GEPA: REFLECTIVE PROMPT EVOLUTION CAN OUTPER-FORM REINFORCEMENT LEARNING

Anonymous authors

 Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly adapted to downstream tasks via reinforcement learning (RL) methods like Group Relative Policy Optimization (GRPO), which often require thousands of rollouts to learn new tasks. We argue that the interpretable nature of *language* often provides a much richer learning medium for LLMs, compared to policy gradients derived from sparse, scalar rewards. To test this, we introduce GEPA (Genetic-Pareto), a prompt optimizer that thoroughly incorporates *natural language reflection* to learn high-level rules from trial and error. Given any AI system containing one or more LLM prompts, GEPA samples trajectories (e.g., reasoning, tool calls, and tool outputs) and reflects on them in natural language to diagnose problems, propose and test prompt updates, and combine complementary lessons from the Pareto frontier of its own attempts. As a result of GEPA's design, it can often turn even just a few rollouts into a large quality gain. Across four tasks, GEPA outperforms GRPO by 10% on average and by up to 20%, while using up to 35x fewer rollouts. GEPA also outperforms the leading prompt optimizer, MIPROv2, by over 10% (e.g., +10% accuracy on AIME-2025).

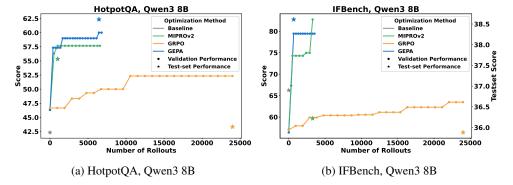


Figure 1: A comparison of learning behavior of the GEPA prompt optimizer against a state-of-the-art prompt optimizer (MIPROv2) and GRPO (24,000 rollouts). As more rollouts are sampled, the prompt optimizers can learn much more quickly than GRPO. GEPA substantially outperforms both GRPO and MIPROv2 in final score. The Test-set star markers demonstrate the performance gap in a held-out set of questions.

1 Introduction

Large language models (LLMs) have enabled development of agents and systems that combine fuzzy natural-language specifications with tools like retrieval and code execution. This raises the question of how LLMs should be optimized for downstream performance. One popular approach is Reinforcement Learning

with Verifiable Rewards (RLVR), e.g. with Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which treats success metrics as end-of-rollout scalar rewards used to estimate policy gradients (Lambert, 2025). While these RL approaches are effective, they typically require tens of thousands of rollouts in practice to fit new tasks. For example, recent works leveraging GRPO typically use up to hundreds of thousands of rollouts for training (Chen et al., 2025b; Wu et al., 2025b; Zhang et al., 2025; Jin et al., 2025; Si et al., 2025; Wang et al., 2025a; Chen et al., 2025a; Sha et al., 2025; Lin et al., 2025a; Peng et al., 2025; Song et al., 2025). This sample inefficiency can quickly become a serious bottleneck: many downstream LLM applications invoke expensive tool calls, have limited inference budget for sampling from the LLM itself, or simply cannot finetune the weights of the largest or best-performing LLMs.

We observe that rollouts sampled from even highly sophisticated LLM systems can be serialized into traces of natural language: they contain nothing but the instructions of each LLM module, the resulting LLM reasoning chains, tool calls, and potentially the internal workings of the reward function (e.g., compiler error messages, before they are collapsed into scalar rewards). Because such serialized trajectories are readily understood by modern LLMs, we argue that *algorithms that learn deliberately in natural language by reflecting on these trajectories* can make more effective use of the strong language priors that LLMs have, compared with standard RL approaches.

We introduce GEPA (Genetic-Pareto), a *reflective* prompt optimizer for compound AI systems that merges textual reflection with multi-objective evolutionary search. GEPA iteratively mutates prompts using natural language feedback drawn from new rollouts. In each mutation, the candidate prompt is derived from an ancestor, accumulating high-level lessons derived from observations and LLM feedback. To avoid local optima that afflict greedy prompt optimization, GEPA maintains a Pareto front: instead of evolving only the global best prompt, it stochastically explores the top-performing prompts for each problem instance. This diversification enables robust generalization and mitigates getting stuck in local minima.

We evaluate GEPA across multi-hop reasoning (HotpotQA; Yang et al. 2018), Math (AIME, LiveBench-Math; Balunović et al. (2025); White et al. (2025)), instruction following (IFBench; Pyatkin et al. 2025b), privacy-aware delegation (PUPA; Li et al. 2025a), and retrieval-augmented verification (HoVer; Jiang et al. 2020), using both open (Qwen3 8B; Yang et al. 2025; Team 2025) and proprietary (GPT-4.1 Mini; OpenAI 2025) models. We find that GEPA generalizes well and is highly sample-efficient: on Qwen3 8B, it outperforms GRPO (24k rollouts) by up to 19% while using up to $35 \times$ fewer rollouts, with an average gain of +10% across tasks. GEPA also surpasses the prior state-of-the-art, MIPROv2 (Opsahl-Ong et al., 2024), on all benchmarks and models, achieving +13% aggregate gains, over double MIPROv2's +5.6%.

Qualitatively, GEPA-learned prompts are quite rich. Figure 2 shows excerpts from a prompt crafted for the query-creation module of a multi-hop question answering system used in HotPotQA, and Figure 25 shows that even a single *reflective* update often yields large gains. These results highlight that reflective prompt evolution with language feedback enables improved sample efficiency and robust generalization, offering a practical approach to optimizing complex AI workflows in data- or budget-constrained environments. We also present preliminary evidence of GEPA's broader utility as an inference-time search strategy for code optimization on NPUEval (Kalade & Schelle, 2025) and KernelBench (Ouyang et al., 2025) in Section 5.

2 PROBLEM STATEMENT

We follow related work in defining a **compound AI system** as any modular system composed of one or more language model (LLM) invocations, potentially interleaved with external tool calls, orchestrated through arbitrary control flow. This definition subsumes a broad class of real-world LLM-based AI systems, including *agents*, *multi-agent systems*, and general-purpose scaffolding techniques like ReAct (Yao et al., 2023), Archon (Saad-Falcon et al., 2025), etc. Following Soylu et al. (2024); Khattab et al. (2024); Opsahl-Ong et al. (2024); Tan et al. (2025), we formalize such a system as $\Phi = (M, C, \mathcal{X}, \mathcal{Y})$, where $M = \langle M_1, \dots, M_{|M|} \rangle$

096 097

098 100 101

107

108

113

114

> 120 121 122

> 123

124

125

119

134

135

136

Seed Prompt for Second-Hop of Multi-Hop QA System

Given the fields question, summary_1, produce the fields query.

GEPA's Optimized Prompt for Second-Hop of Multi-Hop QA System, GPT-4.1 Mini

You will be given two input fields: question and summary_1. Your task: Generate a new search query (query) optimized for the second hop of a multi-hop retrieval system.

- · The original user question is typically complex and requires information from multiple documents to answer.
- · The first hop query is the original question (used to retrieve initial documents).
- · Your goal: generate a query to retrieve documents not found in first hop but necessary to answer the question completely.

 $\textbf{Input Understanding:} \ \text{question is the original multi-hop question posed by the user. } \ \text{summary_1} \ \text{is a concise summary of information from a linear example of the property of$ document retrieved in the first hop, which partially addresses the question.

- Your generated query aims to find the *missing pieces* of information needed to fully answer the question. . . .
- The query must retrieve relevant documents NOT found in first hop ... for final answer extraction.

Key Observations and Lessons:

• First-hop documents often cover one entity or aspect. (... truncated for space ...)

How to Build the Query:

• Identify entities or topics mentioned in summary_1 that are related but different from first-hop documents. (...truncated for space ...)

Practical Strategy:

- · Read the summary_1 carefully to spot references to bigger contexts or other entities not covered in the first hop.
- · Ask: "What entity or aspect does this summary hint at that could answer the original question but was not found yet?"
- · Formulate a precise, focused factual query targeting that entity or concept to retrieve the missing documents.

Output:

- · Produce query as a clear, concise question or keyword phrase designed for efficient retrieval of second-hop documents.
- · Ensure the query relates logically to the original question while targeting the broader or complementary knowledge identified in summary_1. Do not include the original question or simply rephrase it. Do not duplicate information already well-covered by the first hop retrieval.

Figure 2: This figure shows an example prompt generated by GEPA for the second-hop document retrieval to be performed in a multi-hop question-answer system, along with the seed prompt it started with. Appendix N compares GEPA's prompts for all tasks with prompts generated by MIPROv2.

denotes language modules, C specifies control flow logic, and \mathcal{X}, \mathcal{Y} are global input/output schemas. Each module $M_i = (\pi_i, \theta_i, \mathcal{X}_i, \mathcal{Y}_i)$ is an LLM subcomponent: π_i is its (system) prompt including instructions and few-shot demonstrations; θ_i the underlying model weights; \mathcal{X}_i , \mathcal{Y}_i are input/output schemas. At runtime, C orchestrates the sequencing and invocation of modules—e.g., passing outputs from one module to another, invoking modules conditionally, or leveraging tool APIs. This way, C can invoke different modules in any order multiples of times.

Given Φ , let $\Pi_{\Phi} = \langle \pi_1, \dots, \pi_{|M|} \rangle$ denote the collection of all module prompts and $\Theta_{\Phi} = \langle \theta_1, \dots, \theta_{|M|} \rangle$ the set of module weights. The learnable parameters are thus $\langle \Pi, \Theta \rangle_{\Phi}$. For a task instance (x, m)—where xmaps to the input schema \mathcal{X} and m contains evaluator metadata (e.g., gold answers, evaluation rubrics, code unit tests)—the system induces an output $y = \Phi(x; \langle \Pi, \Theta \rangle_{\Phi})$. A metric $\mu: \mathcal{Y} \times \mathcal{M} \to [0, 1]$ then measures the output quality of y with respect to metadata m (for example by calculating, exact match, F1, pass rate, etc.). The optimization problem is thus defined as follows, where \mathcal{T} is a task distribution:

$$\langle \Pi^*, \Theta^* \rangle_{\Phi} = \arg \max_{\langle \Pi, \Theta \rangle_{\Phi}} \mathbb{E}_{(x,m) \sim \mathcal{T}} \left[\mu \left(\Phi(x; \langle \Pi, \Theta \rangle_{\Phi}), m \right) \right]. \tag{1}$$

Figure 3: GEPA proposes a new candidate in every iteration by improving existing candidates using one of the two strategies (Reflective Prompt Mutation (Section 3) or System Aware Merge (Appendix D.1)), first evaluating them on a minibatch, and if improved, evaluating on a larger dataset. Instead of selecting the best performing candidate to mutate always, which can lead to a local-optimum, GEPA introduces Pareto-based candidate sampling (Section 3.1), which filters and samples from the list of best candidates *per* task, ensuring sufficient diversity. Overall, these design decisions allow GEPA to be highly sample-efficient while demonstrating strong generalization.

