feedback evolutionary search by combining prompts from a population (Guo et al., 2025); we measure fitness on training data and evaluate the strongest prompt on the validation set at each iteration. ETGPO uses an optimizer LLM to summarize training errors into a taxonomy and revise the prompt in a single step (Singh et al., 2026). Thus, the three optimizers cover iterative feedback search, non-feedback evolutionary search, and structured one-step revision. Further procedural and implementation details are provided in Appendix B.1.

Datasets. We evaluate on FS-TACRED, FS-FewRel, and MATH-500 (Hendrycks et al., 2021). FS-TACRED is the realistic few-shot benchmark introduced by Sabo et al. (2021). FS-FewRel is the episodic benchmark created by Chakma et al. (2026) from the FewRel dataset (Han et al., 2018), for direct comparison with their work. Both datasets are in 5-way 1-shot episodic form with five sets of $10,000$ episodes. The setting is challenging because the model sees only one support example per candidate relation. It is especially hard because most of the queries do not match any candidate relation and must be mapped to `no_relation`[2]. For MATH-500, we use official MATH training split for optimization and use the official 500-problem test set as test data. MATH-500 is also challenging for smaller models because it contains diverse competition-level problems that require multi-step mathematical reasoning. Additional details about the train and validation split construction are provided in Appendix B.3.

Hyperparameter Optimization. We run the first-stage optimizers for 5 or 10 iterations, following standard APO practice, while ETGPO uses a single iteration as originally proposed. Table 7 analyzes first-stage runs up to 20 iterations. For relation extraction, second-stage hyperparameters are tuned on FS-TACRED using RPO prompts and reused on FS-FewRel. For MATH-500, they are tuned using the RPO-5 and RPO-10 prompts and transferred unchanged to EvoPrompt and ETGPO. Additional details and hyperparameters are provided in Appendix B.1.

Metrics and Evaluation. After optimization, the top prompt from each method is selected using validation performance for test-set inference. Following the standard FS-TACRED evaluation, we use its scorer to report micro-averaged precision, recall, and F1 over positive relation labels. Correct `no_relation` predictions are treated as true negatives and do not contribute directly to these metrics, while false positives and false negatives are penalized. We use mean F1 as the primary selection and comparison metric and report it in the main results; full precision, recall, and F1 results averaged over five runs are provided in Appendix (Tables 4 and 5). For MATH-500, we use the PRM800K answer grader (Lightman et al., 2023) to check the extracted final answer $\hat{y}$ against the ground truth and report accuracy averaged over five decoding seeds. All prompt selection and hyperparameter tuning use validation data only. In the final tables, we report the mean, standard deviation, and statistical significance results under each method's evaluation protocol.

[2] 97% and 95% of episodes of FS-TACRED and FS-FewRel have `no_relation` as gold labels.

Table 2: Results with RPO and EvoPrompt-DE as first-stage optimizers across all three datasets. R and G denote reasoning-based and local refinement iterations, respectively. Cells report test F1 for FS-TACRED and FS-FewRel, and test accuracy for MATH-500, as mean ± standard deviation over five runs; bold marks the best task metric within each first-stage block. The highest-scoring second-stage prompt within each first-stage block is significantly better than scores marked with * ($p < 0.05$) or † ($p < 0.01$). Missing second-stage entries indicate no dev-set improvement; in these cases, we retain the stage-1 prompt, so the test score is unchanged from the first stage and omitted for clarity. For Qwen3-4B on FS-FewRel, the 10-iteration EvoPrompt-DE prompt did not improve over the 5-iteration prompt and is therefore omitted, since retaining the 5-iteration prompt would produce identical results. Overall, GradPO is the most consistent refiner: it improves 15 of the 23 first-stage prompts and gives the best result in 8 blocks, the highest counts among compared refiners.

	#itr.	Qwen3-4B			Gemma3-4B		
Method	R+G	FS-TACRED	FS-FewRel	MATH-500	FS-TACRED	FS-FewRel	MATH-500
Initial Prompt		24.0 ± 0.59	35.6 ± 0.66	72.4 ± 0.26	11.3 ± 0.53	25.3 ± 1.27	61.5 ± 0.87
RPO	5+0	26.5† ± 1.19	34.8† ± 0.60	77.7† ± 0.83	13.9† ± 0.94	30.3† ± 1.14	64.9 ± 1.72
+LPO	5+1	24.8† ± 0.98	35.3* ± 0.82	77.2† ± 0.95	**15.0** ± 0.54	**37.5** ± 1.30	-
+GreaTer	5+1	25.9† ± 1.43	34.5† ± 0.99	**79.1** ± 0.87	14.0† ± 1.01	30.2† ± 1.23	**66.1** ± 0.91
+GreaTer-TG	5+1	27.2† ± 1.53	-	-	-	-	65.2 ± 1.77
+GradPO	5+1	**29.0** ± 1.37	**35.8** ± 0.80	77.6† ± 0.41	13.9† ± 1.06	32.7† ± 1.12	65.6 ± 0.95
+GradPO-Prob	5+1	27.4† ± 0.94	-	77.7† ± 0.70	-	-	64.9 ± 1.29
RPO	10+0	28.3† ± 1.13	36.3† ± 0.74	79.4† ± 0.55	11.7† ± 0.45	32.2† ± 1.30	64.9† ± 1.01
+LPO	10+1	**31.9** ± 1.65	-	78.6† ± 0.37	-	-	**67.5** ± 0.86
+GreaTer	10+1	28.8† ± 1.38	-	80.6 ± 1.12	10.9† ± 0.27	-	-
+GreaTer-TG	10+1	26.9† ± 1.21	36.4† ± 0.74	78.8† ± 1.11	**12.0** ± 0.43	32.7* ± 1.25	-
+GradPO	10+1	28.9† ± 1.46	**36.9** ± 0.82	**81.2** ± 1.07	11.9 ± 0.11	**33.3** ± 1.40	67.4 ± 0.84
+GradPO-Prob	10+1	28.4† ± 0.68	36.6* ± 0.76	-	-	31.8† ± 1.28	66.6 ± 1.69
EvoPrompt-DE	5+0	24.6 ± 1.21	35.9† ± 0.59	77.2† ± 1.01	**14.5** ± 0.53	27.9† ± 1.67	66.6† ± 1.37
+LPO	5+1	-	**38.0** ± 0.56	-	13.1† ± 0.27	30.9† ± 1.79	-
+GreaTer	5+1	24.2† ± 1.04	36.2† ± 0.50	-	-	32.8† ± 1.86	-
+GreaTer-TG	5+1	24.1† ± 1.31	35.9† ± 0.85	-	13.9† ± 0.72	28.3† ± 1.62	-
+GradPO	5+1	**24.8** ± 0.99	-	77.4† ± 0.90	-	**33.7** ± 1.71	-
+GradPO-Prob	5+1	**24.8** ± 1.08	36.7† ± 0.79	**77.8*** ± 1.02	**14.5** ± 0.72	32.0† ± 1.61	**68.5** ± 0.76
EvoPrompt-DE	10+0	24.5 ± 0.98	-	77.7† ± 0.54	10.1† ± 0.42	35.2† ± 1.64	65.7† ± 0.46
+LPO	10+1	24.4 ± 1.26	-	-	10.4 ± 0.59	36.8† ± 1.17	-
+GreaTer	10+1	**24.7** ± 0.78	-	-	10.2* ± 0.39	-	**67.7** ± 1.55
+GreaTer-TG	10+1	-	-	**78.0*** ± 0.69	**10.5** ± 0.39	-	67.6 ± 0.38
+GradPO	10+1	23.4† ± 0.74	-	77.7† ± 0.79	10.3 ± 0.50	**37.5** ± 1.33	65.6† ± 1.45
+GradPO-Prob	10+1	-	-	-	9.9† ± 0.46	35.2† ± 1.51	67.2 ± 1.75

5 RESULTS

Tables 2 and 3 report the main results with our two-stage prompt optimization approach on FS-TACRED, FS-FewRel, and MATH-500. Within each first-stage block, all second-stage optimizers start from exactly the same first-stage optimized prompt. When a second-stage refinement does not improve development performance, we retain the corresponding first-stage prompt; test scores are same with the first stage prompt and omitted from for clarity from tables. For relation extraction, Appendix (Tables 4 and 5) report precision and recall for every evaluated prompt.

Local refinement results in further improvement after reasoning-based optimization. Tables 2 and 3 show that a second-stage local refinement is usually helpful after a first-stage reasoning-based optimizer. Across all 29 valid cases[3], at least one second-stage refiner improves mean test per-

[3]Each case is one first-stage optimized prompt that all second-stage refiners attempt to improve. There are 30 possible cases: 2 target models × 3 datasets × 5 first-stage prompts (RPO and EvoPrompt after 5

Table 3: Results with ETGPO as the first-stage optimizer across all three datasets. R and G denote reasoning-based and local refinement iterations, respectively. Cells report test F1 for FS-TACRED and FS-FewRel, and test accuracy for MATH-500, as mean ± standard deviation over five runs; bold marks the best task metric within each dataset/model block. The highest-scoring second-stage prompt within each first-stage block is significantly better than scores marked with * ($p < 0.05$) or † ($p < 0.01$). Missing second-stage entries indicate no dev-set improvement; in these cases, we retain the stage-1 prompt, so the test score is unchanged from the first stage and omitted for clarity. Overall, GradPO improves 3 of the 6 first-stage prompts, more than any other refiner, and gives the best result in 2 blocks, tied with GreaTer.

	#itr.	Qwen3-4B			Gemma3-4B		
Method	R+G	FS-TACRED	FS-FewRel	MATH-500	FS-TACRED	FS-FewRel	MATH-500
Initial Prompt		24.0 ± 0.59	35.6 ± 0.66	72.4 ± 0.26	11.3 ± 0.53	25.3 ± 1.27	61.5 ± 0.87
ETGPO	1+0	28.3 ± 0.75	35.0† ± 1.06	**73.1** ± 1.32	13.1† ± 0.32	**35.2** ± 1.73	67.4* ± 1.17
+LPO	1+1	28.3 ± 0.89	-	-	12.5† ± 0.41	27.8† ± 1.52	-
+GreaTer	1+1	**28.4** ± 0.67	**35.7** ± 1.24	-	-	-	-
+GreaTer-TG	1+1	27.8 ± 0.92	-	73.0 ± 1.01	-	35.0 ± 1.76	67.8 ± 0.74
+GradPO	1+1	27.7† ± 0.66	35.1† ± 1.04	72.6 ± 0.86	**14.1** ± 0.58	-	**68.9** ± 2.07
+GradPO-Prob	1+1	26.3† ± 1.03	35.2* ± 0.93	71.5† ± 0.88	-	-	68.6 ± 1.45

formance in 26 cases, with a statistically significant improvement in 22 cases ($p < 0.05$). GradPO is the most consistent, improving 18 cases, with 11 statistically significant gains. The original GreaTer improves 12 cases, with 5 significant gains, while LPO improves 9 cases, with 8 significant gains. Even modest gains are notable because they come from a single refinement iteration ($Q = 1$) applied to already optimized first-stage prompts. Overall, the second stage further improves first-stage prompts across both reasoning and non-reasoning tasks.

GradPO outperforms alternatives for refinement. In addition to improving the most first-stage prompts (18 of 29 cases), GradPO gives the highest mean result more often than the other refiners. Across the 29 cases, GradPO is best in 10 cases, and 9 of these results are significantly better than the corresponding stage-1 prompt ($p < 0.05$). The original GreaTer is best in 6 cases, with 3 significant results, while LPO is best in 5 cases, all of which are significant. Thus, GradPO improves first-stage prompts most often and gives the best refinement most often, making it the most consistent refiner.

Gradient guidance yields reliable gains, not random improvements. To determine whether the gains arise from meaningful span selection rather than arbitrary local editing, we replace gradient-selected spans with randomly selected spans while keeping candidate generation and beam search unchanged (Table 6). GradPO passes dev selection in 25 of 29 valid cases, compared with only 17 of 29 under random span selection. Among these dev-selected refinements, GradPO improves over the first-stage prompt on test in 18 cases, whereas random selection improves only 11 cases. These results show that local editing alone is not sufficient: randomly chosen spans lead to substantially less consistent improvements, while gradient information provides a reliable signal for identifying spans where refinement is likely to be useful.

One GradPO step complements longer first-stage search. For relation-extraction cases as examples, Table 7 shows that later first-stage iterations yield only modest average gains and remain variable. Starting from the best prompt found in the first 10 iterations, one GradPO step improves 7 of 8 settings and outperforms the best prompt found over all 20 first-stage iterations in 5 of 8 settings. These results support GradPO as a complementary local refiner rather than a replacement for first-stage optimization.

GradPO improves over previous works on FS-TACRED with Qwen3-4B. Table 8 shows that our prompt optimization approach sets a new state of the art on FS-TACRED with Qwen3-4B. The

and 10 iterations, and ETGPO after one iteration). For Qwen3-4B on FS-FewRel, the 10-iteration EvoPrompt prompt did not improve over its 5-iteration prompt, so we retained the same 5-iteration prompt and omitted this redundant case, leaving 29 cases.

best previous result is 27.6 F1, while our best setting in Table 2 is higher with a score of 29.0. On FS-FewRel, our approach remains competitive: with EvoPrompt, it reaches 37.5 F1 (Table 2), compared with 38.4 by Chakma et al. (2026), who retrieve additional examples from a large annotated corpus using semantic rules for in-context learning, whereas we optimize the task prompt.

6 Qualitative Analysis

Why does GradPO provide stronger local refinement? Our results suggest that effective second-stage refinement requires a balance between edit locality and flexibility; Table 1 summarizes the complete design differences among the refiners. GreaTer operates one token at a time, resulting in a narrow search space and occasionally locally inconsistent substitutions (Appendix Table 14). LPO makes much larger edits, averaging 21.19 tokens per span across all three datasets (Appendix Table 11). In contrast, GradPO uses gradients to identify high-impact spans, prompting with bidirectional context to help keep replacements locally grammatical, and beam search to combine multiple edits coherently. It balances the two extremes by editing 2.76 spans on average, with only 3.20 tokens per span. Overall, GradPO combines balanced local edits with advantages such as bidirectional context, prompting-based replacements, and beam search, helping it reach a better refined prompt than others baselines.

GradPO makes effective local edits with limited impact on prompt readability The prompt examples show that GradPO usually keeps the prompt structure intact and only edits a few local spans (Appendix Table 14). In the RPO example shown in Figure 1, the GradPO prompt still reads very similarly to the stage-1 prompt, but the local edits are enough to improve the score. Overall, the refined prompts remain mostly natural because GradPO changes only a small part of the instruction and uses prompting-based replacements while encouraging low-perplexity outputs. However, GradPO can still introduce minor local inconsistencies despite using prompted replacements. For example, one GradPO edit produces the phrase "query inquery sentence," which is less natural than the original wording. Such inconsistencies are typically limited to a small local span, while the overall structure and readability of the prompt are largely preserved.

Gradient-selected spans primarily capture task-level wording. To examine whether span selection is driven by individual examples, we analyze all 125 top-gradient spans from 29 GradPO runs across relation extraction and MATH-500 (Appendix Table 12). General task wording is the largest category (48.8%), covering both input descriptions and reusable task concepts such as "sentence," "name," and "formulas." Only three selections (2.4%) are tied to a specific relation rule or illustrative example: "province," "hier," and "locker." Thus, 97.6% of the selected spans are not tied to an individual relation or example, suggesting that the averaged gradients primarily capture general reusable prompt wordings.

We provide additional analyses showing that most GradPO gains occur in the first few iterations, its multi-span beam search adds limited computational cost, refinements preserve prompt length, and gradient-guided edits transfer weakly across target models (Appendix B.6).

7 Conclusion

This paper proposed a novel two-stage prompt-optimization strategy that combined complementary forms of optimization: reasoning-based optimization made broad prompt improvements, while GradPO refined high-impact spans with local gradient-guided edits. Our analysis showed that GradPO improved over alternative refiners by combining short gradient-selected span edits, context-aware replacements, and beam search. This complementary combination provided consistent second-stage refinement across relation extraction and mathematical reasoning on MATH-500, yielded a new state of the art on FS-TACRED with Qwen3-4B. Overall, our findings showed that combining global reasoning-based search with local gradient-based refinement was effective across both reasoning and non-reasoning tasks, and GradPO emerged as the most consistent second-stage refiner in our evaluations.

AI USE STATEMENT

AI tools were used to proofread and make minor edits to author-written text and to assist with constructing code templates. The research ideas and initial manuscript drafts were produced by the authors. All AI-assisted text and code were reviewed and verified by the authors.

ETHICS STATEMENT

This work uses publicly available open-weight language models and established benchmark datasets. We did not collect any new human-subject or personal data.

REPRODUCIBILITY STATEMENT

The optimization procedure and experimental setup are described in Sections 3 and 4 and Algorithm 1; Appendix B provides the hyperparameters, data splits, inference details, and additional results, while Appendices D and E provide the prompt formats and examples.