Sample-Efficient Optimization. In many real-world scenarios, rollouts—concretely, invocations of Φ plus evaluation by μ —are often computationally, monetarily, or timewise expensive. The optimizer is thus limited to at most B rollouts on a dataset $\mathcal{D}_{\text{train}} = \{(x,m)_i\}_{i=1}^N$ with full access to μ . The goal is to identify parameters $\langle \Pi^*, \Theta^* \rangle_{\Phi}$ that maximize held-out performance, subject to not exceeding the rollout budget B:

$$\langle \Pi^*, \Theta^* \rangle_{\Phi} = \arg \max_{\langle \Pi, \Theta \rangle_{\Phi}} \mathbb{E}_{(x,m) \sim \mathcal{T}} \left[\mu \left(\Phi(x; \langle \Pi, \Theta \rangle_{\Phi}), m \right) \right], \quad \text{s.t.} \quad \# \text{rollouts} \leq B. \tag{2}$$

The core challenge, then, is: *How do we extract maximal learning signal from every expensive rollout to enable effective adaptation of complex, modular AI systems in low-data or budget-constrained settings?*

3 GEPA: REFLECTIVE PROMPT EVOLUTION

We introduce GEPA (Genetic-Pareto), a sample-efficient optimizer for compound AI systems motivated by three core principles: genetic prompt evolution (Section 3), reflection using natural language feedback (Section 3), and Pareto-based candidate selection (Section 3.1). Figure 3 gives an overview of GEPA and the full GEPA algorithm is formalized in Figure 5. GEPA receives the following inputs: A system Φ instantiated with simple prompts to be optimized, training dataset D_{train} (consisting of task instances (x,m) as described in Section 2), the standard evaluation metric μ for the task, a feedback function μ_f (introduced in Section 3) and the total rollout budget B. Note that GEPA evolves only the set of prompts, denoted as Π_{Φ} , whereas the underlying LLM weights, denoted by Θ_{Φ} remains fixed.

Genetic Optimization Loop: Given an AI system Φ , the goal is to identify parameters Π_{Φ} that maximize task performance. GEPA begins with a *candidate pool* \mathcal{P} containing only the base system, where each *candidate* is a concrete instantiation of $\langle \Pi, \Theta_{frozen} \rangle_{\Phi}$. It then enters an optimization loop, repeatedly proposing new candidates until the evaluation budget is exhausted. Candidates are derived from existing ones via *reflective mutation* or *crossover*, guided by feedback from rollouts, with each inheriting learning signals from its parents and its own rollout so that GEPA accumulates knowledge along the genetic tree. In each iteration, GEPA (i) selects promising candidates, (ii) proposes and evaluates a variant on a minibatch of tasks, and (iii) if it outperforms its parent(s), adds it to \mathcal{P} with ancestry records and evaluate on D_{pareto} , the validation set used for selection. After the budget is exhausted, GEPA returns the candidate with the best aggregate performance on D_{pareto} .

Reflective Prompt Mutation: Natural language traces generated during the execution of a compound AI system offer rich *visibility* into the behavior and responsibilities of each module, as they capture the intermediate inferences and underlying reasoning steps. When these traces are paired with the final outcome of the system (e.g., success or failure), they provide substantial *diagnostic* value, allowing practitioners to trace errors or successes back to specific decisions made at the module level. LLMs can leverage these traces via *reflection* to perform implicit credit assignment, attributing responsibility for the final outcome to the relevant modules. This process of *reflection* can then be used to make targeted updates to individual modules, making large and effective updates to the whole system's behavior.

Given a *candidate* to mutate in the current iteration of the optimization loop, GEPA updates the system with a new set of parameters, selects a target module to improve (via round robin), and generates rollouts on a minibatch, recording outcomes. From execution traces, GEPA extracts the module's inputs, outputs, and reasoning, then uses an LLM to reflectively attribute successes or failures to prompt elements and propose revised instructions. A new candidate is formed by copying the current candidate, with the updated parameters for the target module's prompt. The meta-prompt for reflective prompt updates is shown in Appendix C.

Evaluation traces as diagnostic signals: The text that LLMs produce is the execution trace of the AI system. The text that the environment produces to compute the reward (e.g. compiler error messages before giving reward 0) is the evaluation trace. Beyond reflection on execution traces, we identify a second valuable source of diagnostic information in the evaluation traces. Many evaluation metrics apply rich strategies (e.g., code evaluation may involve compilation, execution, and profiling), producing natural language traces before computing a scalar reward. We propose leveraging these evaluation traces for reflective credit assignment and targeted prompt updates. GEPA achieves this by extending rewards μ into a feedback function μ_f that extracts textual traces during evaluation and returns them with the final score as feedback_text. When available, such feedback can even be module-specific (e.g. in multi-hop systems the evaluator may provide feedback after each hop). Additionally, D_{train} can be augmented with human-written explanations for each instance; during reflection, GEPA can consume these explanations as auxiliary feedback_text to guide targeted prompt updates, even when natural-language feedback from rollouts is limited or unavailable.

3.1 PARETO-BASED CANDIDATE SELECTION

GEPA is a highly modular algorithm that supports various strategies for candidate selection, with the choice of strategy governing the exploration—exploitation tradeoff. A naive approach is to always select the best-performing candidate, but this often traps the optimizer in a local optimum: once a dominant strategy is found, it becomes difficult to surpass, and the optimizer exhausts its budget without learning new, potentially better strategies. Figure 4a illustrates this behavior: after finding one new strategy (the first child node), the search repeatedly attempts to refine it, fails to improve, and ultimately depletes the budget.

To address this, GEPA employs a Pareto-based "illumination" strategy (Mouret & Clune, 2015), shown in Algorithm 2. For each training instance, GEPA records the highest score across all candidates, forming a Pareto frontier. Candidates that achieve the best score on at least one task are retained, while strictly

Table 1: Benchmark results for different optimizers with Qwen3 8B. GEPA and GEPA+Merge achieve better performance with fewer rollouts on almost all benchmarks. For example, for IFBench, GEPA used 678 rollouts to achieve 38.61%, outperforming GRPO's test set score of 35.88% with 24,000 rollouts.

Qwen3 8B	HotpotQA	IFBench	Hover	PUPA	Aggregate	Improvement
Baseline	42.33	36.90	35.33	80.82	48.85	_
MIPROv2	55.33	36.22	47.33	81.55	55.11	+6.26
GRPO	43.33	35.88	38.67	86.66	51.14	+2.29
GEPA	62.33	38.61	52.33	91.85	61.28	+12.44
GEPA+Merge	64.33	28.23	51.67	86.26	57.62	+8.78

Table 2: Benchmark results for different optimizers with GPT-4.1 Mini. Since the system consists of only one prompt for both AIME and LiveBench-Math, GEPA and GEPA+Merge are equivalent. As a prompt-optimization system, GEPA works off-the-shelf on *closed-source* models as well, outperforming the state-of-the-art prompt optimizer MIPROv2.

GPT-4.1 Mini	AIME-2025	LiveBench-Math	HotpotQA	IFBench	Hover	PUPA	Aggregate	Improvement
Baseline	49.33	58.20	38.00	47.79	46.33	78.57	53.03	_
MIPROv2	51.33	61.84	58.00	49.15	48.33	83.37	58.67	+5.64
GEPA	59.33	64.13	69.00	52.72	51.67	94.47	65.22	+12.19
GEPA+Merge	59.33*	64.13*	65.67	55.95	56.67	96.46	66.36	+13.33

dominated ones are pruned. From this pruned set, GEPA stochastically samples a candidate, weighting probabilities by how many tasks each candidate leads. This strategy helps GEPA escape local optima without inflating the search, efficiently balancing exploration and exploitation by focusing resources on candidates that embody "winning" strategies within the optimization budget.

4 EVALUATION

We adopt a standard train/validation/test split. Optimizers have full access to the train split, including text and labels, for program tuning. Although optimizers may monitor the performance of candidate parameters (like model checkpoints) by tracking scores on the validation set (to implement early stopping, for example), direct access to the content of validation instances is restricted. We evaluate on six benchmarks—AIME-2025 (Balunović et al., 2025), LiveBench-Math (White et al., 2025), HotpotQA (Yang et al., 2018), IF-Bench (Pyatkin et al., 2025b), HoVer (Jiang et al., 2020), and PUPA (Li et al., 2025a)—each paired with existing compound AI systems and feedback functions. Experiments use Qwen3 8B (Yang et al., 2025) and GPT-4.1 Mini (OpenAI, 2025) with standardized inference settings, and compare against state-of-the-art optimizers MIPROv2 (Opsahl-Ong et al., 2024) and GRPO¹ (Shao et al., 2024). Appendix E provides further details on benchmarks, systems, and feedback functions (Subsection E.1); models and inference settings (Subsection E.2); and optimizer configurations (Subsection E.3).

Table 1 and Figure 6 summarize our main results, from which we derive the following observations:

Observation 1: Reflective Prompt Evolution is highly sample-efficient and can outperform weight-space reinforcement learning: Across four benchmarks, GEPA adapts rapidly and generalizes robustly in compound AI systems—beating GRPO (24,000 rollouts) by up to 19% while using up to $35\times$ fewer rollouts. It reaches optimal test performance with $4-35\times$ fewer rollouts and exceeds GRPO on the four tasks by 19.0%, 2.73%, 13.66%, and 5.19%. GEPA matches GRPO's best validation after only 402, 330,

¹We use LoRA for GRPO due to its low cost and successful adoption with GRPO (Wang et al., 2025b; Xu et al., 2025b; Li et al., 2025b; Yue et al., 2025; Sun et al., 2025; Hayou et al., 2025; Zhao et al., 2025; Teknium et al., 2024; Zhao et al., 2024; Sidahmed et al., 2024). Additionally, we explore full-parameter finetuning. Figure 9 shows a similar result comparing GEPA to GRPO with full finetuning.

Table 3: Comparing candidate selection strategies across different tasks with Qwen3 8B. At each step, SelectBestCandidate evolves the candidate prompt only from the top-scoring candidate, which is more prone to getting stuck in a local optimum.

Qwen3 8B	HotpotQA	IFBench	Hover	PUPA	Aggregate	Improvement
SelectBestCandidate	58.33	30.44	45.33	85.45	54.89	_
GEPA	62.33	38.61	52.33	91.85	61.28	+6.4

1179, and 306 rollouts—up to $78 \times$ greater sample efficiency. GEPA+Merge widens the gap, outperforming GRPO by 21% at a comparable rollout budget to GEPA.

The majority of GEPA's rollout budget is spent on validation, where scores are utilized solely for candidate selection and not for producing learning signals. If we restrict the analysis to train set rollouts, GEPA requires only 79 to 737 rollouts to reach optimal performance. To match GRPO's best validation scores, GEPA achieves this with only 102, 32, 6, and 179 train rollouts for four tasks, respectively, underscoring the high sample efficiency of learning based on reflective prompt evolution.

Since tracking candidates' validation performance accounts for majority of GEPA's rollout budget, sample efficiency can be further improved by evaluating on a smaller validation set or by tracking scores on dynamically selected validation subsets instead of the full set—both of which we propose as directions for future work. Figures 1a, 1b, 12c and 13c show the full performance-vs-rollouts curve for all optimizers over benchmarks HotpotQA, IFBench, HoVer and PUPA, respectively.

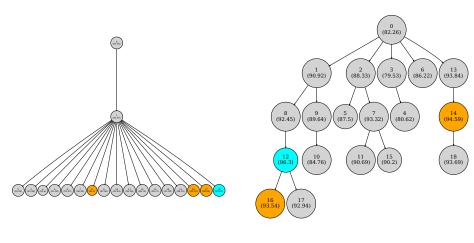
Observation 2: Reflective prompt evolution enables instruction-optimization alone to outperform joint instruction and few-shot optimization: We compare GEPA with MIPROv2, a state-of-the-art instruction and few-shot optimizer, using two leading models across six diverse tasks, and observe that GEPA consistently outperforms MIPROv2 in all settings, achieving margins as high as 11.1% for GPT-4.1 mini and 10.3% for Qwen3 8B. Further, GEPA and GEPA+Merge more than double the aggregate gains over baseline seen with MIPROv2 across all benchmarks and models (+13.33% and +12.19% vs +5.64% for MIPROv2).

While prior works such as Opsahl-Ong et al. (2024) and Wan et al. (2024) have provided compelling evidence for the effectiveness of few-shot example optimization—often outperforming instruction-based approaches—our findings suggest an exciting shift in this trend. We attribute this primarily to recent advances in the instruction-following and self-reflective abilities of LLMs, as well as the design choices in GEPA that capitalize on these improved capabilities. To further contextualize our findings, we redo the study on generalization gap (the difference between validation and test set performance for optimized prompts) as proposed by Wan et al. (2024). The results presented in Figure 14 reinforce these observations: reflectively evolved instructions now demonstrate a lower generalization gap, underscoring both advancements in model capabilities and the benefits of GEPA's design. We see this as a reflection of the continuous evolution of LLMs and GEPA's ability to effectively leverage these improvements.

We provide the full-length optimized prompts produced by GEPA for all systems, benchmarks, and models in Appendix N, alongside MIPROv2 prompts. Notably, in contrast to prior findings where instruction optimization yielded improvements primarily through quasi-exemplars (Wan et al., 2024), GEPA's prompts frequently contain detailed *declarative* instructions for completing the task, as illustrated in Figure 2.

Observation 3: The next-candidate selection strategy strongly influences the optimization trajectory and final performance, with Pareto-based sampling providing a distinct advantage.

GEPA refines prompts iteratively with rollout feedback; to test our Pareto-based selection, we compare against a baseline that always picks the best-performing candidate in the SelectBestCandidate strategy. As shown in Table 3, this baseline often yields suboptimal exploration of the prompt search space, leading



(a) SelectBestCandidate Strategy

(b) Pareto-based candidate sampling.

Figure 4: Comparing the impact of different candidate selection strategies. (Left) As can be seen, selecting the best-performing candidate in every iteration led to a local-optima after one iteration, leading to suboptimal search performance. (Right) On the other hand, using pareto-based candidate selection strategy, GEPA was able to generate a balanced search tree, finding a better performing program within the same budget.

to poor performance. GEPA with Pareto-based sampling outperforms the SelectBestCandidate strategy by up to 8.17%, with an aggregate margin of +6.4% across all benchmarks. Figure 4 highlights the difference in optimization trajectories: always choosing the current best candidate gives immediate improvement but quickly stalls, wasting rollouts on a single candidate. In contrast, our Pareto-based method expands the search by considering all Pareto-optimal candidates (all "winning" strategies found so far), balancing exploration and exploitation and converging to a higher-performing solution within the same rollout budget.

We also find that instruction-optimized prompts are more efficient and generalize better than fewshot demonstrations (Observation 4). Further, System-aware crossover strategies such as Merge can improve performance, but their effectiveness depends on budget allocation between mutation and crossover and on timing (Observation 5). See Appendix F for further discussion.

5 EXTENDED APPLICATIONS OF GEPA

GEPA For Inference-Time Search GEPA can be used as an inference-time search technique by setting the validation set equal to the training set $(D_{\text{val}} = D_{\text{train}})$, under which GEPA iteratively proposes better solutions for each target task. We evaluate this approach on hardware-specific code generation: writing AMD NPU kernels (NPUEval (Kalade & Schelle, 2025)) and generating NVIDIA GPU's CUDA kernels (Kernel-Bench (Ouyang et al., 2025)) using GPT-40 OpenAI (2024). GEPA raises mean kernel vector utilization on NPUEval from 4.25% (sequential refinement baseline) to 30.52%, and increases the fraction of KernelBench tasks with CUDA code surpassing baseline-PyTorch speed from close to 0% to over 20%. Full experimental details are provided in Appendix G.

GEPA for Adversarial Prompt Search GEPA can discover adversarial prompts by inverting the reward—finding instructions that minimize task performance. On AIME-2025 with GPT-5 Mini, starting from a simple instruction, GEPA evolved a trivia-style distractor that reduced pass@1 from 76% to 10%. Adversarial search drew on AIME 2022–2024 problems, with evaluation on AIME-2025 using 5 runs per problem

(30 problems; 150 generations). This robustness probe surfaces brittle instruction-following interactions (e.g., trivia with strict formatting) and yields reusable stress tests and regression suites for deployment and show that reflective prompt evolution is also effective for stress-testing them by locating universal, query-agnostic perturbations that sharply degrade accuracy. These findings parallel recent results on query-agnostic adversarial triggers, which report two- to seven-fold increases in error rates across model families; in our AIME-2025 setting, GEPA's learned universal instruction yields a comparable 3.8× increase (errors from 24% to 90%) (Rajeev et al., 2025). Full experimental details are provided in Appendix H.

6 RELATED WORK

Prompt optimization improves LLMs but often needs manual expertise; for instance, chain-of-thought prompting Wei et al. (2023). To scale this approach, recent methods use LLMs to optimize prompts automatically (Zhou et al., 2022; Yang et al., 2024; Agarwal et al., 2024; Fernando et al., 2024). GEPA leverages LLMs, but differs by incorporating textual environment feedback, Pareto-aware search over candidates, and evolution strategies per submodule within an AI system.

Evolutionary algorithms have been used to optimize prompts, e.g., EvoPrompt (Guo et al., 2024), which evolves prompt populations. GEPA additionally uses domain-specific feedback for targeted mutations achieving higher sample efficiency. AlphaEvolve (Novikov et al., 2025) and OpenEvolve (Sharma, 2025) apply evolutionary search directly to code rewriting, excelling when problem solution can be codified. While AlphaEvolve targets a single hard problem, GEPA brings evolution to prompts across domains, combining Pareto-frontier optimization and prompt evolution to transfer tactics from related problems.

Feedback-driven improvement often uses reinforcement learning, such as majority voting signals (Zuo et al., 2025), but RL can be sample-inefficient when rewards are slow to compute. An alternative is learning in the language space: in-context bandit/self-bootstrapping (Monea et al., 2025; Xu et al., 2025a; Feng et al., 2025; Cheng et al., 2024), workflow memory and skills (Wang et al., 2024; 2025c), and test-time strategy synthesis via Dynamic Cheatsheet (Suzgun et al., 2025), reasoning cache (Chen et al., 2025c). GEPA instead uses examples to propose new *instructions*, yielding task-specific rules.

To optimize compound AI systems and agents (Lin et al., 2025b), DSPy (Khattab et al., 2022; 2024) searches/bootstraps few-shot examples, TextGrad (Yuksekgonul et al., 2025) backpropagates textual feedback, and MIPROv2 (Opsahl-Ong et al., 2024) jointly aligns instructions and examples via Bayesian optimization; these largely rely on global rewards. Optimas (Wu et al., 2025a) introduces globally aligned local rewards per module. GEPA combines global rewards with environment textual feedback per module and maintains a Pareto frontier over individual data instances, matching prompts/agent design to specific examples. The Pareto-guided evolution lets GEPA explore diverse prompt/code/agent design strategies before converging to a robust, generalizable set.

7 CONCLUSION

We introduced GEPA, a prompt optimizer for arbitrary LLM agents and workflows that leverages explicit reflection and Pareto-based selection, showing superior sample efficiency compared to reinforcement learning (GRPO), while outperforming leading prompt optimizers (MIPROv2). By explicitly incorporating natural language feedback and maintaining a diverse pool of Pareto-optimal candidates, GEPA rapidly adapts AI systems to new tasks. Our results across benchmarks and models suggest that language-based reflection can offer a scalable strategy for optimizing complex real-world AI workflows, especially in resource-constrained settings. GEPA also shows promise as an inference-time search strategy, showing the ability to write code in challenging domains.