REFERENCES

Aunabil Chakma, Mihai Surdeanu, and Eduardo Blanco. Structured semantic information helps retrieve better examples for in-context learning in few-shot relation extraction, 2026. URL https://arxiv.org/abs/2601.20803.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 2778–2788, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511998. URL https://doi.org/10.1145/3485447.3511998.

Sarkar Snigdha Sarathi Das, Ryo Kamoi, Bo Pang, Yusen Zhang, Caiming Xiong, and Rui Zhang. GReater: Gradients over reasoning makes smaller language models strong prompt optimizers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fWRBheSJth.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers, 2025. URL https://arxiv.org/abs/2309.08532.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4803–4809, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1514. URL https://aclanthology.org/D18-1514/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Yash Jain and Vishal Chowdhary. Local prompt optimization. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 75–81, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-190-2. doi: 10.18653/v1/2025.naacl-short.7. URL https://aclanthology.org/2025.naacl-short.7/.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Bo Lv, Li Jin, Yanan Zhang, Hao Wang, Xiaoyu Li, and Zhi Guo. Commonsense knowledge-aware prompt tuning for few-shot nota relation classification. *Applied Sciences*, 12(4), 2022. ISSN 2076-3417. doi: 10.3390/app12042185. URL https://www.mdpi.com/2076-3417/12/4/2185.

Xilai Ma, Jing Li, and Min Zhang. Chain of thought with explicit evidence reasoning for few-shot relation extraction. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2334–2352, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.153. URL https://aclanthology.org/2023.findings-emnlp.153/.

Nandana Mihindukulasooriya, Niharika S. D'Souza, Faisal Chowdhury, and Horst Samulowitz. Automatic prompt optimization for knowledge graph construction: Insights from an empirical study, 2025. URL https://arxiv.org/abs/2506.19773.

Ying Mo, Jiahao Liu, Jian Yang, Qifan Wang, Shun Zhang, Jingang Wang, and Zhoujun Li. C-ICL: Contrastive in-context learning for information extraction. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10099–10114, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.590. URL https://aclanthology.org/2024.findings-emnlp.590/.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=WRYhaSrThy.

Ofer Sabo, Yanai Elazar, Yoav Goldberg, and Ido Dagan. Revisiting few-shot relation classification: Evaluation data and classification schemes. *Transactions of the Association for Computational Linguistics*, 9:691–706, 2021. doi: 10.1162/tacl_a_00392. URL https://aclanthology.org/2021.tacl-1.42/.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

Mayank Singh and Eduardo Blanco. Learning to generate rules for realistic few-shot relation classification: An encoder-decoder approach. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5907–5921, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.341. URL https://aclanthology.org/2024.findings-emnlp.341/.

Mayank Singh, Vikas Yadav, and Eduardo Blanco. Error taxonomy-guided prompt optimization, 2026. URL https://arxiv.org/abs/2602.00997.

Robert Vacareanu, George C.G. Barbosa, Enrique Noriega-Atala, Gus Hahn-Powell, Rebecca Sharp, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. A human-machine interface for few-shot rule synthesis for information extraction. In Hannaneh Hajishirzi, Qiang Ning, and Avi Sil (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pp. 64–70, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-demo.8. URL https://aclanthology.org/2022.naacl-demo.8/.

Robert Vacareanu, Fahmida Alam, Md Asiful Islam, Haris Riaz, and Mihai Surdeanu. Best of both worlds: A pliable and generalizable neuro-symbolic approach for relation classification. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2576–2594, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.165. URL https://aclanthology.org/2024.findings-naacl.165/.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. GPT-RE: In-context learning for relation extraction using large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3534–3547, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.214. URL https://aclanthology.org/2023.emnlp-main.214/.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=22pyNMuIoa.

Yangkun Wang, Zihan Wang, and Jingbo Shang. Direct prompt optimization with continuous representations. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2642–2652, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.133. URL https://aclanthology.org/2025.acl-long.133/.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 51008–51025. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a00548031e4647b13042c97c922fadf1-Paper-Conference.pdf.

Xin Xu, Yuqi Zhu, Xiaohan Wang, and Ningyu Zhang. How to unleash the power of large language models for few-shot relation extraction? In Nafise Sadat Moosavi, Iryna Gurevych, Yufang Hou, Gyuwan Kim, Young Jin Kim, Tal Schuster, and Ameeta Agrawal (eds.), *Proceedings of the Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pp. 190–200, Toronto, Canada (Hybrid), July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.sustainlp-1.13. URL https://aclanthology.org/2023.sustainlp-1.13/.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15659–15669, October 2023.

A Appendix

B Additional details and hyperparameters of the optimization methods

B.1 Additional details on optimizer methods

Stable prompt selection. Because validation scores can fluctuate across folds, we favor prompts with both strong and stable performance using

$$\mathrm{Score}(P) = \mathrm{Metric}_{\mathrm{mean}}(P) - \lambda\, \mathrm{Metric}_{\mathrm{std}}(P), \tag{2}$$

where the metric is F1 for relation extraction and accuracy for MATH-500. We use $\lambda = 2$ for relation extraction and $\lambda = 1$ for MATH-500.

Algorithm 1 GradPO

Require: Stage-1 prompt P_R, train set $\mathcal{D}_{\text{train}}$, dev set $\mathcal{D}_{\text{val}}$, iterations Q, beam width M, candidates per span K, mode $m \in \{\text{GEN}, \text{PROB}\}$
Ensure: Optimized prompt P^*

$P^{(0)} \leftarrow P_R$
for $q = 1$ to Q **do**
  // Select high-impact spans using target-model gradients
  Sample $\mathcal{D}_{\text{grad}} \subset \mathcal{D}_{\text{train}}$
  Compute token gradients on $P^{(q-1)}$ using $\mathcal{D}_{\text{grad}}$
  Select top spans $\mathcal{S}$ based on gradients

  // Generate candidate replacements for each selected span
  for all $s = (i, j) \in \mathcal{S}$ **do**
    if $m = \text{GEN}$ **then**
      $C_s \leftarrow \text{LLM}(\text{Mark}(P^{(q-1)}, s))$
    else if $m = \text{PROB}$ **then**
      $C_s \leftarrow \text{TopK}_{x_i}\left(p(x_i \mid P^{(q-1)}_{<i})\right)$
    end if
  end for
  $\mathcal{B} \leftarrow \{P^{(q-1)}\}$

  // Use beam search to compose replacements across spans
  for all $s \in \mathcal{S}$ **do**
    $\mathcal{B}' \leftarrow \{P_b[s \leftarrow c] \mid P_b \in \mathcal{B},\, c \in C_s\}$
    $\mathcal{B} \leftarrow \text{TopM}_{P \in \mathcal{B}'}\left(-\mathcal{L}(P, \mathcal{D}_{\text{grad}})\right)$
  end for

  // Select the best refined prompt based on validation data
  $P^{(q)} \leftarrow \arg\min_{P \in \mathcal{B}} \mathcal{L}(P, \mathcal{D}_{\text{val}})$
end for
$P^* \leftarrow P^{(Q)}$
return P^*

RPO RPO is our adaptation of the common reasoning-based prompt-optimization loop (Pryzant et al., 2023; Yang et al., 2024; Wang et al., 2024): run trials with the current prompt, collect feedback from representative successes and failures, and use that feedback to revise the prompt. Starting from P_0, it maintains a small prompt population and samples a parent at each iteration based on validation performance. A separate optimizer LLM analyzes the trial outcomes, explains them, and generates the revised prompt. The revision is evaluated on the validation set, added to the population, and used to prune weaker prompts. Because the optimizer can rewrite any part of the prompt, RPO can make broad changes in a few iterations. For Qwen3-4B experiments on relation extraction, we use Qwen3-4B itself as the RPO optimizer because it produced stronger validation-selected prompts than Qwen3-14B. At each iteration, we sample 100 training examples: a relation with one support sentence and a query for relation extraction, or a math problem with the target model's generated solution for MATH-500. We retain three feedback examples using mixed selection, including at least one correct and one incorrect prediction when available; correctness is determined by the predicted relation or final mathematical answer, respectively. Parent prompts are sampled with a softmax distribution over Equation 2 using temperature 1.0, and we limit the population size to 10.

EvoPrompt EvoPrompt is a non-feedback evolutionary optimizer that initializes a population around P_0 and generates new prompts by combining existing prompts. We use its DE variant because it outperformed the GE variant in the original work. Unlike the original setup, which evaluates fitness directly on validation data, we measure fitness on training data to reduce validation overfitting and evaluate only the strongest population member on the validation set at each iteration. We use a population size of five. The initial population contains two human-written prompts and three AI-written prompt variants. At each iteration, we sample $1,000$ training examples and score the

Table 4: Full FS-TACRED results for all three first-stage optimizers and second-stage refiners. P, R, and F1 are reported as mean ± standard deviation over five test splits; R and G denote reasoning-based and local-refinement iterations. Bold marks the highest F1 within each first-stage block. The highest-scoring second-stage prompt within each first-stage block is significantly better than scores marked with * ($p < 0.05$) or † ($p < 0.01$). Missing second-stage entries indicate no development-set improvement, so the retained stage-1 prompt has an unchanged test score and the repeated value is omitted.

	#itr.	Qwen3-4B			Gemma3-4B		
	R+G	P	R	F1	P	R	F1
Initial Prompt		17.6 ± 0.44	38.2 ± 1.38	24.0 ± 0.59	10.4 ± 0.56	12.5 ± 0.65	11.3 ± 0.53
RPO	5+0	30.8 ± 1.12	23.3 ± 1.34	26.5† ± 1.19	15.8 ± 1.12	12.4 ± 0.83	13.9† ± 0.94
+LPO	5+1	30.4 ± 1.41	21.0 ± 0.90	24.8† ± 0.98	15.1 ± 0.63	15.0 ± 0.49	**15.0** ± 0.54
+GreaTer	5+1	32.1 ± 1.44	21.8 ± 1.58	25.9† ± 1.43	14.6 ± 1.11	13.5 ± 0.96	14.0† ± 1.01
+GreaTer-TG	5+1	27.9 ± 1.58	26.5 ± 1.69	27.2† ± 1.53	-	-	-
+GradPO	5+1	29.5 ± 1.25	28.5 ± 1.66	**29.0** ± 1.37	15.9 ± 1.25	12.4 ± 0.95	13.9† ± 1.06
+GradPO-Prob	5+1	29.9 ± 1.04	25.3 ± 1.05	27.4† ± 0.94	-	-	-
RPO	10+0	35.8 ± 1.19	23.4 ± 1.26	28.3† ± 1.13	09.8 ± 0.33	14.4 ± 0.72	11.7† ± 0.45
+LPO	10+1	35.3 ± 1.71	29.1 ± 1.72	**31.9** ± 1.65	-	-	-
+GreaTer	10+1	35.8 ± 1.18	24.1 ± 1.57	28.8† ± 1.38	09.6 ± 0.31	12.5 ± 0.31	10.9† ± 0.27
+GreaTer-TG	10+1	37.7 ± 1.76	21.0 ± 1.15	26.9† ± 1.21	09.8 ± 0.27	15.6 ± 0.79	**12.0** ± 0.43
+GradPO	10+1	35.3 ± 1.36	24.5 ± 1.53	28.9† ± 1.46	10.2 ± 0.20	14.3 ± 0.31	11.9 ± 0.11
+GradPO-Prob	10+1	37.9 ± 1.24	22.7 ± 0.73	28.4† ± 0.68	-	-	-
EvoPrompt-DE	5+0	19.4 ± 0.80	33.7 ± 2.24	24.6 ± 1.21	11.7 ± 0.55	19.3 ± 0.55	**14.5** ± 0.53
+LPO	5+1	-	-	-	13.7 ± 0.70	12.7 ± 0.20	13.1† ± 0.27
+GreaTer	5+1	20.0 ± 0.72	30.5 ± 1.81	24.2† ± 1.04	-	-	-
+GreaTer-TG	5+1	20.3 ± 1.06	29.6 ± 1.90	24.1† ± 1.31	11.4 ± 0.71	17.9 ± 0.81	13.9† ± 0.72
+GradPO	5+1	20.7 ± 0.83	31.1 ± 1.49	**24.8** ± 0.99	-	-	-
+GradPO-Prob	5+1	19.9 ± 0.79	33.0 ± 1.86	**24.8** ± 1.08	11.7 ± 0.68	19.1 ± 0.80	**14.5** ± 0.72
EvoPrompt-DE	10+0	21.6 ± 0.53	28.4 ± 1.77	24.5 ± 0.98	12.5 ± 0.54	08.5 ± 0.48	10.1† ± 0.42
+LPO	10+1	21.0 ± 0.95	29.2 ± 1.93	24.4 ± 1.26	11.4 ± 0.74	09.6 ± 0.57	10.4 ± 0.59
+GreaTer	10+1	22.1 ± 0.38	28.2 ± 1.44	**24.7** ± 0.78	12.3 ± 0.70	08.7 ± 0.39	10.2* ± 0.39
+GreaTer-TG	10+1	-	-	-	12.6 ± 0.59	09.0 ± 0.49	**10.5** ± 0.39
+GradPO	10+1	18.5 ± 0.54	32.1 ± 1.56	23.4† ± 0.74	12.2 ± 0.77	08.9 ± 0.44	10.3 ± 0.50
+GradPO-Prob	10+1	-	-	-	12.4 ± 0.58	08.3 ± 0.51	09.9† ± 0.46
ETGPO	1+0	21.5 ± 0.54	41.4 ± 1.80	28.3 ± 0.75	12.1 ± 0.43	14.4 ± 0.36	13.1† ± 0.32
+LPO	1+1	21.3 ± 0.73	42.3 ± 1.56	28.3 ± 0.89	11.0 ± 0.40	14.4 ± 0.62	12.5† ± 0.41
+GreaTer	1+1	21.2 ± 0.56	43.2 ± 1.32	**28.4** ± 0.67	-	-	-
+GreaTer-TG	1+1	27.5 ± 0.87	28.1 ± 1.27	27.8 ± 0.92	-	-	-
+GradPO	1+1	22.0 ± 0.49	37.3 ± 1.55	27.7† ± 0.66	11.5 ± 0.51	18.2 ± 0.77	**14.1** ± 0.58
+GradPO-Prob	1+1	24.1 ± 1.17	29.0 ± 0.91	26.3† ± 1.03	-	-	-

prompts on this sample using Equation 2. Only the best prompt from the population is evaluated on the validation set at the end of each iteration.

ETGPO ETGPO is feedback-driven but revises the prompt without an iterative search loop. It evaluates P_0 on training examples, collects the errors, and asks an optimizer LLM to organize them into a taxonomy of common problems. The optimizer then uses this structured error summary to generate a revised prompt in one step. The original method generates a single prompt from the error taxonomy. Instead, we generate five prompts from the same error taxonomy and evaluate them on the validation set to select the best one. The error taxonomy is constructed from errors observed on $1,000$ sampled training examples: incorrect relation predictions for relation extraction or incorrect final answers for MATH-500. We use the default hyperparameters: taxonomy batch size 6, coverage threshold 0.7, and minimum category size 2. We tune the maximum number of error categories between 5 and 7, and select 5 based on validation performance.

Table 5: Full FS-FewRel results for all three first-stage optimizers and second-stage refiners. P, R, and F1 are reported as mean ± standard deviation over five test splits; R and G denote reasoning-based and local-refinement iterations. Bold marks the highest F1 within each first-stage block. The highest-scoring second-stage prompt within each first-stage block is significantly better than scores marked with * ($p < 0.05$) or † ($p < 0.01$). Missing second-stage entries indicate no development-set improvement, so the retained stage-1 prompt has an unchanged test score and the repeated value is omitted. For Qwen3-4B on FS-FewRel, the 10-iteration EvoPrompt-DE prompt did not improve over the 5-iteration prompt and is therefore omitted, since retaining the 5-iteration prompt would produce identical results.