REFERENCES

- Advanced Micro Devices. Amd xdnaTM architecture. https://www.amd.com/en/technologies/xdna.html#xdna2, 2025. Accessed on: 2025-07-23.
- Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. Promptwizard: Task-aware prompt optimization framework, 2024. URL https://arxiv.org/abs/2405.18369.
- Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv* preprint *arXiv*:2308.16369, 2023.
- Meta AI. llama-prompt-ops: An open-source tool for seamless migration from other llms to llama, and for general prompt optimization. https://github.com/meta-llama/llama-prompt-ops, 2025. Accessed: 2025-07-11.
- Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Aime-2025, February 2025. URL https://huggingface.co/datasets/MathArena/aime_2025. Dataset from MathArena: Evaluating LLMs on Uncontaminated Math Competitions.
- Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.
- Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning, March 2025a. URL http://arxiv.org/abs/2503.19470. arXiv:2503.19470 [cs].
- Peter Chen, Xiaopeng Li, Ziniu Li, Xi Chen, and Tianyi Lin. Spectral Policy Optimization: Coloring your Incorrect Reasoning in GRPO, May 2025b. URL http://arxiv.org/abs/2505.11595.arXiv:2505.11595 [cs].
- Peter Baile Chen, Yi Zhang, Dan Roth, Samuel Madden, Jacob Andreas, and Michael Cafarella. Log-augmented generation: Scaling test-time reasoning with reusable computation, 2025c. URL https://arxiv.org/abs/2505.14398.
- Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization with rich feedback, execution traces, and llms, 2024. URL https://arxiv.org/abs/2406.16218.
- Xidong Feng, Bo Liu, Yan Song, Haotian Fu, Ziyu Wan, Girish A. Koushik, Zhiyuan Hu, Mengyue Yang, Ying Wen, and Jun Wang. Natural language reinforcement learning, 2025. URL https://arxiv.org/abs/2411.14251.
- Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rockt" aschel. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.
- Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz, Kourosh Hakhamaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Evolving skyrl into a highly-modular rl framework, 2025. Notion Blog.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yu-jiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIs08.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. PLoP: Precise LoRA Placement for Efficient Finetuning of Large Models, June 2025. URL http://arxiv.org/abs/2506.20629. arXiv:2506.20629 [cs].

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.309. URL https://aclanthology.org/2020.findings-emnlp.309/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning, July 2025. URL http://arxiv.org/abs/2503.09516. arXiv:2503.09516 [cs].

Sarunas Kalade and Graham Schelle. Npueval: Optimizing npu kernels with llms and open source compilers, 2025. URL https://arxiv.org/abs/2507.14403.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv* preprint arXiv:2212.14024, 2022.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Nathan Lambert. Policy gradient algorithms. In *RLHF Book: Reinforcement Learning from Human Feedback*, chapter 11. RLHF Book, 2025. URL https://rlhfbook.com/c/11-policy-gradients.html. Accessed July 16, 2025.

Siyan Li, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. PAPILLON: Privacy preservation from Internet-based and local language model ensembles. 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 1: Long Papers)*, pp. 3371–3390, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.173. URL https://aclanthology.org/2025.naacl-long.173/.

Xianming Li, Aamir Shakir, Rui Huang, Julius Lipp, and Jing Li. ProRank: Prompt Warmup via Reinforcement Learning for Small Language Models Reranking, June 2025b. URL http://arxiv.org/abs/2506.03487. arXiv:2506.03487 [cs].

Chenyu Lin, Yilin Wen, Du Su, Fei Sun, Muhan Chen, Chenfu Bao, and Zhonghou Lv. Knowledgeable-r1:
Policy Optimization for Knowledge Exploration in Retrieval-Augmented Generation, June 2025a. URL
http://arxiv.org/abs/2506.05154. arXiv:2506.05154 [cs].

- Matthieu Lin, Jenny Sheng, Andrew Zhao, Shenzhi Wang, Yang Yue, Victor Shea Jay Huang, Huan Liu, Jun Liu, Gao Huang, and Yong-Jin Liu. Training of scaffolded language models with language supervision: A survey, 2025b. URL https://arxiv.org/abs/2410.16392.
- Shu Liu, Sumanth Hegde, Shiyi Cao, Alan Zhu, Dacheng Li, Tyler Griggs, Eric Tang, Akshay Malik, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-sql: Matching gpt-4o and o4-mini on text2sql with multi-turn rl, 2025.
- Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. Llms are in-context bandit reinforcement learners, 2025. URL https://arxiv.org/abs/2410.05362.
- Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015. URL https://arxiv.org/abs/1504.04909.
- Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. Technical report, Google DeepMind, 2025. URL https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/AlphaEvolve.pdf. White paper.
- OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2025-09-08.
- OpenAI. GPT-4.1 series, 2025. Large language model series, released April 2025. https://openai.com/index/gpt-4-1/.
- Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.
- Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Re, and Azalia Mirhoseini. Kernelbench: Can LLMs write efficient GPU kernels? In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025. URL https://openreview.net/forum?id=k6V4jb8jkX.
- Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. VerIF: Verification Engineering for Reinforcement Learning in Instruction Following, June 2025. URL http://arxiv.org/abs/2506.09942. arXiv:2506.09942 [cs].
- Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. IF-RLVR-Train, July 2025a. URL https://huggingface.co/datasets/allenai/IF_multi_constraints_upto5.
- Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025b. URL https://arxiv.org/abs/2507.02833.

Meghana Rajeev, Rajkumar Ramamurthy, Prapti Trivedi, Vikas Yadav, Oluwanifemi Bamgbose, Sathwik Tejaswi Madhusudan, James Zou, and Nazneen Rajani. Cats confuse reasoning llm: Query agnostic adversarial triggers for reasoning models, 2025. URL https://arxiv.org/abs/2503.01781.

- Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, E. Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, and Azalia Mirhoseini. Archon: An architecture search framework for inference-time techniques, 2025. URL https://arxiv.org/abs/2409.15254.
- Zeyang Sha, Shiwen Cui, and Weiqiang Wang. SEM: Reinforcement Learning for Search-Efficient Large Language Models, May 2025. URL http://arxiv.org/abs/2505.07903. arXiv:2505.07903 [cs].
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.
- Asankhaya Sharma. Openevolve: Open-source implementation of alphaevolve. https://github.com/codelion/openevolve, 2025. GitHub.
- Shuzheng Si, Haozhe Zhao, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Bofei Gao, Kangyang Luo, Wenhao Li, Yufei Huang, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. Teaching Large Language Models to Maintain Contextual Faithfulness via Synthetic Tasks and Reinforcement Learning, May 2025. URL http://arxiv.org/abs/2505.16483. arXiv:2505.16483 [cs].
- Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Parameter Efficient Reinforcement Learning from Human Feedback, September 2024. URL http://arxiv.org/abs/2403.10704. arXiv:2403.10704 [cs].
- Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning, March 2025. URL http://arxiv.org/abs/2503.05592. arXiv:2503.05592 [cs].
- Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps that work better together, 2024. URL https://arxiv.org/abs/2407.10930.
- Zhongxiang Sun, Qipeng Wang, Haoyu Wang, Xiao Zhang, and Jun Xu. Detection and Mitigation of Hallucination in Large Reasoning Models: A Mechanistic Perspective, May 2025. URL http://arxiv.org/abs/2505.12886. arXiv:2505.12886 [cs].
- Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025. URL https://arxiv.org/abs/2504.07952.
- Shangyin Tan, Lakshya A Agrawal, Arnav Singhvi, Liheng Lai, Michael J Ryan, Dan Klein, Omar Khattab, Koushik Sen, and Matei Zaharia. Langprobe: a language programs benchmark, 2025. URL https://arxiv.org/abs/2502.20315.
- Qwen Team. Qwen/qwen3-8b. https://huggingface.co/Qwen/Qwen3-8B, 2025. Accessed: 2025-07-11.
- Ryan Teknium, Jeffrey Quesnelle, and Chen Guang. Hermes 3 Technical Report, August 2024. URL http://arxiv.org/abs/2408.11857. arXiv:2408.11857 [cs].

615

616

617

618 619

620

621

622

623

624 625

626

627

628

629

630 631

632

633

634

635

636

637

638

639 640

641

642 643

644

645 646

647

648

649

650

651

652

653

654

655

656

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Arik. Teach better or show smarter? on instructions 612 and exemplars in automatic prompt optimization. Advances in Neural Information Processing Systems, 613 37:58174-58244, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ 614 6b031defd145b02bed031093d8797bb3-Abstract-Conference.html.

- Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting Less is Reasoning More! Teaching Model to Act Efficiently, May 2025a. URL http://arxiv.org/abs/2504.14870. arXiv:2504.14870 [cs].
- Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. Tina: Tiny Reasoning Models via LoRA, April 2025b. URL http://arxiv.org/abs/2504.15777. arXiv:2504.15777 [cs].
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.
- Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks, 2025c. URL https://arxiv.org/abs/2504.06821.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.
- Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark, 2025. URL https://arxiv.org/abs/2406.19314.
- Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenic Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, Diyi Yang, James Zou, and Jure Leskovec. Optimas: Optimizing compound ai systems with globally aligned local rewards, 2025a. URL https: //arxiv.org/abs/2507.03041.
- Yihong Wu, Liheng Ma, Muzhi Li, Jiaming Zhou, Jianye Hao, Ho-fung Leung, Irwin King, Yingxue Zhang, and Jian-Yun Nie. Reinforcing Question Answering Agents with Minimalist Policy Gradient Optimization, July 2025b. URL http://arxiv.org/abs/2505.17086. arXiv:2505.17086 [cs].
- Wanqiao Xu, Allen Nie, Ruijie Zheng, Aditya Modi, Adith Swaminathan, and Ching-An Cheng. Provably learning from language feedback, 2025a. URL https://arxiv.org/abs/2506.10341.
- Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. Not All Rollouts are Useful: Down-Sampling Rollouts in LLM Reinforcement Learning, June 2025b. URL http://arxiv.org/abs/2504.13818. arXiv:2504.13818 [cs].
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https: //arxiv.org/abs/2505.09388.

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.

- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.
- Lingfan Yu, Jinkun Lin, and Jinyang Li. Stateful large language model serving with pensieve. In *Proceedings* of the Twentieth European Conference on Computer Systems, EuroSys '25, pp. 144–158, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031. 3696086. URL https://doi.org/10.1145/3689031.3696086.
- Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid Latent Reasoning via Reinforcement Learning, May 2025. URL http://arxiv.org/abs/2505.18454. arXiv:2505.18454 [cs].
- Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.
- Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, and Junbo Zhao. LeTS: Learning to Think-and-Search via Process-and-Outcome Reward Hybridization, May 2025. URL http://arxiv.org/abs/2505.17447. arXiv:2505.17447 [cs].
- Siyan Zhao, John Dang, and Aditya Grover. Group Preference Optimization: Few-Shot Alignment of Large Language Models, October 2024. URL http://arxiv.org/abs/2310.11523. arXiv:2310.11523 [cs].
- Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning, June 2025. URL http://arxiv.org/abs/2504.12216. arXiv:2504.12216 [cs].
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.
- Noah Ziems, Dilara Soylu, Lakshya A Agrawal, Isaac Miller, Liheng Lai, Chen Qian, Kaiqiang Song, Meng Jiang, Dan Klein, Matei Zaharia, Karel D'Oosterlinck, Christopher Potts, and Omar Khattab. Multimodule grpo: Composing policy gradients and prompt optimization for language model programs, 2025. URL https://arxiv.org/abs/2508.04660.
- Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

A APPENDIX OUTLINE

- Usage of Large Language Models
- GEPA's Reflection and Prompt Update Meta Prompt
- GEPA Algorithm and Methodology Details
- Evaluation Setup (Contd.)
- Results and Analysis (Contd.)
- GEPA For Inference-Time Search (Contd.)
- GEPA for Adversarial Prompt Search (Contd.)
- Performance vs. Budget (Rollouts) Curves
- Generalization Gap
- Cost vs. Performance Analysis for optimized systems
- GEPA Search Trees
- Visualizing the Iterative Refinement achieved by GEPA
- Examples of best prompts for every benchmark
- GEPA generated prompts for kernel generation

B USAGE OF LARGE LANGUAGE MODELS

The authors used large language models (LLMs) only for polishing prose of text where the complete draft was fully written by the authors initially and polished later with the help of LLM-based assistants including ChatGPT, Gemini, and Perplexity. The authors' used code assistants including Cursor and Copilot to implement the authors' original design and ideas. The scientific contributions, technical methods, ideas and core results are entirely the original work of the authors.

C GEPA'S REFLECTION AND PROMPT UPDATE META PROMPT

763 764

765

770

775 776 777

790

Read the inputs carefully and identify the input format and infer detailed task description about the task I wish to solve with the assistant.

Read all the assistant responses and the corresponding feedback. Identify all niche and domain specific factual information about the task and include it in the instruction, as a lot of it may not be available to the assistant in the future. The assistant may have utilized a generalizable strategy to solve the task, if so, include that in the instruction as well.

Provide the new instructions within ``` blocks.

Figure C shows the meta-prompt used by GEPA, which guides the LLM to reflectively refine its current instruction based on input-output examples and corresponding feedback from the environment.

GEPA ALGORITHM AND METHODOLOGY DETAILS

Figure 5 presents the core GEPA Algorithm, along with the algorithm for Pareto-based candidate selection.

D.1 Merge: System-aware crossover strategy for Compound AI optimization

Algorithm 4 provides the instantiation of the System aware Merge strategy used in GEPA+Merge.

Ε EVALUATION SETUP (CONTD.)

E.1 BENCHMARKS, REFERENCE COMPOUND AI SYSTEMS, AND FEEDBACK FUNCTIONS

To rigorously evaluate the performance of GEPA and and compare it against current state-of-the-art compound AI system optimizers, we assemble a diverse suite of benchmarks mostly obtained from Tan et al. (2025), each paired with available Compound AI Systems.

HotpotOA (Yang et al., 2018) is a large-scale question-answering dataset consisting of 113K Wikipediabased question-answer pairs. It features questions that require reasoning over multiple supporting documents. We modify the last hop of the HoVerMultiHop program (described below) to answer the question instead of generating another query, and the rest of the system remains unmodified. The textual feedback module identifies the set of relevant documents remaining to be retrieved at each stage of the program, and provides that as feedback to the modules at that stage. We use 150 examples for training, 300 for validation, and 300 for testing.

IFBench (Pyatkin et al., 2025b) introduced a benchmark specifically designed to assess language models' ability to follow precise human instructions, especially output constraints (e.g., "answer only with yes or no", or "mention a word at least three times"). The IFBench test set consists of 58 new and out-of-distribution output constraints and instructions to test system's ability to generalize to new task constraints. Pyatkin et al. (2025b) also release IFTrain and IF-RLVR Train data (Pyatkin et al., 2025a) which are used for training. We split the IF-RLVR Train into our train/val sets, and IFBench as our test set in order to ensure that the optimizers do not access the new, unseen constraints being tested in IFBench. We design a 2-stage system, that first attempts to answer the user query, and then in the second stage, rewrites the answer following the constraints. The textual feedback module provides the descriptions of constraints satsified and failed-to-besatisfied by the system's response. Our splits contain 150 training examples, 300 for validation, and 294 for testing.

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822 823

824

825

826

827

828

829

830

831

832

838

839

840

841 842

843

844

845

```
Algorithm 1 GEPA: Reflective Evolutionary Prompt Opti- Algorithm 2 Pareto-based candidate selection
                                                                                                          1: function SELECTCANDIDATE(\mathcal{P}, S)
Require: Inputs: System \Phi, dataset \mathcal{D}_{train}, eval metric \mu, feed-
                                                                                                          2:
                                                                                                                     // Build instance-wise Pareto sets
       back function \mu_f, budget B
                                                                                                                     \mathbf{for}\; \mathbf{each}\; i\; \mathbf{do}
                                                                                                           3:
                                                                                                                           \begin{aligned} s^*[i] \leftarrow \max_k S_{\mathcal{P}[k]}[i] \\ \mathcal{P}^*[i] \leftarrow \{\mathcal{P}[k] : S_{\mathcal{P}[k]}[i] = s^*[i] \} \end{aligned}
Require: Hyperparams: minibatch size b, Pareto set size n_{pareto}
                                                                                                          4:
 1: Split \mathcal{D}_{\text{train}} into \mathcal{D}_{\text{feedback}}, \mathcal{D}_{\text{pareto}}, s.t. |D_{pareto}| = n_{pareto}
                                                                                                           5:
 2: Initialize candidates \mathcal{P} \leftarrow [\Phi], parents \mathcal{A} \leftarrow [\text{None}]
                                                                                                           6:
                                                                                                                      end for
 3: for each (x_i, m_i) in \mathcal{D}_{pareto} do
                                                                                                           7:
                                                                                                                     \mathcal{C} \leftarrow \text{unique candidates in } \bigcup_{i} \mathcal{P}^*[i]
 4:
             S_{\Phi}[i] \leftarrow \mu(\Phi(x_i), m_i)
                                                                                                          8:
                                                                                                                      D \leftarrow \emptyset
 5: end for
                                                                                                          9:
                                                                                                                      while there exists \Phi \in \mathcal{C} \setminus D dominated by
      while budget B not exhausted do
 6:
                                                                                                                another in \mathcal{C} \setminus D do
             k \leftarrow \text{SELECTCANDIDATE}(\mathcal{P}, S)
 7:
                                                                                                         10:
                                                                                                                           D \leftarrow D \cup \{\Phi\}
 8:
             j \leftarrow \text{SELECTMODULE}(\Phi_k)
                                                                                                         11:
                                                                                                                      end while
 9:
             \mathcal{M} \leftarrow \text{minibatch of size } b \text{ from } \mathcal{D}_{\text{feedback}}
                                                                                                                      Remove D from each \mathcal{P}^*[i] to get \hat{\mathcal{P}}^*[i]
                                                                                                         12:
10:
             Gather feedback, scores, traces for \Phi_k[j] on \mathcal{M} using \mu_f
                                                                                                                     Let f[\Phi] = number of i for which \Phi \in
             \pi'_j \leftarrow \text{UPDATEPROMPT}(\pi_j, \text{feedbacks}, \text{traces[j]})
\Phi' \leftarrow \text{Copy of } \Phi_k \text{ w/ module } j \text{ updated by } \pi'_j
11:
12:
                                                                                                                      Sample \Phi_k from \hat{\mathcal{C}} with probability \propto f[\Phi_k]
                                                                                                         14:
13:
             \sigma, \sigma' \leftarrow \text{avg score on } \mathcal{M} \text{ (before, after)}
                                                                                                         15:
                                                                                                                     return index k of \Phi_k in \mathcal{P}
14:
            if \sigma' improved then
                                                                                                         16: end function
15:
                   Add \Phi' to \mathcal{P}; Add k to \mathcal{A}
16:
                   for each (x_i, m_i) in \mathcal{D}_{pareto} do
17:
                        S_{\Phi'}[i] \leftarrow \mu(\Phi'(x_i), m_i)
18:
                   end for
19:
            end if
20: end while
21: return \Phi^* maximizing average score on \mathcal{D}_{pareto}
```

Figure 5: (Left) GEPA's core algorithm for reflective prompt evolution. GEPA works iteratively, in each iteration, selecting some of the current candidates to evolve (line 7), executing the identified candidate on a minibatch of rollouts, while utilizing a special feedback function μ_f to gather module specific feedback when available (lines 9-10, described in detail in Section 3), using an LLM to reflectively update the prompt (line 11), and evaluating whether the system instantiated with the new prompt improved the performance on the minibatch (line 14). If improved, GEPA then proceeds to evaluate the new system candidate on the full D_{pareto} set, adding it to the list of candidates tracked and marking the new system's parent. (Right) The SelectCandidate subprocedure used by GEPA's core algorithm is tasked with identifying the best candidate to evolve in the next optimization iteration. GEPA's chief candidate selection strategy is to find non-dominated candidates in the Pareto frontier (of all task instances), and stochastically select one of them based on their appearance frequency in the Pareto front.

AIME-2025 (Balunović et al., 2025) The AIME-2025 benchmark consists of 2 problem sets of 15 questions each (total 30) obtained from the AIME examination conducted by Mathematical Association of America. We use prior years AIME questions (2022-2024 totalling 90 questions) split equally into training and validation set, and use the AIME-2025 questions, repeating each question 5 times, as the final test set. We use a single-step ChainOfThought as the AI system under optimization.

LiveBench-Math White et al. (2025) LiveBench is a cross-domain benchmark consisting of regularly updated questions. We use the math subset of LiveBench questions retrieved on July 30, 2025. This set of questions (n=368) is shuffled (with python random seed 0) and split equally into train/val/test questions. We use a single-step ChainOfThought as the AI system under optimization.