	#itr.	Qwen3-4B			Gemma3-4B		
	R+G	P	R	F1	P	R	F1
Initial Prompt		29.9 ± 0.72	44.1 ± 0.76	35.6 ± 0.66	39.9 ± 2.13	18.6 ± 1.12	25.3 ± 1.27
RPO	5+0	42.4 ± 0.97	29.5 ± 0.55	34.8† ± 0.60	35.1 ± 1.06	26.7 ± 1.27	30.3† ± 1.14
+LPO	5+1	43.2 ± 0.91	29.9 ± 0.79	35.3* ± 0.82	32.8 ± 1.11	43.8 ± 2.03	**37.5** ± 1.30
+GreaTer	5+1	42.2 ± 1.32	29.1 ± 0.88	34.5† ± 0.99	35.0 ± 1.13	26.5 ± 1.35	30.2† ± 1.23
+GreaTer-TG	5+1	-	-	-	-	-	-
+GradPO	5+1	42.9 ± 1.27	30.7 ± 0.64	**35.8** ± 0.80	30.2 ± 0.85	35.7 ± 1.70	32.7† ± 1.12
+GradPO-Prob	5+1	-	-	-	-	-	-
RPO	10+0	33.5 ± 0.98	39.6 ± 0.66	36.3† ± 0.74	34.8 ± 1.39	29.9 ± 1.42	32.2† ± 1.30
+LPO	10+1	-	-	-	-	-	-
+GreaTer	10+1	-	-	-	-	-	-
+GreaTer-TG	10+1	32.8 ± 0.93	40.9 ± 0.70	36.4† ± 0.74	34.6 ± 1.20	31.1 ± 1.47	32.7* ± 1.25
+GradPO	10+1	34.2 ± 0.94	40.2 ± 0.94	**36.9** ± 0.82	32.5 ± 1.28	34.1 ± 1.77	**33.3** ± 1.40
+GradPO-Prob	10+1	33.1 ± 1.03	40.8 ± 0.48	36.6* ± 0.76	35.2 ± 1.19	29.0 ± 1.54	31.8† ± 1.28
EvoPrompt-DE	5+0	35.5 ± 0.71	36.3 ± 0.67	35.9† ± 0.59	42.8 ± 1.54	20.7 ± 1.57	27.9† ± 1.67
+LPO	5+1	35.8 ± 0.68	40.6 ± 0.48	**38.0** ± 0.56	38.8 ± 1.37	25.6 ± 1.89	30.9† ± 1.79
+GreaTer	5+1	35.5 ± 0.55	37.0 ± 0.80	36.2† ± 0.50	38.7 ± 1.47	28.4 ± 2.05	32.8† ± 1.86
+GreaTer-TG	5+1	35.7 ± 1.11	36.2 ± 0.78	35.9† ± 0.85	42.4 ± 1.23	21.3 ± 1.58	28.3† ± 1.62
+GradPO	5+1	-	-	-	37.3 ± 1.43	30.7 ± 1.91	**33.7** ± 1.71
+GradPO-Prob	5+1	33.5 ± 1.05	40.6 ± 0.51	36.7† ± 0.79	40.2 ± 1.29	26.5 ± 1.69	32.0† ± 1.61
EvoPrompt-DE	10+0	-	-	-	37.2 ± 1.33	33.3 ± 1.89	35.2† ± 1.64
+LPO	10+1	-	-	-	35.7 ± 0.85	37.9 ± 1.54	36.8† ± 1.17
+GreaTer	10+1	-	-	-	-	-	-
+GreaTer-TG	10+1	-	-	-	-	-	-
+GradPO	10+1	-	-	-	35.7 ± 1.09	39.5 ± 1.73	**37.5** ± 1.33
+GradPO-Prob	10+1	-	-	-	37.4 ± 1.28	33.2 ± 1.70	35.2† ± 1.51
ETGPO	1+0	35.6 ± 1.35	34.5 ± 0.93	35.0† ± 1.06	33.8 ± 1.50	36.6 ± 2.14	**35.2** ± 1.73
+LPO	1+1	-	-	-	35.9 ± 1.96	22.7 ± 1.30	27.8† ± 1.52
+GreaTer	1+1	36.2 ± 1.44	35.1 ± 1.21	**35.7** ± 1.24	-	-	-
+GreaTer-TG	1+1	-	-	-	35.8 ± 1.51	34.2 ± 2.08	35.0 ± 1.76
+GradPO	1+1	34.3 ± 1.22	36.0 ± 0.95	35.1† ± 1.04	-	-	-
+GradPO-Prob	1+1	33.8 ± 1.14	36.8 ± 0.81	35.2* ± 0.93	-	-	-

GradPO *Common settings.* Across both tasks and target models, we tune the maximum number of spans $S \in \{3, 5, 7\}$, maximum tokens per span $T \in \{3, 5\}$, candidates per span $C \in \{5, 7, 10\}$, span-expansion threshold $H \in \{0.30, 0.45, 0.60\}$, and maximum balanced gradient-set size $G \in \{100, 200, 300, 500\}$. For both tasks, we balance the gradient set using equal numbers of correct and incorrect predictions so that neither outcome dominates the gradient signal. For relation extraction, we further balance TP, TN, FP, and FN instances equally; for MATH-500, we use equal numbers of correct and incorrect final answers. GradPO uses one refinement iteration and beam width five; we tune the fluency weight between 0 and 1 and select 0.5. We use log perplexity rather than exponential perplexity, following GreaTer, and include the original span as a candidate so beam search can leave it unchanged. The retained beam prompts are selected using Equation 2.

Table 6: Random-span control comparing stage-1 prompts with GradPO and random-span refinement. Values are test F1 over five sets of 10,000 episodes for FS-TACRED and FS-FewRel, and test accuracy over five decoding seeds for MATH-500, reported as mean ± standard deviation. Candidate generation and beam search are unchanged; only span selection differs. Bold indicates best improvement over the corresponding stage-1 prompt. Missing second-stage entries indicate no dev-set improvement; in these cases, we retain the stage-1 prompt, so the test score is unchanged from the first stage and omitted for clarity. The Qwen3-4B FS-FewRel EvoPrompt-DE prompt at 10 iterations was not a valid first-stage prompt.

		Qwen3-4B			Gemma3-4B		
First-stage	#itr.	Stage 1	GradPO	Random	Stage 1	GradPO	Random
Performance on FS-TACRED							
RPO	5	26.5 ± 1.19	**29.0** ± 1.37	-	13.9 ± 0.94	13.9 ± 1.06	-
RPO	10	28.3 ± 1.13	**28.9** ± 1.46	27.9 ± 0.98	11.7 ± 0.45	**11.9** ± 0.11	10.9 ± 0.22
EvoPrompt-DE	5	24.6 ± 1.21	24.8 ± 0.99	**24.9** ± 1.12	14.5 ± 0.53	-	14.5 ± 0.59
EvoPrompt-DE	10	24.5 ± 0.98	23.4 ± 0.74	-	10.1 ± 0.42	**10.3** ± 0.50	-
ETGPO	1	28.3 ± 0.75	27.7 ± 0.66	28.2 ± 0.58	13.1 ± 0.32	**14.1** ± 0.58	13.9 ± 0.70
Performance on FS-FewRel							
RPO	5	34.8 ± 0.60	35.8 ± 0.80	**37.4** ± 0.60	30.3 ± 1.14	**32.7** ± 1.12	-
RPO	10	36.3 ± 0.74	**36.9** ± 0.82	-	32.2 ± 1.30	**33.3** ± 1.40	32.0 ± 1.21
EvoPrompt-DE	5	35.9 ± 0.59	-	-	27.9 ± 1.67	**33.7** ± 1.71	-
EvoPrompt-DE	10	-	-	-	35.2 ± 1.64	37.5 ± 1.33	**38.2** ± 1.44
ETGPO	1	35.0 ± 1.06	**35.1** ± 1.04	-	35.2 ± 1.73	-	**36.4** ± 1.73
Performance on MATH-500							
RPO	5	77.7 ± 0.83	77.6 ± 0.41	77.4 ± 0.86	64.9 ± 1.72	**65.6** ± 0.95	-
RPO	10	79.4 ± 0.55	**81.2** ± 1.07	79.8 ± 0.52	64.9 ± 1.01	**67.4** ± 0.84	-
EvoPrompt-DE	5	77.2 ± 1.01	77.4 ± 0.90	**79.0** ± 0.57	66.6 ± 1.37	-	-
EvoPrompt-DE	10	77.7 ± 0.54	77.7 ± 0.79	**79.3** ± 1.12	65.7 ± 0.46	65.6 ± 1.45	**66.6** ± 1.14
ETGPO	1	73.1 ± 1.32	72.6 ± 0.86	**73.4** ± 0.43	67.4 ± 1.17	**68.9** ± 2.07	67.9 ± 0.50

Relation extraction. We sample initial pools of 20,000 pairs for Qwen3-4B and 30,000 for Gemma3-4B. Since more than 90% of the labels are `no_relation`, the balanced gradient set is capped by G and the smallest available TP, TN, FP, or FN group. The selected (S, T, C, H, G) settings are $(5, 5, 5, 0.60, 300)$ for Qwen3-4B and $(3, 5, 5, 0.60, 300)$ for Gemma3-4B.

For GradPO-Prob, we provide context about the full prompt; otherwise, when suggesting tokens for the spans near the beginning of the prompt, the model often generates irrelevant replacement tokens due to lack of context. Specifically, we prepend the original prompt with an instruction to paraphrase it as additional context.

Moreover, we restrict span selection to preserve task-specific terminology used consistently across the static instruction and episode-specific input fields. In particular, spans containing `relation`, `query`, `support`, `yes`, or `no` are split so that these tokens are not edited. Although allowing edits to these terms improves development scores even more, it may introduce incoherent prompts or inconsistencies with the fixed input template; therefore, we keep these terms unchanged to preserve prompt fluency and consistency. We did not need to apply this protected-term constraint to MATH-500 because, unlike relation extraction, it uses a simpler initial prompt without fixed class labels or input-field names that must remain consistent with the task template.

MATH-500. The selected (S, T, C, H, G) settings are $(3, 3, 7, 0.30, 200)$ for Qwen3-4B and $(7, 5, 7, 0.45, 200)$ for Gemma3-4B. The balanced gradient set contains up to 100 correct and 100 incorrect predictions, drawn from initial pools of 1,600 examples for Qwen3-4B and 600 for Gemma3-4B. To avoid repeatedly regenerating long reasoning traces while scoring beam candidates, we keep the source reasoning trace fixed and measure how each local edit changes the likelihood of the gold answer. The five retained beam prompts are then evaluated with freshly generated reasoning on the development set. GradPO-Prob and the random-span control reuse the corresponding GradPO configuration.

Table 7: Dev-set analysis of later iterations of reasoning-based prompt optimization and a one-step GradPO refinement after iteration 10 for relation extraction task. Later reasoning-based iterations still show slight average improvements, but their prompt quality remains unstable across iterations. A single GradPO step, which searches a larger local edit space, usually provides an additional stable improvement over an already strong stage-1 prompt.

	itr.	Qwen3-4B		Gemma3-4B	
Method	range	Avg. F1	Best F1	Avg. F1	Best F1
Performance on FS-TACRED					
RPO	1–10	26.0 ± 3.39	30.3	9.0 ± 3.79	13.8
RPO	11–20	26.1 ± 4.46	**33.6**	9.4 ± 3.79	16.5
RPO+ 1-step GradPO-Gen	after 10	-	31.3	-	**17.1**
EvoPrompt	1–10	21.0 ± 2.56	23.1	13.6 ± 3.07	19.0
EvoPrompt	11–20	22.0 ± 0.68	**24.6**	10.4 ± 0.76	18.0
EvoPrompt+ 1-step GradPO-Gen	after 10	-	23.5	-	**20.3**
Performance on FS-FewRel					
RPO	1–10	22.3 ± 3.02	30.8	14.9 ± 4.62	15.8
RPO	11–20	23.9 ± 3.89	31.1	15.0 ± 3.01	17.7
RPO+ 1-step GradPO-Gen	after 10	-	**31.5**	-	**19.7**
EvoPrompt	1–10	20.8 ± 3.09	24.7	12.8 ± 5.92	18.5
EvoPrompt	11–20	22.3 ± 1.72	**25.1**	15.0 ± 4.75	19.7
EvoPrompt+ 1-step GradPO-Gen	after 10	-	23.3	-	**21.0**

Table 8: Comparison of previous work on FS-TACRED with our best prompt optimization setting using Qwen3-4B. All methods use the same episodic FS-TACRED benchmark in the 1-shot setting. With Qwen3-4B, our two-stage prompt optimization approach improves over the previous best result reported with the same model size.

	FS-TACRED		
	P	R	F1
MNAV (Sabo et al., 2021)	-	-	12.4 ± 1.01
OdinSynth (Vacareanu et al., 2022)	23.5 ± 1.46	11.5 ± 1.02	15.4 ± 1.21
CKPT (Lv et al., 2022)	-	-	15.1 ± 1.12
Anchor+gen. rules (Singh & Blanco, 2024)	19.6 ± 0.63	31.9 ± 1.04	24.2 ± 0.72
SoftRules (Vacareanu et al., 2024)	33.5 ± 1.47	19.7 ± 1.14	24.8 ± 1.22
ICL with Qwen3-4B (Chakma et al., 2026)	26.4 ± 0.69	28.9 ± 1.88	27.6 ± 1.20
RPO+GradPO with Qwen3-4B (Ours)	29.5 ± 1.25	28.5 ± 1.66	**29.0** ± 1.37

GreaTer *Common settings.* Both GreaTer variants use one refinement iteration and fluency weight 0.2 (tuned between 0 to 1). Across both tasks and target models, we tune the maximum balanced gradient-set size $G \in \{100, 200, 300, 500\}$. We use the largest feasible candidate funnel: generate 50 token replacements, retain 25 for task-loss scoring, and evaluate the best five on the validation set.

Relation extraction. For consistency with GradPO, we use the same training sample sizes: 20,000 pairs for Qwen3-4B and 30,000 pairs for Gemma3-4B. We also use the same balanced subset construction, capped at $G = 300$ and limited by the smallest TP, TN, FP, or FN group. Gradients and training losses are computed on this balanced subset.

Moreover, in our relation extraction inference templates, episode-specific input details are appended after the instruction prompt. In contrast, GreaTer prepends the input details in its original prompting setup before generating reasoning and collecting token probabilities. To adapt GreaTer to our task, we provide input context for collecting token suggestions by prepending a simple scenario description with a sample input, similar to GradPO-Prob.

Table 9: Cross-model transfer evaluation on FS-TACRED, where prompts are optimized on one model and evaluated on the other model. R and G denote the numbers of reasoning-based and gradient-based iterations, respectively. The transfer gains are limited, which suggests that GradPO edits are largely model-specific. This is expected because the refinements depend on the target model's loss and gradient signals.

	#itr.	Qwen3-4B (Optimized on Gemma3-4B)			Gemma3-4B (Optimized on Qwen3-4B)		
Method	R+G	P	R	F1	P	R	F1
RPO	5+0	24.4 ± 0.66	30.9 ± 1.05	27.2 ± 0.68	14.7 ± 1.31	04.6 ± 0.51	07.0 ± 0.70
+GradPO	5+1	24.0 ± 0.62	30.4 ± 1.69	26.8 ± 0.97	12.9 ± 0.41	06.2 ± 0.52	08.4 ± 0.55
RPO	10+0	17.5 ± 0.68	35.2 ± 2.02	23.3 ± 1.03	19.5 ± 1.65	02.9 ± 0.36	05.0 ± 0.59
+GradPO	10+1	15.3 ± 0.39	41.1 ± 1.62	22.3 ± 0.62	17.1 ± 1.25	03.4 ± 0.26	05.7 ± 0.42
RPO	5+0	25.6 ± 0.66	51.1 ± 0.93	34.1 ± 0.74	43.9 ± 3.18	12.1 ± 1.15	19.0 ± 1.60
+GradPO	5+1	17.6 ± 0.67	56.5 ± 1.00	26.9 ± 0.87	44.5 ± 2.53	13.6 ± 0.94	20.8 ± 1.27
RPO	10+0	27.3 ± 0.86	52.6 ± 1.50	36.0 ± 1.00	46.2 ± 3.43	07.1 ± 0.96	12.3 ± 1.56
+GradPO	10+1	23.6 ± 1.00	54.9 ± 2.08	33.0 ± 1.30	46.6 ± 4.27	05.8 ± 0.91	10.3 ± 1.53

Table 10: Results with longer runs of gradient-based prompt optimization on FS-TACRED and FS-FewRel. R and G denote the numbers of reasoning-based and gradient-based iterations, respectively. This table shows how much the gradient-based methods improve when they are allowed to run for more iterations after starting from the same initial prompt.

	#itr.	Qwen3-4B			Gemma3-4B		
	R+G	P	R	F1	P	R	F1
Performance on FS-TACRED							
Initial Prompt		17.6 ± 0.44	38.2 ± 1.38	24.0 ± 0.59	10.4 ± 0.56	12.5 ± 0.65	11.3 ± 0.53
RPO	10+0	35.8 ± 1.19	23.4 ± 1.26	28.3 ± 1.13	09.8 ± 0.33	14.4 ± 0.72	11.7 ± 0.45
EvoPrompt	10+0	21.6 ± 0.53	28.4 ± 1.77	24.5 ± 0.98	12.5 ± 0.54	08.5 ± 0.48	10.1 ± 0.42
ETGPO	10+0	21.5 ± 0.54	41.4 ± 1.80	28.3 ± 0.75	12.1 ± 0.43	14.4 ± 0.36	13.1 ± 0.32
GreaTer	0+10	20.5 ± 0.81	28.9 ± 2.07	24.0 ± 1.21	09.0 ± 0.42	11.1 ± 0.65	09.9 ± 0.43
GradPO	0+10	17.5 ± 0.46	40.3 ± 1.17	24.4 ± 0.52	09.8 ± 0.53	10.2 ± 0.66	10.0 ± 0.57
Performance on FS-FewRel							
Initial Prompt		29.9 ± 0.72	44.1 ± 0.76	35.6 ± 0.66	39.9 ± 2.13	18.6 ± 1.12	25.3 ± 1.27
RPO	10+0	33.5 ± 0.98	39.6 ± 0.66	36.3 ± 0.74	34.8 ± 1.39	29.9 ± 1.42	32.2 ± 1.30
EvoPrompt	10+0	35.5 ± 0.71	36.3 ± 0.67	35.9 ± 0.59	37.2 ± 1.33	33.3 ± 1.89	35.2 ± 1.64
ETGPO	10+0	35.6 ± 1.35	34.5 ± 0.93	35.0 ± 1.06	33.8 ± 1.50	36.6 ± 2.14	35.2 ± 1.73
GreaTer	0+10	33.1 ± 0.85	39.8 ± 1.04	36.2 ± 0.85	34.7 ± 1.10	29.7 ± 1.63	32.0 ± 1.39
GradPO	0+10	32.1 ± 0.78	43.7 ± 0.38	37.0 ± 0.63	37.3 ± 0.99	29.2 ± 1.39	32.7 ± 1.17

Table 11: Average edit size across both target models and all three datasets. GradPO updates multiple short spans, balancing GreaTer's one-token edits and LPO's much broader rewrites.

Method	Avg. Spans Updated	Avg. Span Length
LPO	3.43	21.19
GradPO	2.76	3.20
GradPO-Prob	3.15	2.29
GreaTer	1.00	1.00
GreaTer-TG	1.00	1.00

Table 12: Content categories of all top-gradient spans selected by GradPO, including runs that fail development selection. RE contains 75 selected-span occurrences from 19 valid runs, and MATH-500 contains 50 from 10 runs; these are selected edit locations and are not necessarily changed in the final prompt. We manually assign each span to one category based on its role in the prompt.

Span category	Examples	RE	Math	All	%
General task wording	"sentence," "formulas"	38	23	61	48.8
Generic wording	"You are," "Then"	23	3	26	20.8
Output/decision wording	"Answer," "final result"	12	10	22	17.6
Reasoning wording	"concise reasoning," "Think step by"	0	13	13	10.4
Example/relation-specific content	"province," "locker"	2	1	3	2.4
Total		**75**	**50**	**125**	**100.0**

MATH-500. The initial training pools contain 1,600 examples for Qwen3-4B and 600 for Gemma3-4B. We select $G = 200$ for Qwen3-4B and $G = 300$ for Gemma3-4B as best settings; GreaTer-TG uses the same model-specific settings.

LPO *Common settings.* LPO uses one refinement iteration and five generated candidate prompts. Across both tasks and target models, we tune the maximum number of incorrect feedback examples in $\{3, 5\}$, maximum marked spans for editing in $\{3, 5\}$, and maximum words per marked span in $\{3, 5\}$. These limits are instructions in the meta-prompt rather than hard truncation constraints, and candidates are evaluated using Equation 2.

Relation extraction. For LPO, we use the same balanced subset construction as GradPO and GreaTer. The selected maxima for feedback examples, spans, and words per span are $(3, 3, 3)$ for Qwen3-4B and $(3, 5, 3)$ for Gemma3-4B.

MATH-500. We use a 512-example training pool. The selected maxima for feedback examples, spans, and words per span are $(3, 3, 3)$ for Qwen3-4B and $(3, 5, 3)$ for Gemma3-4B.

B.2 Task formulations

Episodic few-shot relation extraction. We use the standard N-way M-shot episodic setting and focus on one-shot evaluation ($M = 1$). Each episode contains N candidate relations, one support example and a description for each relation, and a query sentence with a subject–object entity pair. We formulate the episode as N binary decisions. For each candidate, the model receives its relation name, description, support sentence, and the query, and predicts `yes` or `no`. The model does not generate a reasoning trace for these decisions ($r = \emptyset$). We then recover the final episodic prediction $\hat{y}$: if multiple relations are positive, we select one at random; if none is positive, we output `no_relation`. The optimized prompt is the shared instruction used for every binary decision.

Mathematical reasoning. Each MATH-500 input is a mathematical problem, and the model is prompted to generate a step-by-step reasoning trace r followed by a final answer $\hat{y}$. The optimized prompt is the shared instruction that specifies how the model should reason and format this output. Prompt selection uses the correctness of $\hat{y}$ on validation data, and test performance is measured from the extracted $\hat{y}$;

B.3 Dataset split details

For FS-TACRED, the train sets and validation sets are created from the corresponding original train and validation splits. For FS-FewRel, the train sets and validation sets are created from the original FewRel (Han et al., 2018) train split, because the test episodes are created from the original validation split. The original FewRel train split contains 64 unique relations, which we divide into 54 train relations and 10 validation relations. For both datasets, we create the validation set as three sets of $3{,}000$ episodes.

For MATH-500, we remove one duplicate from the 7,500-problem official MATH training split and use seed 42 to create 5,999 training and 1,500 validation examples. For second-stage tuning and selection, we use fixed 300-example subsets from each of three validation folds, for 900 examples in total. Prompts are selected using Equation 2. The test set is the untouched 500-problem MATH-500 split and questions do not overlap across the processed training, validation, and test sets.

B.4 Entity-match Filtering for Relation Extraction

For relation extraction only, following Chakma et al. (2026), we employ entity-match filtering during test inference. Before relation matching, we prompt the model to check whether the subject and object entities match the entity types required by the candidate relation. If this check fails, we skip relation matching for that candidate and directly assign a negative prediction. As shown in prior work, this filtering improves F1 by reducing invalid positive predictions.

B.5 Decoding and inference Details

For relation inference and entity-type checking, we formulate prediction as binary classification using the logits of the first generated token, restricted to "yes" and "no". We predict "yes" when $P(\texttt{yes}) \geq P(\texttt{no})$, and "no" otherwise.

For generation-based components, we use the default decoding configuration of each model. For Qwen models, we use sampling with temperature 0.6, top-$p = 0.95$, and top-$k = 20$. For Gemma models, we use sampling with temperature 1.0, top-$p = 0.95$, and top-$k = 64$. We do not truncate input prompts, and set the max tokens for generation to $10{,}000$ for any prompting calls.

For MATH-500 answer generation, we instead use a 4,096-token output limit and instruct the model to place its final answer inside `<answer>...</answer>`. We extract the last complete answer block, or falling back to find the last `\boxed{...}` expression if answer tags are missing; outputs with neither format are counted as incorrect. The primary metric is accuracy using the PRM800K `grade_answer` implementation (Lightman et al., 2023). We report the mean and sample standard deviation over seeds 42, 1, 100, 1000, and 10000, evaluating every selected prompt on the same 500 problems.

Significance testing. For both task types, we use a one-sided paired bootstrap with 10,000 resamples. For relation extraction, we resample 150,000 paired query predictions from five sets of 10,000 episodes with three queries per episode; each query prediction is recovered from five binary relation decisions. For MATH-500, we resample 2,500 paired outputs from 500 problems evaluated with five decoding seeds.

B.6 Additional Qualitative Analysis

GradPO explores a richer local prompt space with limited additional cost. Most computation comes from scoring candidate prompts during beam search and evaluating the retained prompts on the validation set. The few prompting calls used to generate replacements and apply span edits are minor in comparison. The dominant training-side cost is beam-candidate scoring, which requires

$$GC\left[1 + B(S-1)\right], \tag{3}$$

LLM forward calls, where G is the gradient-set size, C is the number of candidates per span, B is the beam width, and S is the number of selected spans. For relation extraction, our largest setting for Qwen3-4B uses $G = 300$, $C = 5$, $B = 5$, and $S = 5$, requiring 31,500 forward calls, compared with 7,500 for the 25 candidates scored by GreaTer. Evaluating five retained prompts on 9,000

development episodes, each requiring five binary predictions, costs 225,000 forward calls for either method. Relative to this validation cost, candidate search adds 14.0% for GradPO and 3.3% for GreaTer. On MATH-500, all refiners evaluate at most five prompts on 900 development problems; with a maximum reasoning length of 4,096 tokens, this requires up to 18.4 million autoregressive decoding steps per method, each corresponding to one forward call. Candidate scoring does not generate a new solution: for each candidate and example, one forward call computes the likelihood of the gold answer using the same reasoning trace (See the details of GradPO on Math-500 in the Appendix B.1). For Qwen3-4B, $G = 200$, $C = 7$, $B = 5$, and $S = 3$, requiring 15,400 forward calls, or 0.08% of this upper bound. For Gemma3-4B, $G = 200$, $C = 7$, $B = 5$, and $S = 7$, requiring 43,400 calls, or 0.24%; the corresponding GreaTer costs are 5,000 and 7,500 calls (0.03% and 0.04%). Beyond the shared validation cost, LPO uses one location-selection generation and five short rewrite generations with a separate optimizer model, which are also minor relative to validation. Overall, across both tasks, GradPO explores combinations of several span edits with limited overhead relative to the shared validation cost.

Most GradPO gains come in the first few iterations. GradPO usually improves the prompt in the early iterations, and the first iteration often gives the largest gain. Later iterations can still help, but the additional improvements are usually small. This is also consistent with the main results tables, where one GradPO step already improves the stage-1 prompt in most cases. In Table 10, across the four 10-iteration GradPO runs with Gemma3-4B and Qwen3-4B on FS-TACRED and FS-FewRel, two runs reach their best score at the first iteration, one reaches it at the third iteration, and one reaches it at the seventh iteration. Overall, GradPO works best as a fast local refiner, where most of the benefit comes from the first few updates.

Table 13: Average instruction-prompt length before and after refinement across all valid attempts on FS-TACRED, FS-FewRel, and MATH-500. GradPO keeps prompt length nearly unchanged; GreaTer has no length change because it replaces one tokenizer token with one token, whereas LPO produces longer prompts in our experiments. Counts use the target model's tokenizer, and Δ denotes after minus before.

Target Model	Refiner	Before	After	Δ
Qwen3-4B	GradPO	223.93	224.07	+0.14
	GradPO-Prob	223.93	220.93	-3.00
	LPO	223.93	242.29	+18.36
	GreaTer	223.93	223.93	0.00
	GreaTer-TG	223.93	223.93	0.00
Gemma3-4B	GradPO	189.13	186.07	-3.07
	GradPO-Prob	189.13	188.67	-0.47
	LPO	189.13	239.40	+50.27
	GreaTer	189.13	189.13	0.00
	GreaTer-TG	189.13	189.13	0.00

GradPO improves prompts without increasing their length. Table 13 compares prompt lengths before and after refinement across all three datasets. Across all valid attempts, GradPO keeps the average prompt length nearly unchanged for both Qwen3-4B and Gemma3-4B. In contrast, LPO produces substantially longer prompts, adding about 18 and 50 tokens, respectively. The GreaTer variants preserve length by construction because they replace one tokenizer token at a time. Thus, GradPO preserves the length and structure of the stage-1 prompt while making small local edits.

GradPO edits are model-specific and transfer weakly across models. Table 9 shows that cross-model transfer is not very effective for GradPO. Only 3 out of 8 transferred cases improve when applying the refined prompt to a different target model. This is expected because GradPO relies on model-specific losses and gradient signals. As a result, local edits that help one model do not always transfer well to another.

C EXAMPLES COMPARISON OF SECOND-STAGE REFINEMENT

Table 14: Example edits from second-stage refinements of the same RPO prompt after five first-stage iterations using Qwen3-4B on FS-TACRED. The examples highlight different editing behaviors: GreaTer can produce locally inconsistent single-token substitutions, while LPO makes larger rewrites in this example. In contrast, GradPO makes short, targeted local edits. Full prompts appear in Appendix E.

Method	**Original fragment**	**Refined fragment**
GreaTer	"You are given a relation name ..."	"I are given a relation name ..."
LPO	"Confirm that the query sentence explicitly links the Subject and Object as per the relation's description."	"Confirm that the query sentence explicitly states the link between the Subject and Object as per the relation's description, using direct assertions (e.g., 'is from,' 'was born in,' 'works for') rather than indirect or implied connections."
GradPO	"a description of the relation in brackets, a support sentence exemplifying the relation"	"a definition of the relation in bracketed text, a support example sentence exemplifying the relation"

D INPUT PROMPT FORMAT

We construct the full input prompt as three components: the `Instruction Prompt`, the `Answer Instruction Prompt`, and the `Input Details Prompt`. The `Instruction Prompt` defines the task and decision guidelines, the `Answer Instruction Prompt` specifies the required output format in one simple sentence, and the `Input Details Prompt` contains the task-specific input: the relation name, relation description, support sentence, and query sentence for relation extraction, or a MATH-500 question.

During optimization, we modify only the `Instruction Prompt`. At inference time, the final prompt is formed by concatenating the three components with newline separators:

```
```
{Instruction Prompt}
{Answer Instruction Prompt}
{Input Details Prompt}
```
```

The complete set of prompts used in our framework, including input prompts, meta-prompts, and optimized prompts, is provided in Appendix E.

E PROMPTS

We include the initial prompts, EvoPrompt-DE seed prompts, and optimizer meta-prompts for relation extraction and MATH-500. To avoid repetition and an unnecessarily long PDF, we show three first-stage cases per model and task, one each from RPO, EvoPrompt-DE, and ETGPO, together with their available second-stage refinements.

E.1 RELATION EXTRACTION INPUT PROMPT TEMPLATES

Instruction prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.

A relation connects the Subject and the Object entities.  The Subject
and the Object entities are indicated with the subject and object
tags, respectively.
You need to decide whether the relation holds between the Subject and
the Object entities in the query sentence.

If the relation holds between the Subject and the Object entities in
the query sentence, answer "yes"; otherwise, answer "no".
```

Answer instruction prompt

```
Output only "yes" or "no" as answer, with no explanation or
additional text.
```

Input details prompt

```
==

Relation name:  "#RELATION#" (#RELATION_DESCRIPTION#)

Support sentence:  #SUPPORT_SENTENCE#

Query Sentence:  #QUERY_SENTENCE#

Answer:
```

E.2 MATH-500 INPUT PROMPT TEMPLATES

Both target models use the same initial instruction and answer-format template.

Initial instruction prompt

```
Solve the following math problem.  Think step by step carefully
before answering.
```

Answer instruction prompt

```
After you finish reasoning, output the final answer exactly once
between the tags <answer> and </answer>.  Put only the final answer
inside the tags, using LaTeX notation when needed.
```

Input details prompt

```
Question:
#QUESTION#
```

E.3 EvoPrompt-DE seed prompts

For each task, the five-prompt initial population contains the initial instruction and the four seed variants below.

E.3.1 Relation extraction

The initial instruction is shown in Appendix E.1.

Prompt 1

```
Decide whether the given relation holds between the tagged Subject
and Object entities in the query sentence.  The support sentence is
an example of the relation.  Answer "yes" if the relation holds in
the query; otherwise answer "no".
```

Prompt 2

```
You are given a relation name, its description, a support sentence
that exemplifies the relation, and a query sentence.  Determine
whether the query sentence expresses that exact relation between the
tagged Subject and Object entities.  Check the relation definition,
the semantic type of each entity, the direction from Subject to
Object, and whether the evidence is explicit or strongly implied.
Treat the support sentence as an example of the relation, not as
evidence about the query.  If the relation holds in the query
sentence, answer "yes"; otherwise answer "no".
```

Prompt 3

```
Read the relation description, then read the query sentence.  The
Subject and Object are already marked with tags.  Decide if the query
says that this relation is true for that Subject and Object.  The
support sentence is only an example to help you understand the
relation.  Answer only "yes" or "no".
```

Prompt 4

```
Perform binary relation verification for the query sentence.  First
identify the tagged Subject and Object, then compare their roles to
the relation name and definition.  Confirm that the relation
direction is Subject-to-Object, that the Object has the required type
or value for the relation, and that the query sentence itself
supplies sufficient explicit or contextually implied evidence.
Ignore distracting entities and do not transfer facts from the
support sentence beyond its illustration of the relation schema.
Output "yes" exactly when the relation holds; otherwise output "no".
```

E.3.2 MATH-500

The initial instruction is shown in Appendix E.2.

Prompt 1

```
Solve this in the simplest way you can and give the final result
clearly.
```

Prompt 2

```
Read the problem carefully and reason through it one step at a time.
Keep the solution simple, check the important calculations, and don't
overthink much by unnecessary details.  Give the final answer clearly
at the end.
```

Prompt 3

```
Reason through the problem before answering.  Keep the explanation
concise and state the final answer clearly.
```

Prompt 4

```
First identify the type of mathematical reasoning required by the
problem, such as arithmetic, algebra, geometry, probability, or
number reasoning.  Then derive the answer carefully from the
information provided, showing enough intermediate reasoning to make
the solution reliable.  Avoid unnecessary or overly complicated
methods.  End with the exact final answer requested.
```

E.4 Relation extraction meta-prompts

E.4.1 GradPO meta-prompts

Replacement generation prompt

```
You are an expert on suggesting replacements for targeted spans in a
prompt.

The editable spans are marked using tags such as <span_1>...</span_1>,
<span_2>...</span_2>, and so on.

Input Prompt with editable spans:
'''
#MARKED_PROMPT#
'''

Editable spans:
#REGION_CANDIDATE_REQUEST_BLOCKS#

Task:
Suggest #NUM_CANDIDATES# replacements for each editable span so that
every replacement fits naturally in context.