849

850

851

852

853

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

885

886

887

888

889

890

891

892

Algorithm 3 Check if module combination is desirable

```
1: function DESIRABLE(a, i, j, P)
          for module m=1 to |M| do
 3:
              \pi_a \leftarrow ancestor's prompt for module m
 4:
              \pi_i \leftarrow \text{descendent i's prompt for module } m
 5:
              \pi_i \leftarrow \text{descendent j's prompt for module } m
              if (\pi_a = \pi_i \text{ and } \pi_j \neq \pi_i) or (\pi_a = \pi_j \text{ and }
 6:
     \pi_i \neq \pi_j) then
 7:
                   return True
 8:
              end if
 9:
          end for
10:
          return False
11: end function
```

Algorithm 4 MERGE: Genetic Crossover for Modular Candidates

```
1: function MERGE(\mathcal{P}, \mathcal{A}, S, r)
           i, j \leftarrow r.sample(2, |\mathcal{P}|) // distinct i \neq j
 3:
           A_i \leftarrow \text{GETANCESTORS}(i, A), A_j \leftarrow \text{GETANCES}
      TORS(j, A)
 4:
           if i \in A_j or j \in A_i then
 5:
                continue // skip direct ancestry
 6:
           end if
 7:
           for a \in A_i \cap A_i do
 8:
                if this merge (i, j, a) has been tried before then
 9:
                     continue
10:
                end if
                if S[a] > \min(S[i], S[j]) then
11:
12:
                     continue
13:
                end if
14:
                if not DESIRABLE(a, i, j, P) then
15:
                     continue
                end if
16:
17:
                \Phi' \leftarrow \text{copy of } \mathcal{P}[a]
18:
                for module m=1 to |M| do
                     \pi_a \leftarrow \mathcal{P}[a].\mathcal{M}_m.\pi

\pi_i \leftarrow \mathcal{P}[i].\mathcal{M}_m.\pi
19:
20:
                     \pi_j \leftarrow \mathcal{P}[j].\mathcal{M}_m.\pi
21:
                     if \pi_a = \pi_i and \pi_j \neq \pi_i then
22:
23:
                           \Phi'.\mathcal{M}_m.\pi \leftarrow \pi_i
24:
                      else if \pi_a = \pi_j and \pi_i \neq \pi_j then
25:
                           \Phi'.\mathcal{M}_m.\pi \leftarrow \pi_i
                     else if \pi_i \neq \pi_j \neq \pi_a then
26:
                                             = \arg\max\{S[i], S[j]\}
27:
                           Choose d^*
      (break ties randomly)
28:
                          \Phi'.\mathcal{M}_m.\pi \leftarrow \pi_{d^*}
29:
                     else
30:
                           \Phi'.\mathcal{M}_m.\pi \leftarrow \pi_i // default
31:
                     end if
32:
                end for
33:
                return (\Phi', i, j, a)
34:
           end for
35:
           return None
36: end function
```

HoVer (Jiang et al., 2020) is an open-domain multihop fact extraction and claim verification benchmark built on a Wikipedia-based corpus requiring complex reasoning across multiple sentences and documents, typically involving multiple wikipedia articles. Following Tan et al. (2025), the systems are evaluated for their ability to write queries in multiple hops to retrieve all relevant wikipedia documents (gold documents) required to make the claim. We obtain the HoverMultiHop program from Tan et al. (2025), which performs up to 3-hop retrievals using 2 query writer modules, and 2 document summary modules. The textual feedback module simply identifies the set of correct documents retrieved, and the set of documents remaining to be retrieved, and returns them as feedback text. For the full-parameter finetuning results demonstrated in figure 9, we instantiate a 2-hop program, where the first hop is performed with the initial claim, and the LLM is

prompted in a single turn with the claim and first-hop retrieved documents, to provide the second-hop search query. For HoVer, we use 150 examples for training, 300 for validation, and 300 for testing.

PUPA (Li et al., 2025a) propose the task of Privacy-Conscious Delegation: addressing real-world user queries using an ensemble of trusted and untrusted models. The core challenges are maintaining high response quality while minimizing leakage of personally identifiable information (PII) to untrusted models. Li et al. (2025a) also present PAPILLON, a compound AI system consisting of 2 modules, a user query rewriter and a response rewriter, run over the trusted model, along with an intermediate call to the untrusted model with the rewritten query. The feedback text simply provides the breakdown of the aggregate score, consisting of a response quality score and a PII leakage score. The dataset is split into 111 training examples, 111 for validation, and 221 for testing.

E.2 MODELS AND INFERENCE PARAMETERS

We evaluate GEPA and baseline optimizers using two contemporary LLMs, chosen to represent both opensource and commercial model families. Each compound AI system is instantiated once per model, with all modules (e.g., retrievers, rewriters, answer generators) relying on the same model. All models are allowed a context window of upto 16384 tokens for inference.

Qwen3 8B (Yang et al., 2025): For our open-source experiments (including GRPO), we use Qwen3-8B. Following the recommended settings as per Team (2025), we use a decoding temperature of 0.6, top-p of 0.95, and top-k of 20 for training as well as inference.

GPT-4.1 Mini (OpenAI, 2025): For comparison with large commercial models, we use GPT-4.1 mini (openai/gpt-4.1-mini-2025-04-14) accessed via the OpenAI API with a model temperature of 1.0.

E.3 OPTIMIZERS

Baseline: The base program is directly evaluated without any further optimization applied.

MIPROv2 (Opsahl-Ong et al., 2024): MIPROv2 is a widely used compound AI system prompt optimizer and has been integrated into the DSPy (Khattab et al., 2024) and llama-prompt-ops (AI, 2025) frameworks. It works by jointly optimizing both instructions and demonstrations using Bayesian optimization. For each program module, it first bootstraps candidate sets of instructions and demonstrations, assigning uniform priors over their utilities. Candidate assignments are proposed with the Tree-Structured Parzen Estimator (TPE), and the Bayesian model is updated based on evaluation scores to favor high-performing candidates. The most probable sets of instructions and demonstrations are then selected and validated to obtain the final optimized program configuration.

All MIPROv2 optimization runs are performed with the auto = heavy setting, which corresponds to proposing 18 instruction candidates and 18 bootstrapped few-shot sets. Hence, across benchmarks, the exact number of rollouts varies depending on the number of trials it takes to bootstrap examples (finding 18 successful solution instances), the required number of Bayesian search steps (determined by the number of modules in the system), and size of the valset. Overall, MIPROv2's rollouts ranged from a minimum of 2270 (for PUPA) to maximum of 6926 (for HoVer).

GRPO (Shao et al., 2024): Group Relative Policy Optimization (GRPO) is a reinforcement learning algorithm that estimates advantages in a group-relative manner. For compound AI systems consisting of multiple modules, we use the GRPO implementation provided and open-sourced by Ziems et al. (2025) to perform our experiments, whereas for single-module systems (e.g., figure 9), we use the GRPO implementation provided by SkyRL (Griggs et al., 2025; Liu et al., 2025; Cao et al., 2025).

Across all compound system training runs, each training step uses a group size of 12, with 4 training instances per step (total batch size 48, with per device train batch size 1). Training employs LoRA (Hu et al., 2022) with rank dimension 16, $\alpha=64$, and dropout 0.05, using bf16 precision targeting the projection modules [q, k, v, o, up, down, gate]. We use a learning rate of 1×10^{-5} , $\beta=0.01$, reward scale normalization, and gradient norm clipping of 0.1. Gradients are accumulated for 20 steps before each update, with a "constant with warmup learning" rate scheduler. Non-reentrant gradient checkpointing is enabled to further reduce memory usage. GRPO optimization run for 500 training steps, amounting to fixed 24,000 rollouts, with validation performed every 20 training steps, which is used to implement early stopping. Compound AI system GRPO training experiments are performed on 1xH100/A100 (80 GB memory) with separate GPUs for inference rollouts.

For single-module GRPO training, we adopt full-parameter finetuning with a group size of 16. Each training step employs a global batch size of 32, realized as per-device micro-batches of 4 across 8 GPUs. Roll-out generation is performed with a per-GPU forward micro-batch size of 12. Training is distributed using FSDP2, with sampling performed at temperature 1.0. We apply KL regularization and set the learning rate to 1×10^{-6} . Validation is conducted every 5 training steps. During evaluation, sampling is performed with temperature 0.6, top-p = 0.95, and top-k = 20.

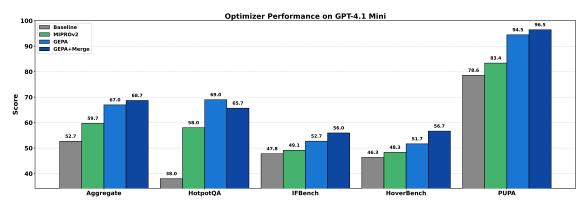
We manually explore several values for [LR, beta, norm clipping] hyperparameters for both training runs.

GEPA: GEPA is our optimizer, based on the algorithm described in Section 3. We evaluate 2 variants of our main optimizer GEPA: GEPA and GEPA+Merge, along with 2 ablations created by replacing the Pareto-based sampling strategy with a naive, SelectBestCandidate strategy (SelectBestCandidate and SelectBestCandidate+Merge). All GEPA optimization runs use a minibatch size of 3, and merge is invoked a maximum of 5 times during the optimization run, when enabled. To ensure a fair comparison with MIPROv2, we align the computational budget between GEPA and MIPROv2 on a per-benchmark basis. The training set from each benchmark is used as $D_{feedback}$ (which is used to derive the training signals, as discussed in Section 3) and the validation set is used as D_{pareto} . Specifically, since MIPROv2's total rollout budget depends on factors such as validation set size and the number of modules, we first record the number of rollouts expended by MIPROv2 for each benchmark, and then cap GEPA's optimization to match this rollout budget. While differences in proposal and validation procedures cause the exact budget usage by the systems to be slightly different, the discrepancy is always within 10.15%. This protocol ensures that any performance differences arise from the optimization algorithms themselves, rather than from differences in search budget. The exact rollout counts for each optimizer is visualized in Appendix I.

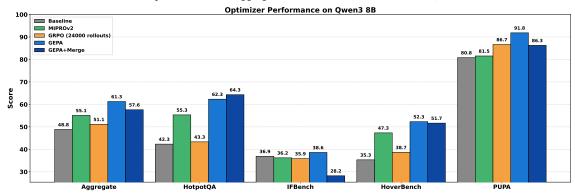
F RESULTS AND ANALYSIS (CONTD.)