Rules:
- Preserve the meaning and role of each editable span
- Treat each tagged span as a single unit
- Return replacements only for the editable spans
- A candidate may be identical to the original span text if keeping
it unchanged is the best option

Output only the replacements in the following JSON format:
'''json
{
''span_1'':  {
''candidates'':  [
''replacement 1'',
''...'',
''replacement #NUM_CANDIDATES#''
]
```

```
},
...
''span_#NUM_REGIONS#'': {
''candidates'': [
''replacement 1'',
''...'',
''replacement #NUM_CANDIDATES#''
] } }
```

Span replacement prompt

```
You are an expert prompt generator for a relation extraction
inference task.

A relation captures the connection between two entities in a sentence
by describing their relationship. We will refer to these entities as
the subject and object entities.
The task requires inferring a binary (yes/no) answer based on whether
the query sentence expresses this relation between the subject and
the object entities.

You are given the current instruction prompt with targeted spans
below:
'''
#ALL_MARKED_PROMPT#
'''
Each editable span is marked with tags such as <span_1>...</span_1>,
<span_2>...</span_2>, and so on.

Replacements for each spans are given below:
#SELECTED_REPLACEMENTS#

Your task is to generate a revised instruction prompt by applying
the given replacements to the corresponding spans.
Use the replacements exactly as provided, except for minimal local
adjustments if necessary for spelling and grammatical correctness or
coherence.
Do not modify any other parts of the prompt (but remove the span
tags).

Output only the revised prompt.
```

Additional context prepended for GradPO-Prob token generation

```
Original Prompt:
'''#FULL_INSTRUCTION_PROMPT#'''

Task:
Write a natural revised version of the original prompt while
preserving meaning, structure, and tone. Prefer paraphrase or other
clear, robust, and effective wording.

Revised prompt:
'''#PROMPT_PREFIX_TEXT#
```

E.4.2 LPO META-PROMPTS

Span identification prompt

```
A relation extraction prompt helps an LLM decide whether a query
sentence expresses a target relation between the subject and object
entities, using a support sentence for that relation.  The classifier
must answer with exactly one token:  ``yes'' or ``no''.

Current prompt:
```
#INFERENCE_PROMPT#
```

Feedback examples from the current prompt:
#FEEDBACK_EXAMPLES#

First, think about the issues with the prompt.
Identify the scope of tokens within the prompt where edits should
take place.
Prompt edits include adding, deleting or modifying tokens.
Mark the scope of the prompt that needs editing by putting <edit>,
</edit> tags.
You can have multiple <edit> tags and each <edit> tag should not
entail more than #MAX_WORDS_PER_EDIT_TAG# words.
Use at most #MAX_EDIT_TAGS# <edit> tags.
Do not cover the whole sentence with multiple <edit> tags.

Reply with the full prompt enclosed within <p> and </p> tags, with
the edit locations marked using <edit> and </edit> tags.
Do not include any other text.
```

Span replacement prompt

```
A relation extraction prompt helps an LLM decide whether a query
sentence expresses a target relation between the subject and object
entities, using a support sentence for that relation.  The classifier
must answer with exactly one token:  ``yes'' or ``no''.

Current prompt with local edit scopes:
```
#TAGGED_PROMPT#
```

Feedback examples from the current prompt:
#FEEDBACK_EXAMPLES#

Generate one revised full prompt by editing only the text inside the
<edit>, </edit> tags.
You may add, delete, or modify tokens inside those local scopes.
Keep the rest of the prompt unchanged except for minimal grammar
cleanup at the edit boundaries.
Keep the <edit> and </edit> tags around the edited local scopes in
the revised prompt.

Please reason through the problem, but output only the revised prompt
enclosed within the <p> and </p> tags.
```

E.4.3 GREATER META-PROMPTS

Additional prepended context for token generation

```
You are optimizing an instruction prompt for a binary relation
extraction classifier.

The instruction prompt will appear before the classifier input.
Later, the classifier will receive an input similar to the following
example and must decide whether the target relation is expressed
between the Subject and Object entities in the query sentence.

The classifier must answer with exactly one token:  ``yes'' or
``no''.

Example classifier input:

Relation name:  #RELATION# (#RELATION_DESCRIPTION#)
Support Sentence:  #SUPPORT_SENTENCE#
Query Sentence:  #QUERY_SENTENCE#

Write a instruction prompt that should appear before this type of
input and help the classifier solve the relation extraction problem.

Instruction:
```

E.4.4 ETGPO META-PROMPTS

First batch taxonomy creation prompt

```
You are an expert at analyzing why language models fail on relation
extraction tasks with binary yes/no inference.

#FAILURES_TEXT#

## Your Task

Analyze each failure and identify the root cause of each error.
Focus on the underlying failure mechanism.
Important:  the reasoning field is post-hoc feedback describing the
most likely cause of the incorrect binary prediction, not a verbatim
chain-of-thought from the original model.  Use it as probabilistic
evidence together with the input, gold label, and wrong answer.

For each failure, find:
1.  The EARLIEST point in the reasoning where something went wrong
2.  What specifically went wrong (calculation error, wrong approach,
misunderstanding, etc.)
3.  Why this error led to the wrong final answer

Create issue categories that capture each type of error.  Categories
should be general enough to potentially apply to other traces, but
specific enough to be meaningful, without becoming tied to relation-
or example-specific details.

IMPORTANT: Each category must be SELF-CONTAINED and understandable
by someone who has NOT seen the original problems.

## Output Format

Return a JSON object with:
```

```
```json
{
''categories'': [
{
''category_name'': ''Short descriptive name for this type of error'',
''summary'': ''One sentence describing the core error pattern.'',
''description'': ''Describe the general failure mechanism in this
category.'',
''example'': ''A concrete, self-contained example. Format:
'Problem: [simple problem]. Error: [what the model does wrong].
Correct: [what should happen].''',
''error_type'': ''Type of error (e.g., Calculation Error, Wrong
Approach, Conceptual Misunderstanding, Missing Step, Logical Fallacy,
Factual Error, Incomplete Reasoning, Misreading the Problem)'',
''why_leads_to_wrong_answer'': ''Explanation of how this error causes
wrong answers''
}
],
''failure_assignments'': [
{
''failure_id'': 1,
''problem_idx'': <problem_idx>,
''run_id'': <run_id>,
''category_name'': ''Name of the category this failure belongs to'',
''trace_details'': {
''trace_specific_location'': ''Where in the reasoning the error
occurred'',
''trace_specific_details'': ''Specific details about what went
wrong''
} } ] }
```
```

Subsequent batch taxonomy creation prompt

```
You are an expert at analyzing why language models fail on relation
extraction tasks with binary yes/no inference.

## Existing Issue Categories

#EXISTING_CATEGORIES_TEXT#

#FAILURES_TEXT#

## Your Task

For each failure:
1. Determine if the error fits one of the EXISTING categories
2. OR create a NEW category if the error is fundamentally different
Important: the reasoning field is post-hoc feedback describing the
most likely cause of the incorrect binary prediction, not a verbatim
chain-of-thought from the original model. Use it as probabilistic
evidence together with the input, gold label, and wrong answer.

New categories should stay general and reusable, rather than becoming
tied to relation- or example-specific details.

## Output Format

Return a JSON object with:

```json
{
```
```

```
''new_categories'': [
{
''category_name'': ''Short descriptive name for NEW error type'',
''summary'': ''One sentence describing the core error pattern.'',
''description'': ''Describe the general failure mechanism in this
category.'',
''example'': ''A concrete example.'',
''error_type'': ''Type of error (e.g., Calculation Error, Wrong
Approach, Conceptual Misunderstanding, Missing Step, Logical Fallacy,
Factual Error, Incomplete Reasoning, Misreading the Problem)'',
''why_leads_to_wrong_answer'': ''Explanation of how this error causes
wrong answers''
}
],
''failure_assignments'': [
{
''failure_id'': 1,
''problem_idx'': <problem_idx>,
''run_id'': <run_id>,
''is_new_category'': false,
''category_name'': ''Name of existing or new category'',
''trace_details'': {
''trace_specific_location'': ''Where in the reasoning the error
occurred'',
''trace_specific_details'': ''Specific details about what went
wrong''
} } ] }
'''

Note: ''new_categories'' should only contain categories that don't
exist yet.
```

Guidance generation prompt

```
You are an expert at improving language model performance on relation
extraction with binary yes/no inference.

I have identified the following recurring error categories from model
failures. Use them to revise the current instruction prompt so the
model can avoid these mistakes.

## Current Instruction Prompt

'''text
#CURRENT_INSTRUCTION_PROMPT#
'''

## Error Categories

'''text
#ERROR_CATEGORIES_TEXT#
'''

## Your Task

Generate a revised instruction prompt that:
1. Addresses each failure category with specific, actionable advice
2. Improves and generalizes binary relation extraction decisions

## Critical Constraints

- Do not revise the instruction in a way that requires step-by-step
```

```
reasoning or explanatory output; the task should remain direct binary
yes/no inference.
- Preserve compatibility with the existing prompt structure where the
answer instruction and input template are appended separately.
- The goal is ACCURACY, not caution.  Never generate guidance that
encourages the model to refuse, abstain, or say ``not specified''
when an answer can be provided.

## Output Format

Return a JSON object with:
{
``revised_instruction_prompt'':  ``the revised instruction prompt
text''
}
```

E.4.5 RPO META-PROMPTS

Feedback generation prompt

```
You are an expert feedback model for a relation extraction inference
task.  Specifically, you are skilled at providing feedback explaining
why a relation extraction system arrived at a particular yes/no
decision, for both correct and incorrect predictions.

A relation captures the connection between two entities in a sentence
by describing their relationship.  We will refer to these entities as
the subject and object entities.
The task requires inferring a binary (yes/no) answer based on whether
the query sentence expresses this relation between the subject and
the object.

You are given an input instance for relation inference below that
contains:
- A relation and its description
- A support instance (an example where the relation holds)
- A query sentence
- A ground-truth label (yes/no)
- A yes/no inference made by another LLM on this instance

The ground-truth label indicates whether the LLM inference was
correct or incorrect and is provided only as contextual information.
The feedback model's task is to explain the most likely reasoning
process that led to the model's answer, not to re-evaluate, judge, or
correct the prediction.  In particular:
- If the prediction matches the label, explain what cues or evidence
likely led to that choice.
- If it does not match, explain what misunderstanding, missing
evidence, or heuristic likely caused the prediction.

Instance:
```
Relation: #RELATION#
Relation Description: #RELATION_DESCRIPTION#
Support Instance: #SUPPORT_INSTANCE#

Query: #QUERY#

Label: #LABEL#
LLM Inference: #INFERENCE#
```
```

```
Please reason through the problem, but provide your final feedback
only within the <f> and </f> tags.
```

Mutation prompt

```
You are an expert prompt generator for a relation extraction
inference task.  You specialize in revising and improving prompts
based on feedback from previous model predictions.

A relation captures the connection between two entities in a sentence
by describing their relationship.  We will refer to these entities as
the subject and object entities.
The task requires inferring a binary (yes/no) answer based on whether
the query sentence expresses this relation between the subject and
the object.

You are given below a prompt that is used by another LLM to make an
inference for the task:
'''
#INFERENCE_PROMPT#
'''

Using this prompt, another LLM was tested on three instances of the
task.  Below, you are given the inputs, the inference made by the
other LLM, and feedback for each task.
'''
Task 1
Relation:  #RELATION_1#
Relation Description:  #RELATION_DESCRIPTION_1#
Support Instance:  #SUPPORT_INSTANCE_1#
Query Sentence:  #QUERY_1#
Ground-Truth Label:  #LABEL_1#
LLM Inference:  #INFERENCE_1#
Feedback:  #FEEDBACK_1#

Task 2
Relation:  #RELATION_2#
Relation Description:  #RELATION_DESCRIPTION_2#
Support Instance:  #SUPPORT_INSTANCE_2#
Query Sentence:  #QUERY_2#
Ground-Truth Label:  #LABEL_2#
LLM Inference:  #INFERENCE_2#
Feedback:  #FEEDBACK_2#

Task 3
Relation:  #RELATION_3#
Relation Description:  #RELATION_DESCRIPTION_3#
Support Instance:  #SUPPORT_INSTANCE_3#
Query Sentence:  #QUERY_3#
Ground-Truth Label:  #LABEL_3#
LLM Inference:  #INFERENCE_3#
Feedback:  #FEEDBACK_3#
'''

Carefully read the inputs, outputs, and feedback to identify problems
with the current prompt.
Your task is to generate a revised version of the prompt that helps
the other LLM generalize better when using it.
You may modify, add to, or remove any instructions or content in
the current prompt in order to improve the prediction and enhance
generalization.
```

```
Please reason through the problem, but output only the revised prompt
enclosed within the <p> and </p> tags.
```

E.4.6 EvoPrompt-DE meta-prompts

DE operation prompt

```
Please follow the instruction step-by-step to generate a better
prompt.
1.  Identify the different parts between Prompt 1 and Prompt 2:
Prompt 1:  Determine whether the named relation holds between the
tagged subject and tagged object in the query sentence.  Use the
support sentence only as an example of the relation.
Prompt 2:  Decide if the query sentence expresses the requested
relation from Subject to Object.  Check the relation definition,
entity roles, direction, and whether the evidence is explicit or
clearly implied.
2.  Randomly mutate the different parts.
3.  Combine the different parts with Prompt 3, selectively replace it
with the different parts from step 2, and generate a new prompt.
Prompt 3:  Carefully compare the relation definition with the query
sentence, verify that the tagged Subject and Object have the right
semantic types and direction, and avoid using irrelevant facts from
the support example.
4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with #PROMPT_OPEN_TAG# and
#PROMPT_CLOSE_TAG#:
Basic Prompt:  You are given a relation name, a relation description,
a support sentence that exemplifies the relation, and a query
sentence.  Answer only yes or no depending on whether the relation
holds between the tagged Subject and Object in the query sentence.

1.  Identifying the different parts between Prompt 1 and Prompt 2:
Prompt 1:  Determine whether the named relation holds between the
tagged subject and tagged object in the query sentence.  Use the
support sentence only as an example of the relation.
Prompt 2:  Decide if the query sentence expresses the requested
relation from Subject to Object.  Check the relation definition,
entity roles, direction, and whether the evidence is explicit or
clearly implied.
Different parts:
''named relation holds'' vs ''query sentence expresses the requested
relation''
''tagged subject and tagged object'' vs ''Subject to Object''
''Use the support sentence only as an example'' vs ''Check the
relation definition, entity roles, direction''

2.  Randomly mutate the different parts:
''named relation holds'' -> ''specified relation is valid''
''query sentence expresses the requested relation'' -> ''query
provides enough evidence for the target relation''
''tagged subject and tagged object'' -> ''marked Subject and Object
entities''
''Check the relation definition, entity roles, direction'' ->
''verify the definition, argument roles, semantic types, and
direction''

3.  Combine the different parts with Prompt 3, selectively replace
it with the different parts in step 2 and generate a new prompt:
Prompt 3:  Carefully compare the relation definition with the query
sentence, verify that the tagged Subject and Object have the right
```

```
semantic types and direction, and avoid using irrelevant facts from
the support example.
New Prompt:  Carefully compare the relation definition with the query
sentence and decide whether the specified relation is valid for the
marked Subject and Object entities.  Verify the definition, argument
roles, semantic types, and direction, and use the support sentence
only as a guide to the relation type.

4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with #PROMPT_OPEN_TAG# and
#PROMPT_CLOSE_TAG#:
Basic Prompt:  You are given a relation name, a relation description,
a support sentence that exemplifies the relation, and a query
sentence.  Answer only yes or no depending on whether the relation
holds between the tagged Subject and Object in the query sentence.
Final Prompt:  #PROMPT_OPEN_TAG#Carefully compare the relation
definition with the query sentence and decide whether the specified
relation is valid for the tagged Subject and Object entities.  Use
the support sentence only as an example of the relation type.  Verify
the argument roles, semantic types, direction of the relation, and
whether the query gives explicit or clearly implied evidence.  If the
relation holds between the Subject and Object in the query sentence,
answer ``yes''; otherwise, answer ``no''.#PROMPT_CLOSE_TAG#

Please follow the instruction step-by-step to generate a better
prompt.
1.  Identify the different parts between Prompt 1 and Prompt 2:
Prompt 1:  <prompt1>
Prompt 2:  <prompt2>
2.  Randomly mutate the different parts.
3.  Combine the different parts with Prompt 3, selectively replace it
with the different parts in step 2, and generate a new prompt.
Prompt 3:  <prompt3>
4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with #PROMPT_OPEN_TAG# and
#PROMPT_CLOSE_TAG#:
Basic Prompt:  <prompt0>

1.
```

E.5 MATH-500 META-PROMPTS

E.5.1 GRADPO META-PROMPTS

For replacement generation, MATH-500 uses the same meta-prompt as relation extraction (Appendix E.4.1).

Span replacement prompt

```
You are an expert prompt generator for a mathematical problem-solving
task.
A mathematical problem may require numerical, algebraic, geometric,
combinatorial, or symbolic reasoning.
The task requires deriving the solution carefully and then outputting
an exact final answer.
You are given the current instruction prompt with targeted spans
below:
```
#ALL_MARKED_PROMPT#
```
```

```
Each editable span is marked with tags such as <span_1>...</span_1>,
<span_2>...</span_2>, and so on.
Replacements for each spans are given below:
#SELECTED_REPLACEMENTS#
Your task is to generate a revised instruction prompt by applying the
given replacements to the corresponding spans.
Use the replacements exactly as provided, except for minimal local
adjustments if necessary for spelling and grammatical correctness or
coherence.
Do not modify any other parts of the prompt (but remove the span
tags).
*** Remove the span tags (e.g., <span_1>...</span_1>,
<span_2>...</span_2>, and so on.)  from the revised prompt.  ***
Output the complete revised instruction prompt exactly once in the
following format:
<prompt>
[complete revised instruction prompt]
</prompt>
Do not output anything outside the <prompt> tags.
```

E.5.2 LPO META-PROMPTS

Span identification prompt

```
A mathematical problem may require numerical, algebraic, geometric,
combinatorial, or symbolic reasoning.
The task requires deriving the solution carefully and then outputting
an exact final answer.
Current prompt:
```
#INFERENCE_PROMPT#
```
Feedback examples from the current prompt:
#FEEDBACK_EXAMPLES#
First, think about the issues with the prompt.
Identify the scope of tokens within the prompt where edits should
take place.
Prompt edits include adding, deleting or modifying tokens.
Mark the scope of the prompt that needs editing by putting <edit>,
</edit> tags.
Do not rewrite, add, or delete any prompt text at this stage; only
insert the <edit> and </edit> tags into the exact current prompt.
You can have multiple <edit> tags and each <edit> tag should not
entail more than #MAX_WORDS_PER_EDIT_TAG# words.
Use at most #MAX_EDIT_TAGS# <edit> tags.
Do not cover the whole sentence with multiple <edit> tags.
Reply with the full prompt enclosed within <p> and </p> tags, with
the edit locations marked using <edit> and </edit> tags.
Do not include any other text.
```

Span replacement prompt

```
A mathematical problem may require numerical, algebraic, geometric,
combinatorial, or symbolic reasoning.
The task requires deriving the solution carefully and then outputting
an exact final answer.
Current prompt with local edit scopes:
```
#TAGGED_PROMPT#
```
```

```
```

Feedback examples from the current prompt:
#FEEDBACK_EXAMPLES#

Generate one revised full prompt by editing only the text inside the <edit>, </edit> tags.
You may add, delete, or modify tokens inside those local scopes.
Keep the rest of the prompt unchanged except for minimal grammar cleanup at the edit boundaries.
Keep the <edit> and </edit> tags around the edited local scopes in the revised prompt.

Please reason through the problem, but output only the revised prompt enclosed within the and tags.
```

### E.5.3 RPO META-PROMPTS

**Feedback generation prompt**

```
You are an expert feedback model for a mathematical problem-solving task. You specialize in explaining why an LLM arrived at a particular answer, for both correct and incorrect predictions.

A mathematical problem may require numerical, algebraic, geometric, combinatorial, or symbolic reasoning.
The task requires deriving the solution carefully and then outputting an exact final answer.

You are given one task instance containing the question, ground-truth answer, the LLM's reasoning, its selected answer, and whether that answer was correct or incorrect.

The ground-truth answer and outcome are provided as contextual information. Your task is to explain the most likely reasoning or decision process that led to the LLM's answer, not to solve the question again.

Analyze the reasoning in the LLM response and explain how it led to the selected answer.
- If the answer is correct, explain which reasoning steps, evidence, or cues were useful.
- If the answer is incorrect, explain which reasoning step, misunderstanding, missing evidence, or heuristic likely caused the error.
- If reasoning reaches the output-token limit without a final answer, classify it as "reasoning token limit exceeded" and identify unnecessary repetition, checking, or detours that could be shortened. Distinguish this from an incorrect final answer or a missing final answer below the limit; do not assume that the ground-truth answer or outcome is mislabeled.

Instance:
```
#MATH_EXAMPLE#
```

Please reason through the problem, but provide your final feedback only inside <feedback> and </feedback>.
```

**Prompt revision prompt**

```
You are an expert prompt generator for a mathematical problem-solving task. You specialize in revising and improving prompts based on feedback from previous model predictions.

A mathematical problem may require numerical, algebraic, geometric, combinatorial, or symbolic reasoning.
```
```

```
The task requires deriving the solution carefully and then outputting
an exact final answer.
You are given below a prompt that is used by another LLM to answer
the questions:
```
#CURRENT_INSTRUCTION_PROMPT#
```
Using this prompt, another LLM was tested on 3 task instances.
Below, you are given the inputs, responses, and feedback for each
task.
```
#FEEDBACK_EXAMPLE_1#
#FEEDBACK_EXAMPLE_2#
#FEEDBACK_EXAMPLE_3#
```
Carefully read the inputs, outputs, and feedback to identify problems
with the current prompt.
Your task is to generate a revised version of the prompt that helps
the other LLM generalize better when using it.
You may modify, add to, or remove any instructions or content in the
current prompt to improve prediction and generalization.
Revise only the task instruction and task details.  Do not add
answer-format instructions, answer tags, input placeholders, or a
model-response template; these are handled separately from the prompt
being optimized.
For token-limit failures, encourage concise, focused reasoning that
reaches a final answer within the available budget; avoid unnecessary
repeated verification.  Keep guidance general rather than specific to
the example problems.
Please reason through the problem, but output only the revised prompt
inside <prompt> and </prompt>.
```

E.5.4 EvoPrompt-DE meta-prompt

DE operation prompt

```
Please follow the instruction step-by-step to generate a better
prompt.
1.  Identify the different parts between Prompt 1 and Prompt 2:
Prompt 1:  Read the mathematical problem carefully, reason step by
step, and derive the exact answer.
Prompt 2:  Identify the relevant quantities and constraints, choose a
suitable method, and verify the result.
2.  Randomly mutate the different parts.
3.  Combine the different parts with Prompt 3, selectively replace it
with the different parts from step 2, and generate a new prompt.
Prompt 3:  Solve the problem systematically, keep symbolic work
exact, and check that the final result satisfies the original
conditions.
4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with <prompt> and </prompt>:
Basic Prompt:  Solve the following math problem.  Think step by step
carefully before answering.
1.  Identifying the different parts between Prompt 1 and Prompt 2:
Prompt 1:  Read the mathematical problem carefully, reason step by
step, and derive the exact answer.
Prompt 2:  Identify the relevant quantities and constraints, choose a
suitable method, and verify the result.
Different parts:
"read the mathematical problem carefully" vs "identify the relevant
quantities and constraints"
```

```
"reason step by step" vs "choose a suitable method"
"derive the exact answer" vs "verify the result"
2.  Randomly mutate the different parts:
"read the mathematical problem carefully" -> "translate every
condition into a precise mathematical statement"
"reason step by step" -> "organize the derivation into connected
steps"
"choose a suitable method" -> "select an efficient mathematical
strategy"
"verify the result" -> "substitute the result back into the original
conditions"
3.  Combine the different parts with Prompt 3, selectively replace it
with the different parts in step 2 and generate a new prompt:
Prompt 3:  Solve the problem systematically, keep symbolic work
exact, and check that the final result satisfies the original
conditions.
New Prompt:  Translate every condition precisely, select an efficient
mathematical strategy, organize the exact derivation into connected
steps, and substitute the result back into the original conditions.
4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with <prompt> and </prompt>:
Basic Prompt:  Solve the following math problem.  Think step by step
carefully before answering.
Final Prompt:  <prompt>Translate the problem conditions precisely,
choose an appropriate mathematical strategy, derive the result step
by step using exact symbolic work, and verify it against the original
conditions before answering.</prompt>
Please follow the instruction step-by-step to generate a better
prompt.
1.  Identify the different parts between Prompt 1 and Prompt 2:
Prompt 1:  #DONOR_PROMPT_1#
Prompt 2:  #DONOR_PROMPT_2#
2.  Randomly mutate the different parts.
3.  Combine the different parts with Prompt 3, selectively replace it
with the different parts in step 2, and generate a new prompt.
Prompt 3:  #DONOR_PROMPT_3#
4.  Crossover the prompt in step 3 with the following basic prompt
and generate a final prompt bracketed with <prompt> and </prompt>:
Basic Prompt:  #TARGET_PROMPT#
1.
```

E.5.5 ETGPO META-PROMPTS

First batch taxonomy creation prompt

```
You are an expert at analyzing why language models fail on
mathematical problem-solving.
#ERROR_EXAMPLES#
## Your Task
Analyze every failure and identify its root cause.  Be as descriptive
as possible.

For each failure:
1.  Find the EARLIEST point in the response's reasoning where it went
wrong.
2.  Explain the mathematical error, missing condition, or invalid
inference.
3.  Explain why that error led to the wrong final answer.
4.  If the reasoning is correct, check whether the final answer is
mathematically equivalent to the ground-truth answer and expressed
```

precisely; otherwise, classify it as a final-answer precision or formatting error.

Create issue categories that capture each type of error. Categories should be general enough to potentially apply to other traces, but specific enough to be meaningful.

Identify reusable reasoning or decision errors rather than question-specific topics. Do not create categories based only on particular entities, answer choices, scientific terms, or isolated facts. Group failures that share the same underlying error even when their question topics differ.

IMPORTANT: Each category must be SELF-CONTAINED and understandable by someone who has NOT seen the original problems.

Assign every failure to exactly one meaningful category. Do not use "None", "Correct Answer", "Uncategorized", "Unknown", or "Other" as a category.

Output Format

Return only one JSON object matching this structure:

```json
{
"categories": [
{
"category_name": "Short descriptive name",
"summary": "One-sentence error pattern",
"description": "Detailed self-contained description",
"example": "A brief topic-independent illustration based on the observed error pattern",
"error_type": "Type of error",
"why_leads_to_wrong_answer": "How this error causes wrong answers"
}
],
"failure_assignments": [
{
"failure_id": 1,
"problem_id": "ID shown in the failure",
"category_name": "Assigned category name",
"trace_details": {
"trace_specific_location": "Where the error occurred",
"trace_specific_details": "What specifically went wrong"
} } ] }
```

Subsequent batch taxonomy update prompt

You are an expert at analyzing why language models fail on mathematical problem-solving.

Existing Issue Categories

Category: #CATEGORY_NAME#
- Summary: #CATEGORY_SUMMARY#
- Description: #CATEGORY_DESCRIPTION#
- Example: #CATEGORY_EXAMPLE#
- Error Type: #ERROR_TYPE#
- Why it leads to wrong answer: #WHY_WRONG#
- Failures assigned so far: 1

New Failures

#NEW_ERROR_EXAMPLES#

Your Task

For every new failure, decide whether its root cause fits an existing category. Reuse that category whenever it fits. Create a new category only when the error is fundamentally different. New

categories must be self-contained, generalizable, specific, and actionable.

Identify reusable reasoning or decision errors rather than question-specific topics. Do not create categories based only on particular entities, answer choices, scientific terms, or isolated facts. Group failures that share the same underlying error even when their question topics differ.

If the reasoning is correct, check whether the final answer is mathematically equivalent to the ground-truth answer and expressed precisely; otherwise, classify it as a final-answer precision or formatting error.

Assign every failure to exactly one meaningful category. Reuse the closest existing category when it captures the underlying error; otherwise, create a generalizable new category. Do not use "None", "Correct Answer", "Uncategorized", "Unknown", or "Other" as a category.

Output Format

Return only one JSON object matching this structure:

```json
{
"new_categories": [
{
"category_name": "Short descriptive name for a new error",
"summary": "One-sentence error pattern",
"description": "Detailed self-contained description",
"example": "A brief topic-independent illustration based on the observed error pattern",
"error_type": "Type of error",
"why_leads_to_wrong_answer": "How this error causes wrong answers"
}
],
"failure_assignments": [
{
"failure_id": 1,
"problem_id": "ID shown in the failure",
"is_new_category": false,
"category_name": "Existing or new category name",
"trace_details": {
"trace_specific_location": "Earliest reasoning location",
"trace_specific_details": "What specifically went wrong"
} } ] }
```

The "new_categories" list must contain only categories that do not already exist.

Guidance generation prompt

You are an expert at improving language model performance on mathematical problem-solving.

I have identified the following error categories from model failures. Generate guidance to help avoid these errors.

Category 1: #CATEGORY_NAME#

Statistics: 1 failures (100.0%), 1 problems

Summary: #CATEGORY_SUMMARY#

Description: #CATEGORY_DESCRIPTION#

Example: #CATEGORY_EXAMPLE#

Error Type: #ERROR_TYPE#

Why it leads to wrong answer: #WHY_WRONG#

````
---

## Your Task

Generate guidance text that:
1. Addresses each failure category with specific, actionable advice
2. Is written as instructions TO the model
3. Expresses advice as a reusable reasoning or decision strategy
4. Is prioritized by frequency

Generate SHORT, CONCISE guidance. Each item should be 1-2 sentences.

## Critical Constraints

- The goal is ACCURACY, not caution. Never generate guidance that
encourages the model to refuse, abstain, or say "not specified" when
an answer can be reasonably provided.
- Keep the guidance task-general. Do not copy question-specific
entities, passage details, answer choices, scientific terms, or
isolated facts from the categories.
- Do not invent new WRONG/CORRECT question-answer examples.
- Preserve this task behavior: A mathematical problem may require
numerical, algebraic, geometric, combinatorial, or symbolic
reasoning.
The task requires deriving the solution carefully and then outputting
an exact final answer.
- General guidance may ask for a concise and precise final answer,
but do not add answer tags, input placeholders, or a model-response
template; these are handled separately from the prompt being
optimized.
- Do not mention the taxonomy, error analysis, ground-truth answers,
or evaluation metrics in the completed prompt.

## Output Format

Return a JSON object with:
```json
{
"guidance_items": [
{
"category_name": "Name of the category",
"guidance_text": "The full guidance text for this category"
}
],
"preamble": "1-2 sentence introduction",
"full_prompt": "Complete enhanced prompt starting with base
instruction"
}
```

The "full_prompt" should start with:
"#CURRENT_INSTRUCTION_PROMPT#"

Then add the preamble and guidance items.
````

GreaTer. GreaTer uses no optimizer-model meta-prompt. It proposes fixed-length single-token replacements directly from target-model token gradients and ranks those candidates with the task-loss and fluency objective.

E.6 Optimized prompts by Qwen3-4B on FS-TACRED

E.6.1 RPO-optimized prompt at iteration 5 and its second-stage refinements

RPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and Object entities, which
are indicated with the subject and object tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
1.  **Verify the subject and object types**:
- For relations starting with "org:", the **subject must be an
organization** (not a person or role).
- The **object must align with the relation's definition** (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly links the Subject and Object as per the relation's
description.  For example, "org:subsidiaries" requires a hierarchical
connection between an organization and its subsidiary (e.g., "Company
A owns Company B"), not a personal or abstract relationship.
3.  **Recognize object types**:
- The object can be a **specific entity** (e.g., "US-based
Ryder Public Transportation Services") or a **category** (e.g.,
"Christian", "Islamist") if the relation explicitly allows it.
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's description explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "yes"; otherwise, answer "no".
```

LPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and Object entities, which
are indicated with the subject and object tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
1.  **Verify the subject and object types**:
- For relations starting with "org:", the **subject must be an
organization** (not a person or role).
- For per:  relations, the subject must be a person.
- The **object must align with the relation's definition** (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly states the link between the Subject and Object as per
the relation's description, using direct assertions (e.g., "is
from," "was born in," "works for") rather than indirect or implied
connections.
3.  **Recognize object types**:
```

```
- The object can be a **specific entity** (e.g., "US-based Ryder
Public Transportation Services") or a **explicitly permitted
category (e.g., 'Christian,' 'Islamist,' 'Lakota') if the relation
explicitly allows it, but avoid treating nationalities, tribes,
or administrative regions as valid objects unless explicitly
permitted.**
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's description explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "yes"; otherwise, answer "no".
```

GreaTer-optimized prompt

```
I are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and Object entities, which
are indicated with the subject and object tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
1.  **Verify the subject and object types**:
- For relations starting with "org:", the **subject must be an
organization** (not a person or role).
- The **object must align with the relation's definition** (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly links the Subject and Object as per the relation's
description.  For example, "org:subsidiaries" requires a hierarchical
connection between an organization and its subsidiary (e.g., "Company
A owns Company B"), not a personal or abstract relationship.
3.  **Recognize object types**:
- The object can be a **specific entity** (e.g., "US-based
Ryder Public Transportation Services") or a **category** (e.g.,
"Christian", "Islamist") if the relation explicitly allows it.
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's description explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "yes"; otherwise, answer "no".
```

GreaTer-TG-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and Object entities, which
are indicated with the subject and object tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
```

```
1.  **Verify the subject and object types**:
- For relations starting with "org:", the **subject must be an
organization** (not a person or role).
- The **object must align with the relation's definition** (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly links the Subject and Object as per the relation's
description.  For example, "org:subsidiaries" requires a hierarchical
connection between an organization and its subsidiary (e.g., "Company
A owns Company B"), not a personal or abstract relationship.
3.  **Recognize object types**:
- The object can be a **specific entity** (e.g., "US-based
Ryder Public Transportation Services") or a **category** (e.g.,
"Christian", "Islamist") if the relation explicitly allows it.
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's description explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "your"; otherwise, answer "no".
```