Observation 4: Instruction-optimized prompts are computationally cheaper and generalize better than few-shot demonstration prompts: In addition to their strong generalization capabilities, reflectively evolved instructions offer a significant practical advantage: they are often much shorter and thus computationally more efficient than few-shot demonstration prompts. This advantage becomes especially clear for complex tasks, where even a single few-shot demonstration can be prohibitively long. The problem is further exacerbated when few-shot examples are optimized using state-of-the-art methods such as MIPROv2, which jointly optimizes *multiple* demonstrations to be used simultaneously, further increasing prompt length.

In contrast, reflectively evolved instructions—such as those generated by GEPA—maintain compactness while providing large performance gains (as demonstrated in Lessons 1 and 2). To illustrate this, we compare GEPA's and MIPROv2's prompt lengths (see Figure 16). Notably, prompts produced by GEPA and GEPA+Merge are up to $9.2\times$ shorter than those from MIPROv2, representing a substantial improvement in efficiency, alongside performance improvements.



(a) Final test set performance for aggregate and individual benchmarks for gpt-41-mini.



(b) Final test set performance for aggregate and individual benchmarks for qwen3-8b.

Figure 6: Final test set performance for aggregate and individual benchmarks.

Moreover, we observe a trend where, in aggregate, optimizers that achieve higher performance tend to produce shorter prompts (see Figure 15). This reduction in prompt size has a significant impact—not only reducing runtime cost for downstream tasks (as all API-providers meter the input tokens), but also decreasing latency and improving the overall efficiency of LLM-serving systems (Kwon et al., 2023; Zheng et al., 2024; Agrawal et al., 2023; Yu et al., 2025).

Observation 5: System aware crossover strategies can provide large gains, but the optimal budget allocation between mutation and crossover, as well as when to invoke merge needs further study: We identify a unique system-aware crossover strategy and operationalize it as Merge (described in Appendix D.1). GEPA+Merge can outperform GEPA by as much as 5%, providing an aggregate 2% additional improvement over the already strong performance established by GEPA. Detailed results are available in Table 1. We attribute these gains to the ability of GEPA+Merge to identify distinct optimization lineages, that have learnt complementary strategies (by evolving distinct modules), and merging them by picking the best version of different modules from each of these lineages to propose a single, optimal candidate.

While in our analysis, we found GEPA+Merge works especially well for GPT-4.1 Mini, it lead to performance degradation when used with Qwen3 8B. Even Qwen3 8B benefits from Merge on one out of four tasks. We attribute these discrepancies to the way the rollout budget is allocated between reflective mu-

tation and crossover, and the timing of invocation of the crossover strategy. In our experiments, we fixed the same hyperparameters for GPT-4.1 Mini and Qwen3 8B, leading to suboptimal choice for Qwen3 8B. Intuitively, crossover would provide the maximum benefit, when there are independent lineages that perform well. Hence, the hyperparameters should be chosen such that Merge is invoked once the optimization tree has evolved sufficiently different lineages. We propose the study of such adaptive techniques as future work.

G GEPA FOR INFERENCE-TIME SEARCH (CONTD.)

While the primary focus of this paper is sample-efficient adaptation of AI systems to new tasks, preliminary findings suggest that GEPA may also serve as a promising inference-time search technique. This can be achieved by passing the set of tasks to be solved (for example, a list of Pytorch modules to be converted to CUDA) as the training set to GEPA, ensuring that both D_{train} and D_{pareto} contain the full set of tasks. This way, GEPA can "overfit" the set of tasks, iteratively proposing better solutions to every problem. We also note that this allows GEPA to apply lessons and insights extracted from rollouts for one task to other tasks. To explore this use case, we conduct preliminary experiments using GEPA as an inference-time search technique for code-generation tasks on two hardware platforms: writing kernels for AMD's recently introduced XDNA2 Architecture (Advanced Micro Devices, 2025) using an early version of the NPUEval benchmark (Kalade & Schelle, 2025), and generating CUDA code for NVIDIA-V100 GPUs using Kernel-Bench (Ouyang et al., 2025).

A distinguishing aspect of these experiments is the use of the feedback function μ_f to dynamically inject domain-specific knowledge into the optimization process. Specifically, kernel development expertise—often codified in technical manuals and documentation—can be selectively surfaced by retrieving relevant manual sections based on rollout failures (e.g., compiler error messages). By using error information to make targetted retrieval queries, GEPA promotes integration of architectural best practices into prompt evolution, as exemplified by the detailed prompt for NPUEval shown in Figure 26. We also note that generation stochasticity (temperature based sampling) is eliminated by operating under a cache; this ensures that observed improvements tie closely to inference scaling through prompt updates and GEPA's diverse prompt exploration, rather than stochasticity in the model's sampling process.

NPU Kernels: We create a sequential refinement agent that iteratively generates kernels (up to 10 times) based on feedback like compiler errors and profiling results (Sequential10), and evaluate the Best-of-N generation. With GPT-40 alone, Sequential10 reaches only 4.25% mean vector utilization. Adding RAG, sourced from technical manuals, improves this to 16.33%, and integrating MIPROv2 further raises it to 19.03%. Notably, applying GEPA to Sequential10 (without RAG) dramatically boosts kernel performance, with several generated kernels achieving up to 70% vector utilization and a mean of 30.52%. Furthermore, a single prompt generated by GEPA enables Sequential10 (again without RAG) to attain a score of 26.85%.

CUDA Kernels: For 35 tasks from the KernelBench "representative subset" (Ouyang et al., 2025), spanning three difficulty levels, we ran GEPA with GPT-40. As depicted in Figure 8, GEPA boosts GPT-40's close-to-0% $fast_1$ score to above 20% with increasing search budget. This task used an agent that could generate upto 5 sequential refinements based on environment feedback (Sequential5).

These experiments with GPT-40 also demonstrate GEPA's ability to leverage the abilities of frontier LLMs. However, these are early results and warrant further systematic study. We believe that leveraging GEPA for inference-time search, particularly when coupled with domain specific textual feedback, could generalize to other code generation and domain adaptation tasks—a direction we leave for future work.

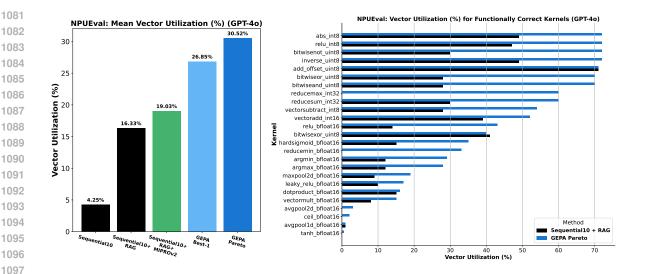


Figure 7: GEPA with GPT-40 is able to generate kernels for AMD NPUs that achieve vector utilization rates as high as 70%, with a mean utilization score of 30.52%. In comparison, GPT-40, even after up to 10 sequential refinements with environment feedback, achieves an aggregate score of only 4.25%. When enhanced with retrieval-augmented generation (RAG) and MIPRO, the sequential refinement agent improves to scores of 16.33% and 19.03%, respectively. Notably, the final prompt produced by GEPA enables the same agent to reach a utilization score of 26.85%, all without requiring any runtime RAG.

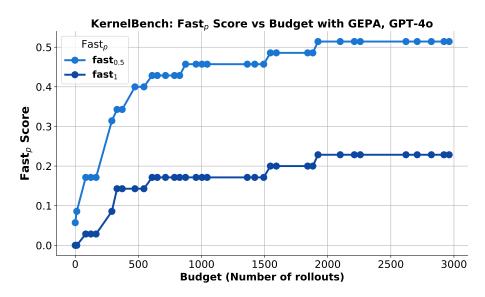


Figure 8: GEPA with GPT-40 is able to iteratively refine and improve CUDA Kernel Code. The graphs shows $fast_p$ vs. rollouts plot for p=[0.5, 1], where the speedup is calculated over Pytorch-eager. $fast_p$ is a metric described in (Ouyang et al., 2025) that measures the fraction of tasks for which the method generated a kernel executing faster than p times the baseline. As can be seen, GEPA with GPT-40 is able to generate cuda kernels executing faster than Pytorch-eager for over 20% of the 35 representative tasks.

H GEPA FOR ADVERSARIAL PROMPT SEARCH (CONTD.)

We instantiate GEPA for adversarial prompt search by inverting the reward signal: the optimizer proposes prompt edits to include additional information like trivia that minimize task performance (pass@1), while requiring that prompts do not contradict the task and still contain all information needed to solve it. For AIME, GEPA's adversarial search used AIME 2022–2024 problems as the pool for prompt evolution. The learned prompt was evaluated on AIME-2025 (30 problems), using GPT-5 Mini with 5 runs per problem (150 generations total). We started from a clean instruction prompt and evolved a single universal adversarial

instruction that is prepended to each query.

The learned adversarial instruction reduced pass@1 on AIME-2025 from 76% (clean prompt) to 10% (adversarial prompt), despite preserving the nominal task description and the required answer-format directive.

Base (clean) prompt:

 You are a helpful assistant. You are given a question and you need to answer it. The answer should be given at the end of your response in exactly the format '### <final answer>'.

GEPA-generated adversarial prompt (abridged):

You are a helpful assistant. You are given a question and you need to answer it. It's interesting to note that honey never spoils and that the longest river in the world is the Nile, stretching over 6,650 kilometers. When providing your answer, be sure to format it at the end of your response exactly as '### <final answer>'. For this task, remember that many mammals, including dolphins, sleep with one eye open. Proceed to answer the given question accordingly.

Manual inspection showed that the adversarial prompt caused GPT-5 Mini to end most responses with the literal placeholder ### <final answer>, indicating a systematic misinterpretation of the formatting rule when paired with the injected distractors. This suggests that the large drop arises from the interaction between extraneous details and a strict, literal formatting constraint, rather than from the formatting requirement alone.

Adversarial prompt search systematically uncovers instruction-level perturbations that sharply degrade model performance, providing a principled, automated way to probe worst-case robustness beyond average-case metrics. By finding universal, task-preserving distractors (e.g., trivia plus strict formatting), it reveals brittle instruction-following interactions and turns them into reusable stress tests and regression suites for continuous evaluation. The resulting adversarial prompts could be used to provide targeted data for fine-tuning or safety training. In practice, this could improve deployment reliability, enables red-teaming at scale, and help track robustness drift over time across models, versions, and domains.