GradPO-optimized prompt

```
You are given a relation name, a definition of the relation in
bracketed text, a support example sentence exemplifying the relation,
and a query inquiry sentence.  A relation connects the Subject and
Object entities, which are indicated with the subject and object
tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
1.  **Verify the subject and object types**:
- For relations starting with "org:", the **subject must be an
organization** (not a person or role).
- The **object must align with the relation's definition** (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly links the Subject and Object as per the relation's
description.  For example, "org:subsidiaries" requires a hierarchical
connection between an organization and its subsidiary (e.g., "Company
A owns Company B"), not a personal or abstract relationship.
3.  **Recognize object types**:
- The object can be a **specific entity** (e.g., "US-based
Ryder Public Transportation Services") or a **category** (e.g.,
"Christian", "Islamist") if the relation explicitly allows it.
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's definition explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "yes"; otherwise, answer "no".
```

GradPO-Prob-optimized prompt

```
You are given a relation name, a description of the relation in
[brackets], a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and Object entities, which
are indicated with the subject and object tags, respectively.

To determine if the relation holds between the Subject and Object
entities in the query sentence, strictly follow these steps:
1.  **Verify the subject and object types**:
- For relations starting with "org:", the subject must be an
organization (not a person or role).
- The object must align with the relation's definition (e.g.,
"org:subsidiaries" requires a subsidiary entity, not a role or
category; "org:political/religious_affiliation" accepts a religious
or political category, not a person).
2.  **Check the relationship**:  Confirm that the query sentence
explicitly links the Subject and Object as per the relation's
description.  For example, "org:subsidiaries" requires a hierarchical
connection between an organization and its subsidiary (e.g., "Company
A owns Company B"), not a personal or abstract relationship.
3.  **Recognize object types**:
- The object can be a specific entity (e.g., "US-based Ryder
Public Transportation Services") or a category (e.g., "Christian",
"Islamist") if the relation explicitly allows it.
- Avoid treating abstract terms (e.g., "Christian") as roles or
people unless the relation's description explicitly permits it.
4.  **Focus on explicit connections**:  Only answer "yes" if the
query sentence directly establishes the relation between the subject
and object as defined.  For example, "org:number_of_employees/members"
requires a numerical value (e.g., "42,000") to indicate the count,
not a vague term like "many."

If the relation holds, answer "yes"; otherwise, answer "no".
```

E.6.2 EvoPrompt-DE-optimized prompt at iteration 5 and its second-stage refinements

EvoPrompt-DE-optimized prompt

```
You are given a relation name, its description, a support sentence
that exemplifies the relation, and a query sentence.  Analyze
the relation description and the query sentence.  The Subject and
Object are labeled in the query sentence.  Determine if the relation
is expressed between the marked entities.  Check the relation
definition, the semantic type of each entity, the direction from
Subject to Object, and whether the evidence is explicit or strongly
implied.  Treat the support sentence as a guide to the relation,
not evidence for the query.  Confirm that the relation direction is
Subject-to-Object, that the Object has the required type or value,
and that the query provides sufficient explicit or contextually
implied evidence.  Ignore any irrelevant entities or facts not
directly stated in the query.  If the relation holds in the query
sentence, answer "yes"; otherwise answer "no".
```

GreaTer-optimized prompt

```
I are given a relation name, its description, a support sentence
that exemplifies the relation, and a query sentence.  Analyze
the relation description and the query sentence.  The Subject and
```

```
Object are labeled in the query sentence.  Determine if the relation
is expressed between the marked entities.  Check the relation
definition, the semantic type of each entity, the direction from
Subject to Object, and whether the evidence is explicit or strongly
implied.  Treat the support sentence as a guide to the relation,
not evidence for the query.  Confirm that the relation direction is
Subject-to-Object, that the Object has the required type or value,
and that the query provides sufficient explicit or contextually
implied evidence.  Ignore any irrelevant entities or facts not
directly stated in the query.  If the relation holds in the query
sentence, answer "yes"; otherwise answer "no".
```

GreaTer-TG-optimized prompt

```
You are given a relation name, its description, a support sentence
that exemplifies the relation, and a query sentence.  Analyze
the relation description and the query sentence.  The Subject and
Object are labeled in the query sentence.  Determine if the relation
is expressed between the marked entities.  Check the relation
definition, the semantic type of each entity, the direction from
Subject to Object, and whether the evidence is explicit or strongly
implied.  Treat the support sentence as a guide to the relation,
not evidence for the query.  Confirm that the relation direction is
Subject-to-Object, that the Object has the required type or value,
and that the query provides sufficient explicit or contextually
implied evidence.  Ignore any irrelevant entities or facts not
directly stated in the query.  If the relation holds in the query
sentence, answer "your"; otherwise answer "no".
```

GradPO-optimized prompt

```
You are told given a relation name, its description, a support text
that demonstrates the relation, and a query sentence.  Analyze
the relation description and the query sentence.  The Subject and
Object are labeled in the query sentence.  Evaluate if the relation
is expressed between the marked entities.  Check the relation
definition, the semantic type of each entity, the direction from
Subject to Object, and whether the evidence is explicit or strongly
implied.  Treat the support sentence as a guide to the relation,
not evidence for the query.  Confirm that the relation direction is
Subject-to-Object, that the Object has the required type or value,
and that the query provides sufficient explicit or contextually
implied evidence.  Ignore any irrelevant entities or facts not
directly stated in the query.  If the relation holds in the query
sentence, answer "yes"; otherwise answer "no".
```

GradPO-Prob-optimized prompt

```
The provided relation name, its description, a support example that
exemplifies the relation, and a query sentence.  Analyze the relation
description and the query sentence.  The Subject and Object are
labeled in the query sentence.  Check if the relation is expressed
between the marked entities.  Check the relation definition, the
semantic type of each entity, the direction from Subject to Object,
and whether the evidence is explicit or strongly implied.  Treat the
support sentence as a guide to the relation, not evidence for the
query.  Confirm that the relation direction is Subject-to-Object,
that the Object has the required type or value, and that the query
provides sufficient explicit or contextually implied evidence.
```

```
Ignore any irrelevant entities or facts not directly stated in the
query.  If the relation holds in the query sentence, answer "yes";
otherwise answer "no".
```

E.6.3 ETGPO-OPTIMIZED PROMPT AND ITS SECOND-STAGE REFINEMENTS

ETGPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and the Object entities
indicated by subject/object tags.  You must determine if the relation
holds between the tagged entities in the query sentence, following
these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship type
specified (e.g., 'familial' vs.  'friendship').  Do not conflate
non-familial roles with familial relations.
4.  **Role Implication Awareness**:  Recognize that certain roles
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.

Answer 'yes' if the relation holds under these constraints;
otherwise, answer 'no'.
```

LPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and the Object entities
indicated by subject/object tags.  You must determine if the relation
holds between the tagged entities in the query sentence, following
these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country, URLs).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship
type specified (e.g., 'familial' vs.  'friendship').  Do not
conflate non-familial roles with familial relations.  Organizational
components such as congresses may imply membership if explicitly
named.
4.  **Role Implication Awareness**:  Recognize that certain roles
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.  Include
```

```
organizational roles (e.g., 'member', 'committee') that directly
indicate membership.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.  Subnational entities such as states or provinces
are not sovereign.

Answer 'yes' if the relation holds under these constraints;
otherwise, answer 'no'.
```

GreaTer-optimized prompt

```
Given are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and the Object entities
indicated by subject/object tags.  You must determine if the relation
holds between the tagged entities in the query sentence, following
these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship type
specified (e.g., 'familial' vs.  'friendship').  Do not conflate
non-familial roles with familial relations.
4.  **Role Implication Awareness**:  Recognize that certain roles
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.

Answer 'yes' if the relation holds under these constraints;
otherwise, answer 'no'.
```

GreaTer-TG-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.  A relation connects the Subject and the Object entities
indicated by subject/object tags.  You must determine if the relation
holds between the tagged entities in the query sentence, following
these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship type
specified (e.g., 'familial' vs.  'friendship').  Do not conflate
non-familial roles with familial relations.
```

```
4.  **Role Implication Awareness**:  Recognize that certain roles
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.

Answer 'none' if the relation holds under these constraints;
otherwise, answer 'no'.
```

GradPO-optimized prompt

```
You are supplied a relation name, a description of the relation in
curly brackets, a support sentence exemplifying the relation, and
a query sentence.  A relation connects the Subject and the Object
entities indicated by subject/object tags.  You must determine if
the relation holds between the tagged entities in the query sentence,
following these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship type
specified (e.g., 'familial' vs.  'friendship').  Do not conflate
non-familial roles with familial relations.
4.  **Role Implication Awareness**:  Recognize that certain roles
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.

Answer 'yes' if the relation holds under these constraints;
otherwise, answer 'no'.
```

GradPO-Prob-optimized prompt

```
Answer the question by determining if the relation holds under these
constraints.  A relation connects the Subject and the Object entities
indicated by subject/object tags.  You must determine if the relation
holds between the tagged entities in the query sentence, following
these rules:

1.  **Entity Resolution**:  Treat abbreviations (e.g., 'Minn.')
as their full forms (e.g., 'Minnesota'), and ensure the object
matches the required entity type (e.g., state/province vs.  city vs.
country).
2.  **Explicit Matching**:  Only use information explicitly stated in
the query.  Do not infer events (e.g., death) or relationships (e.g.,
familial ties) unless directly mentioned.
3.  **Relation Type Precision**:  Match the exact relationship type
specified (e.g., 'familial' vs.  'friendship').  Do not conflate
non-familial roles with familial relations.
4.  **Role Implication Awareness**:  Recognize that certain roles
```

```
(e.g., 'managing principal') inherently imply specific relationships
(e.g., 'shareholder') even if not explicitly stated.
5.  **Geographic Hierarchy**:  Distinguish between sovereign
countries (e.g., 'Cuba') and subnational entities (e.g.,
'California').  A country cannot satisfy a relation requiring a
state/province.
```

E.7 Optimized prompts by Gemma3-4B on FS-TACRED

E.7.1 RPO-optimized prompt at iteration 5 and its second-stage refinements

RPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, and a query sentence.

Your task is to determine whether the *specific relation* described
holds between the Subject and the Object entities in the *query
sentence*.  The Subject and the Object entities are indicated with
the subject and object tags, respectively.  Carefully examine the
query sentence, paying close attention to keywords and phrases
commonly associated with the specified relation.

Consider the following:

* **Explicit Statements:** Look for direct statements that express
the relation (e.g., "is a subsidiary of," "was born in").
* **Strong Indicators:** Recognize that relations can be implied or
indicated through related language.  For example, "native of" often
implies "city of birth," and "subsidiary of" directly indicates an
organizational relationship.
* **Contextual Clues:** Evaluate the surrounding context to confirm
the relationship.

Do not answer "yes" unless the query sentence explicitly or strongly
implies the specified relation.  Base your decision *solely* on the
content of the query sentence.  Ignore any support examples provided.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

LPO-optimized prompt

```
You are given a relation name, a description of the relation
in brackets, and a query sentence.  Completely disregard any
support sentences.  Your decision must be based *exclusively*
on the information presented within the query sentence and the
subject/object tags.

Your task is to determine whether the *specific relation* described
holds between the Subject and the Object entities in the *query
sentence*.  The Subject and the Object entities are indicated with
the subject and object tags, respectively.  Carefully examine the
query sentence, paying close attention to keywords and phrases
commonly associated with the specified relation.

Consider the following:

* **Explicit Statements:** Look for direct statements that express
the relation (e.g., "is a subsidiary of," "was born in").
```

```
* **Strong Indicators:** Recognize that relations can be implied or
indicated through related language.  For example, "native of" often
implies "city of birth," and "subsidiary of" directly indicates an
organizational relationship.
* **Contextual Clues:** Evaluate the surrounding context to confirm
the relationship.

Do not answer "yes" unless the query sentence explicitly or strongly
implies the specified relation.  Base your decision *solely* on the
content of the query sentence.  Do not consider any support examples
provided.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

GreaTer-optimized prompt

```
Class are given a relation name, a description of the relation in
brackets, and a query sentence.

Your task is to determine whether the *specific relation* described
holds between the Subject and the Object entities in the *query
sentence*.  The Subject and the Object entities are indicated with
the subject and object tags, respectively.  Carefully examine the
query sentence, paying close attention to keywords and phrases
commonly associated with the specified relation.

Consider the following:

* **Explicit Statements:** Look for direct statements that express
the relation (e.g., "is a subsidiary of," "was born in").
* **Strong Indicators:** Recognize that relations can be implied or
indicated through related language.  For example, "native of" often
implies "city of birth," and "subsidiary of" directly indicates an
organizational relationship.
* **Contextual Clues:** Evaluate the surrounding context to confirm
the relationship.

Do not answer "yes" unless the query sentence explicitly or strongly
implies the specified relation.  Base your decision *solely* on the
content of the query sentence.  Ignore any support examples provided.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

GradPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, and a query sentence.

Your task is to determine whether the *specific relation* described
holds between the Subject and the Object entities in the *query
sentence*.  The Subject and the Object entities are indicated with
the subject and object tags, respectively.  Carefully examine the
query sentence, paying close attention to keywords and phrases
commonly associated with the specified relation.

Consider the following:

* **Explicit Statements:** Look for direct statements that express
```

```
the relation (e.g., "is a subsidiary for," "was born in").
* **Strong Indicators:** Recognize that relations can be implied or
indicated through related language.  For example, "native of" often
implies "city of birth," and "subsidiary for" directly indicates an
organizational relationship.
* **Contextual Clues:** Evaluate the surrounding context to confirm
the relationship.

Do not answer "yes" unless the query sentence explicitly or strongly
implies the specified relation.  Base your decision solely on the
content of the query sentence.  Ignore any support examples provided.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

E.7.2 EvoPrompt-DE-optimized prompt at iteration 5 and its second-stage refinements

EvoPrompt-DE-optimized prompt

```
You will be provided with a relation name, a detailed description
of the relation, a support sentence to illustrate the relation's
core characteristics, and a query sentence.  Examine the relation
description, then carefully analyze the query sentence.  The
Subject and Object are clearly identified.  First, identify the
tagged Subject and Object.  Then, verify the consistency between
the relation's definition, the entities' semantic roles, the
intended direction, and the presence of supporting evidence,
whether directly stated or logically inferable.  Pay attention
to potential ambiguities in the query sentence.  Evaluate solely
on the information presented within the query sentence; external
information is irrelevant.  Treat the support sentence as a purely
illustrative example; do not incorporate any information from it
into your judgment.  Respond with a clear 'yes' or 'no' to indicate
whether the relation is confirmed, considering only the information
presented within the query sentence.
```

LPO-optimized prompt

```
You will be provided with a relation name, a detailed description
of the relation, a support sentence to illustrate the relation's
core characteristics, and a query sentence.  Carefully examine the
relation description.  then carefully analyze the query sentence.
The Subject and Object are clearly identified.  First, identify the
tagged Subject and Object.  Then, verify the consistency between
the relation's definition, the entities' semantic roles, the
intended direction, and the presence of supporting evidence, whether
directly stated or logically inferable.  Pay attention to potential
ambiguities in the query sentence.  Evaluate based solely on the
information present within the query sentence; external information,
including the provided support sentence, is irrelevant.  Treat the
support sentence as an example for context only; completely disregard
it when making your judgment.  Respond with a clear 'yes' or 'no'
to indicate whether the relation is confirmed, considering only the
query sentence.
```

GreaTer-optimized prompt

```
You will be provided with a relation name, a detailed description
of the relation, a support sentence to illustrate the relation's
core characteristics, and a query sentence.  Examine the relation
description, then carefully analyze the query sentence.  The
Subject and Object are clearly identified.  First, identify the
tagged Subject and Object.  Then, verify the consistency between
the relation's definition, the entities' semantic roles from the
intended direction, and the presence of supporting evidence,
whether directly stated or logically inferable.  Pay attention
to potential ambiguities in the query sentence.  Evaluate solely
on the information presented within the query sentence; external
information is irrelevant.  Treat the support sentence as a purely
illustrative example; do not incorporate any information from it
into your judgment.  Respond with a clear 'yes' or 'no' to indicate
whether the relation is confirmed, considering only the information
presented within the query sentence.
```

GradPO-Prob-optimized prompt

```
You will be provided with a relation name, a detailed description
of the relation, a support sentence to illustrate the relation's
core characteristics, and a query sentence.  Examine the relation
description, then carefully analyze the query sentence.  The
Subject and Object are clearly identified.  First, identify the
tagged Subject and Object.  Then, verify the consistency between
the relation's definition, the entities' semantic roles, the
intended direction, and the presence of supporting evidence,
whether directly stated or logically inferable.  Pay attention to
potential ambiguities in the query sentence.  Evaluate solely on
the informational presented within the query sentence; external
information is irrelevant.  Treat the support sentence as a purely
illustrative example; do not incorporate any information from it
into your judgment.  Respond with a clear 'yes' or 'no' to indicate
whether the relation is confirmed, considering only the informational
presented within the query sentence.
```

E.7.3 ETGPO-OPTIMIZED PROMPT AND ITS SECOND-STAGE REFINEMENTS

ETGPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.

A relation connects the Subject and the Object entities.  The Subject
and the Object entities are indicated with the subject and object
tags, respectively.

Your task is to determine whether the relation holds between the
Subject and the Object entities in the query sentence.  Focus on the
*meaning* of the relationship described, not just the presence of
shared words.  Consider the broader context and what the relationship
*represents*, not just whether the entities appear together.

When evaluating, avoid fixating on specific details from the support
sentence.  Generalize the relational pattern to apply to different
entities and contexts.  The support sentence is an example; the query
sentence may use different entities or phrasing to express the same
```

```
relationship.

Be mindful of nuances in meaning.  The relation type describes a
specific connection; do not infer the relationship unless it is
clearly and directly present in the query sentence, even if related
concepts are mentioned.  Focus on the *core* of the relation.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

LPO-optimized prompt

```
You are given a relation name, a description of the relation in
brackets, a support sentence exemplifying the relation, and a query
sentence.

A relation connects the Subject and the Object entities.  The Subject
and the Object entities are indicated with the subject and object
tags, respectively.

Your task is to determine if the query sentence *explicitly and
directly* expresses the relationship described by the relation name
and exemplified in the support sentence, between the Subject and the
Object entities.  Prioritize a clear, demonstrable connection over
subtle or implied relationships.  Focus on the *essential* meaning
of the relationship, rather than just shared words.  Consider the
broader context, but primarily assess whether the query sentence
*states* the relationship.

Do not fixating on specific details from the support sentence.
Generalize the relational pattern to apply to different entities and
contexts.  The support sentence is an example; the query sentence may
use different entities or phrasing to express the same relationship.

Be mindful of nuances in meaning.  The relation type describes a
specific connection; do not infer the relationship unless it is
clearly and directly present in the query sentence, even if related
concepts are mentioned.  Focus on the *core* of the relation.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

GradPO-optimized prompt

```
You are given a relation category, a description of the relation in
brackets, a demonstration of the relation, and a query sentence.

A relation connects the Subject and the Object entities.  The Subject
and the Object entities are indicated with the subject and object
tags, respectively.

Your task is to determine whether the relation holds between the
Subject and the Object entities in the query sentence.  Focus on the
*meaning* of the relationship described, not just the presence of
shared words.  Consider the broader context and what the relationship
*represents*, not just whether the entities appear together.

When evaluating, avoid fixating on specific details from the
demonstration of sentence.  Generalize the relational pattern to
apply to different entities and contexts.  The demonstration of
```

```
sentence is an example; the query sentence may use different entities
or phrasing to express the same relationship.

Be mindful of nuances in meaning.  The relation category describes
a specific connection; do not infer the relationship unless it is
clearly and directly present in the query sentence, even if related
concepts are mentioned.  Focus on the *core* of the relation.

If the relation holds between the Subject and the Object entities
in the query sentence, answer "yes"; otherwise, answer "no".
```

E.8 Optimized prompts by Qwen3-4B on MATH-500

E.8.1 RPO at iteration 10 and its second-stage refinements

First-stage prompt: RPO at iteration 10

```
Solve the following math problem.  Think step by step, but be concise
and focused.  Once you derive the solution, state the final answer
clearly and succinctly without unnecessary repetition or
verification.
```

GreaTer-refined prompt

```
Okayolve the following math problem.  Think step by step, but be
concise and focused.  Once you derive the solution, state the final
answer clearly and succinctly without unnecessary repetition or
verification.
```

GreaTer-TG-refined prompt

```
Solve the following mathematical problem.  Think step by step, but be
concise and focused.  Once you derive the solution, state the final
answer clearly and succinctly without unnecessary repetition or
verification.
```

LPO-refined prompt

```
Solve the following math problem.  Think step by step, but be concise
and focused.  Once you derive the solution, state the final answer
clearly and succinctly without unnecessary repetition or
verification.  Make sure to box your final answer using \boxed{}.
```

GradPO-refined prompt

```
Address the following math problem.  analyze step by step, but be
concise and focused.  Once you derive the solution, state the final
answer clearly and succinctly without unnecessary repetition or
verification.
```

E.8.2 EvoPrompt-DE at iteration 10 and its second-stage refinements

First-stage prompt: EvoPrompt-DE at iteration 10

```
Identify the type of mathematical reasoning required (e.g., algebra,
geometry, probability, etc.), derive the solution logically from the
provided information, write out each reasoning step in detail, apply
only the most effective and relevant mathematical tools, provide the
```

```
final answer in the specified format, maintain clarity and brevity in
the explanation, and clearly box or highlight the final answer.
```

GreaTer-TG-refined prompt

```
Identify the type of mathematical reasoning required (e.g., algebra,
geometry, probability, etc.), derive the solution logically from the
provided information, write out each reasoning step in detail, apply
only the most effective and relevant mathematical tools, provide the
final answer in the specified format, maintain clarity and brevity in
the explanation, and clearly indicate or highlight the final answer.
```

GradPO-refined prompt

```
Identify the type of mathematical reasoning required (e.g., algebra,
geometry, probability, etc.), derive the solution logically from the
provided information, write out each reasoning step in detail, apply
only the most effective and relevant mathematical tools, provide the
final answer in the prescribed format, maintain clarity and brevity
in the explanation, and clearly present the final answer.
```

E.8.3 ETGPO AND ITS SECOND-STAGE REFINEMENTS

First-stage prompt: ETGPO

```
Solve the following math problem.  Think step by step carefully
before answering.  Prioritize precise reasoning and strict adherence
to formatting requirements to ensure correctness and completeness.
Verify mathematical principles at each step; cross-check formula
applicability and logical flow before proceeding to the final answer.
Explicitly confirm the required answer format (e.g., numerical value,
list, expression) and ensure full compliance before final submission.
```

GreaTer-TG-refined prompt

```
Solve the following math puzzle.  Think step by step carefully before
answering.  Prioritize precise reasoning and strict adherence to
formatting requirements to ensure correctness and completeness.
Verify mathematical principles at each step; cross-check formula
applicability and logical flow before proceeding to the final answer.
Explicitly confirm the required answer format (e.g., numerical value,
list, expression) and ensure full compliance before final submission.
```

GradPO-refined prompt

```
Solve the following mathematical question.  Break down the problem
step by step carefully before submitting the answer.  Prioritize
precise reasoning and strict adherence to formatting requirements to
ensure correctness and completeness.  Verify mathematical principles
at each step; cross-check formula applicability and logical flow
before proceeding to the final answer.  Explicitly confirm the
required answer format (e.g., numerical value, list, expression) and
ensure full compliance before final submission.
```

GradPO-Prob-refined prompt

```
Solve the following math problem.  Work through the problem and
carefully before answering.  Prioritize precise reasoning and strict
adherence to formatting requirements to ensure correctness and
```

```
completeness.  Verify mathematical principles at each step;
cross-check formula applicability and logical flow before proceeding
to the final answer.  Explicitly confirm the required answer format
(e.g., numerical value, list, expression) and ensure full compliance
before final submission.
```

E.9 Optimized prompts by Gemma3-4B on MATH-500

E.9.1 RPO at iteration 10 and its second-stage refinements

First-stage prompt: RPO at iteration 10

```
Solve the following math problem.  Break down the solution into
small, manageable steps.  Explicitly state any formulas or principles
you use at each step.  When dealing with functions, clearly indicate
the domain or range being considered.  For geometric problems,
include diagrams or sketches to support your reasoning.  When solving
for rotations, be mindful of positive and negative angles and
equivalent rotations.  **At each step, carefully double-check your
calculations and algebraic manipulations to ensure accuracy, paying
close attention to signs, order of operations, and variable
substitutions.  Explicitly state the result of each calculation
before proceeding to the next step.  Focus on identifying underlying
mathematical structures and general patterns; avoid getting bogged
down in specific numerical details unless they are critical for
establishing those patterns.  If the problem involves multiple cases
or possibilities, analyze each case systematically and explain why
certain cases are valid or invalid.  When comparing equations or
expressions, ensure you account for all possible transformations and
relationships.  Before finalizing your answer, carefully review all
steps and ensure your solution is consistent with the problem
statement.**
```

LPO-refined prompt

```
Solve the following math problem.  Break down the solution into small
steps, explaining your reasoning clearly at each stage.  Explicitly
state any formulas or principles you use.  When dealing with
functions, clearly indicate the domain or range being considered.
For geometric problems, include diagrams or sketches to support your
reasoning.  When solving for rotations, be mindful of positive and
negative angles and equivalent rotations.  At each step, carefully
double-check your calculations and algebraic manipulations to ensure
accuracy, paying close attention to signs, order of operations, and
variable substitutions.  Explicitly state the result of each
calculation before proceeding to the next step.  Focus on identifying
underlying mathematical structures and general patterns; avoid
getting bogged down in specific numerical details unless they are
critical for establishing those patterns.  If the problem involves
multiple cases or possibilities, analyze each case systematically and
explain why certain cases are valid or invalid.  When comparing
equations or expressions, ensure you account for all possible
transformations and relationships.  Before finalizing your answer,
carefully review all steps and ensure your solution is consistent
with the problem statement.  Show all working and include the final
answer.  **At each step, carefully double-check your calculations and
algebraic manipulations to ensure accuracy, paying close attention to
signs, order of operations, and variable substitutions.  Explicitly
state the result of each calculation before proceeding to the next
step.  Focus on identifying underlying mathematical structures and
general patterns; avoid getting bogged down in specific numerical
details unless they are critical for establishing those patterns.  If
```

```
the problem involves multiple cases or possibilities, analyze each
case systematically and explain why certain cases are valid or
invalid.  When comparing equations or expressions, ensure you account
for all possible transformations and relationships.  Before
finalizing your answer, carefully review all steps and ensure your
solution is consistent with the problem statement.**
```

GradPO-refined prompt

```
Solve the following math problem.  Break down the solution into
small, manageable steps.  Explicitly state any formulas or principles
you use in each step.  When dealing with rotational transformations,
be mindful of positive and negative angles and equivalent rotations.
Illustrate your reasoning with diagrams or sketches to support your
reasoning.  When solving for rotations, be mindful of positive and
negative angles and equivalent rotations.  At each step, carefully
double-check your calculations and algebraic manipulations to ensure
accuracy, paying close attention to signs, order of operations, and
variable substitutions.  Explicitly state the result of each
calculation before giving the final answer.  Focus on identifying
underlying mathematical structures and general patterns; avoid
getting bogged down in specific numerical details unless they are
critical for establishing those patterns.  If the problem involves
multiple cases or possibilities, analyze each case systematically and
explain why certain cases are valid or invalid.  When comparing
equations or expressions, ensure you account for all possible
transformations and relationships.  Before giving the final answer,
carefully review all steps and ensure your solution is consistent
with the problem statement.
```

GradPO-Prob-refined prompt

```
Solve the problem below, break down the solution into small,
manageable steps.  Explicitly state any formulas or principles you
use at each step.  When dealing with functions, clearly indicate the
domain or range being considered.  For geometric problems, include
diagrams or sketches to support your reasoning.  When solving for
rotations, be mindful of positive and negative angles and equivalent
rotations.  Before each step, carefully double-check your
calculations and algebraic manipulations to ensure accuracy, paying
close attention to signs, order of operations, and variable
substitutions.  Explicitly state the result of each calculation
before proceeding to the next step.  Focus on identifying underlying
mathematical structures and general patterns; avoid getting bogged
down in specific numerical details unless they are critical for
establishing those patterns.  If the problem involves multiple cases
or possibilities, analyze each case systematically and explain why
certain cases are valid or invalid.  When comparing equations or
expressions, ensure you account for all possible transformations and
relationships.  Before finalizing your answer, carefully review all
steps and ensure your solution is consistent with the problem
statement.
```

E.9.2 EvoPrompt-DE at iteration 10 and its second-stage refinements

First-stage prompt: EvoPrompt-DE at iteration 10

```
Solve the following math problem.  Prioritize a direct approach and
efficient calculations.  First, identify the type of mathematical
reasoning required.  Break down the problem into smaller, manageable
components.  Establish a clear plan before starting calculations.
```

```
Then, derive the answer carefully, clearly articulating each step
with logical justification, maintaining clarity and accuracy
throughout.  Avoid unnecessary detours or overly complicated methods.
Focus on a streamlined and understandable process and state the final
result with appropriate units, providing a well-structured solution.
```

GreaTer-refined prompt

```
Calculate the following math problem.  Prioritize a direct approach
and efficient calculations.  First, identify the type of mathematical
reasoning required.  Break down the problem into smaller, manageable
components.  Establish a clear plan before starting calculations.
Then, derive the answer carefully, clearly articulating each step
with logical justification, maintaining clarity and accuracy
throughout.  Avoid unnecessary detours or overly complicated methods.
Focus on a streamlined and understandable process and state the final
result with appropriate units, providing a well-structured solution.
```

GreaTer-TG-refined prompt

```
Solve the following question problem.  Prioritize a direct approach
and efficient calculations.  First, identify the type of mathematical
reasoning required.  Break down the problem into smaller, manageable
components.  Establish a clear plan before starting calculations.
Then, derive the answer carefully, clearly articulating each step
with logical justification, maintaining clarity and accuracy
throughout.  Avoid unnecessary detours or overly complicated methods.
Focus on a streamlined and understandable process and state the final
result with appropriate units, providing a well-structured solution.
```

GradPO-refined prompt

```
Solve the following math problem.  Prioritize a direct approach and
efficient calculations.  First, identify the level of mathematical
reasoning required.  Break down the problem into smaller, manageable
components.  Establish a clear plan before starting calculations.
Then, derive the solution carefully, clearly articulating each step
with logical justification, maintaining clarity and accuracy
throughout.  Avoid unnecessary detours or overly complicated methods.
Focus on a streamlined and understandable process and state the final
result with appropriate units of measurement, providing a
well-structured solution.
```

GradPO-Prob-refined prompt

```
Solve the following math problem.  Prioritize a direct approach and
efficient calculations.  First, identify the type of mathematical
reasoning required.  Break down the problem into smaller, manageable
components.  Establish a clear plan before starting calculations.
Then, carefully derive the answer carefully, clearly articulating
each step with logical justification, maintaining clarity and
accuracy throughout.  Avoid unnecessary detours or overly complicated
methods.  Focus on a streamlined and understandable process and state
the final result with appropriate units, providing a well-structured
solution.
```

E.9.3 ETGPO AND ITS SECOND-STAGE REFINEMENTS

First-stage prompt: ETGPO

```
Solve the following math problem.  Think step by step carefully
before answering.
To improve problem-solving accuracy, apply the following strategies
throughout your reasoning process and final answer generation.
Maintain a focus on precise calculations and accurate representation
of problem constraints.
After completing your reasoning, meticulously review your final
calculation and ensure the answer is presented in the correct format
and units as requested by the problem.
Carefully examine the mathematical relationships in combinatorial
problems; avoid assuming constant ratios between counts without
rigorous justification, as combinatorial structures can change
dramatically with scale.
When encountering problems involving alternating actions, ensure you
always begin with the first unfulfilled condition (e.g., the first
closed locker) to accurately simulate the process.
```

GreaTer-TG-refined prompt

```
Solve the following math problem.  Think step by step carefully
before answering.
To improve problem-solving accuracy, apply the following strategies
throughout your reasoning process and final answer calculation.
Maintain a focus on precise calculations and accurate representation
of problem constraints.
After completing your reasoning, meticulously review your final
calculation and ensure the answer is presented in the correct format
and units as requested by the problem.
Carefully examine the mathematical relationships in combinatorial
problems; avoid assuming constant ratios between counts without
rigorous justification, as combinatorial structures can change
dramatically with scale.
When encountering problems involving alternating actions, ensure you
always begin with the first unfulfilled condition (e.g., the first
closed locker) to accurately simulate the process.
```

GradPO-refined prompt

```
Solve the following math problem.  Think carefully consider step
carefully before answering.
To improve problem-solving accuracy, apply the following strategies
throughout your reasoning process and solution development.  Maintain
a focus on precise calculations and accurate representation of
problem constraints.
After completing your reasoning, meticulously review your final
calculation and ensure the answer is presented in the correct format
and quantities as requested by the problem.
Carefully examine the mathematical relationships in permutation
problems; avoid assuming constant ratios between counts without
rigorous justification, as combinatorial structures can change
dramatically with scale.
When encountering problems involving alternating actions, ensure you
always begin with the first unfulfilled condition (e.g., the first
closed space) to accurately simulate the process.
```

GradPO-Prob-refined prompt

```
Solve the problem below, think steps and carefully consider before
answering.  To improve problem-solving accuracy, apply the following
strategies throughout your reasoning process and final answer
generation.  Maintain a focus on precise calculations and accurate
representation of problem constraints.  After completing your
reasoning, meticulously review your final calculation and ensure the
answer is presented in the correct format and units as requested by
the problem.  Carefully examine the mathematical relationships in
combinatorial problems; avoid assuming constant ratios between counts
without rigorous justification, as combinatorial structures can
change dramatically with scale.  When encountering problems involving
alternating actions, ensure you always begin with the first
unfulfilled condition (e.g., the first closed lock) to accurately
simulate the process.
```