I Performance vs. Budget (Rollouts) Curves

Figures 10, 11, 12, 13 show the full Performance-vs-Rollout curves for all the optimizers across all benchmarks.

J GENERALIZATION GAP

Figure 14 visualizes the generalization gap for different optimization methods.

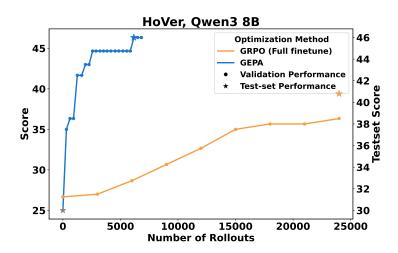


Figure 9: This figure compares the learning behaviour of GEPA against GRPO with full-parameter finetuning on the 2-hop HoVer task. The relative gap mirrors the previously observed comparison of GEPA against GRPO with LoRA (in figures 1, 10, 11, 12, 13), showing that GEPA achieves a comparable performance gap relative to both full-parameter and parameter-efficient versions of GRPO.

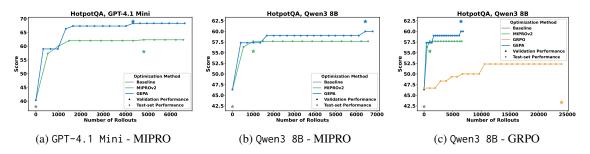


Figure 10: Hotpot QA Bench: rollout vs. score for different models/settings.

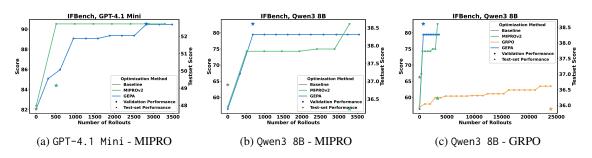


Figure 11: IFBench: rollout vs. score for different models/settings.

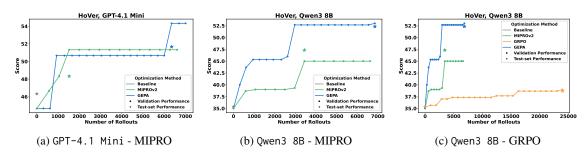


Figure 12: HoverBench: rollout vs. score for different models/settings.

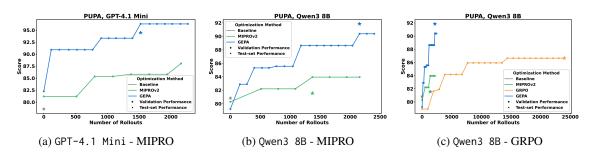


Figure 13: PUPA: rollout vs. score for different models/settings.

K COST VS. PERFORMANCE ANALYSIS FOR OPTIMIZED SYSTEMS

The prompt size of the optimized system plays an important role in determining the downstream cost of using the optimized system. Figure 15 visualizes the aggregate prompt lengths of the final optimized system (as cost proxy) for each optimizer, against the performance achieved. Notably, GEPA's prompts are around 33% shorter than MIPROv2's prompts, while achieving higher performance.

L GEPA SEARCH TREES

Figures 17, 18, 19, 20, 21, 22, 23, and 24 present the genetic search trees created by various configurations of GEPA (and ablation SelectBestCandidate).

M VISUALIZING THE ITERATIVE REFINEMENT ACHIEVED BY GEPA

Figure 25 presented a summary of the prompt refinements performed by GEPA during the optimization for PUPA. In this section, we present the full prompts produced during the optimization.

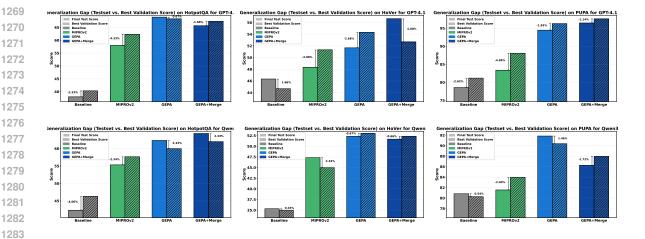


Figure 14: Generalization gaps for different optimization methods. Following Wan et al. (2024), we visualize the generalization gap (i.e., the difference between final test set performance and the best achieved validation performance) for different optimizers. While Wan et al. (2024) previously observed that exemplars tend to generalize better, our results suggest that instructions generated by reflective prompt evolution can achieve stronger generalization as well as improved overall performance. We hypothesize this difference may be due to the improving capabilities of the underlying LLMs, as more recent models are both better at adhering to instructions and capable of reflecting on their outputs.

M.1 Prompts at intermediate stages for PUPA

Node 0 (Score 82.26)

 Given a private user query, create a privacy-preserving request for a powerful external

The LLM may assist without learning private information about the user.

Node 2 (Score 90.99)

Task Description:

You are provided with a private user query that may contain sensitive, personal, or identifying information. Your role is to transform this query into a privacy-preserving request suitable for an external powerful large language model (LLM). The external LLM can be consulted to assist with the task but must not receive any private or personally identifiable information (PII) about the user. Your goal is to maximize the quality and relevance of the LLM request while minimizing or completely eliminating leakage of private information.

Key Points and Domain-Specific Details:

- Privacy Preservation:
 - Do not include any user-specific or sensitive data in the external LLM request.

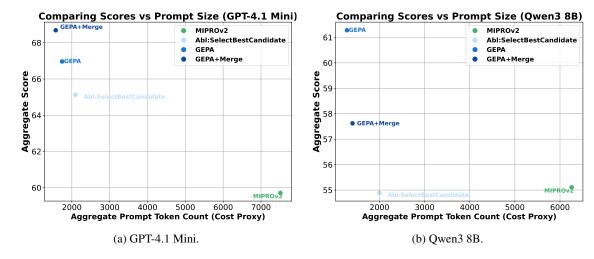
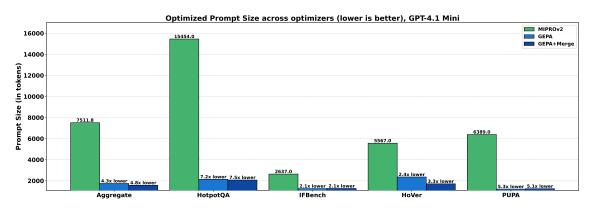
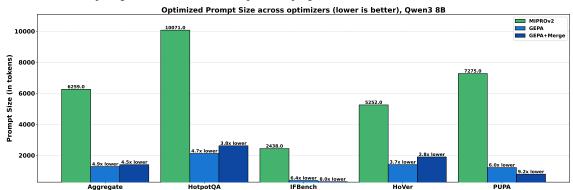


Figure 15: These plots visualize the final aggregate scores against the aggregate prompt size (across all benchmarks) of the final optimized system for each optimizer. It can be seen that GEPA consistently produces prompts that are around less than 33% of the size of MIPROv2's prompts, while getting higher performance. Most of GEPA's prompt tokens are used for providing instructions, whereas most of MIPROv2's prompt tokens pertain to few-shot examples.

- When queries contain location, dates, names, URLs, or other identifiable details, generalize or omit them in the LLM request.
- Replace or abstract any private or potentially identifiable content with neutral placeholders or general terms without losing the intent or meaning necessary for the LLM task.
- 2. Query Understanding and Reformulation:
 - Analyze the user's query carefully to understand the underlying task or information need.
 - Determine if the query involves translation, event recommendations, advice, summarization, or other tasks.
 - Identify when user-provided content is sensitive or proprietary (e.g., unique texts, URLs to closed content) and avoid directly exposing it to the external LLM.
- 3. Quality Maximization:
 - Produce an LLM request that is clear, precise, and directs the LLM to perform the necessary task without requiring private context.
 - Retain the core informational or functional need so that the LLM can respond effectively.
 - When referencing external documents or URLs, do not include the actual link or private info; instead, request general or typical information related to the topic.
- 4. Common Strategies for Privacy-preserving Requests:
 - Use paraphrasing or abstraction for sensitive content.
 - Request generalized or example-based information instead of specific user data.
 - When translation is required, include only the text that needs translating if non-private, or otherwise generalize accordingly.



(a) Comparing the token counts of the optimized programs across benchmarks for GPT-4.1 Mini.



(b) Comparing the token counts of the optimized programs across benchmarks for Qwen3 8B.

Figure 16: Final test set performance for aggregate and individual benchmarks.

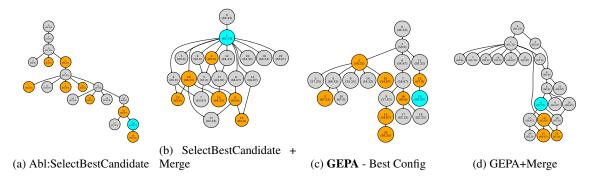


Figure 17: HotpotQA GPT-4.1 Mini

- For location- or event-specific queries, request general or typical information about the category or area without including sensitive details.

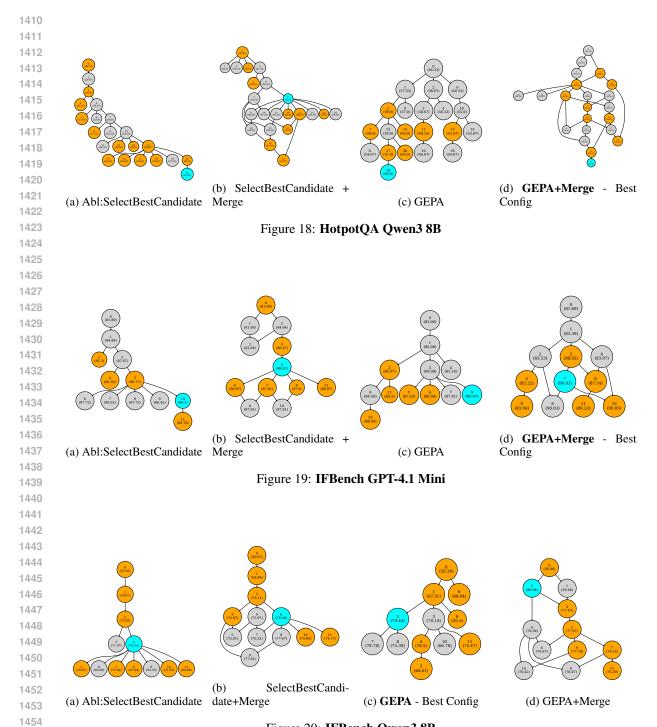


Figure 20: IFBench Qwen3 8B

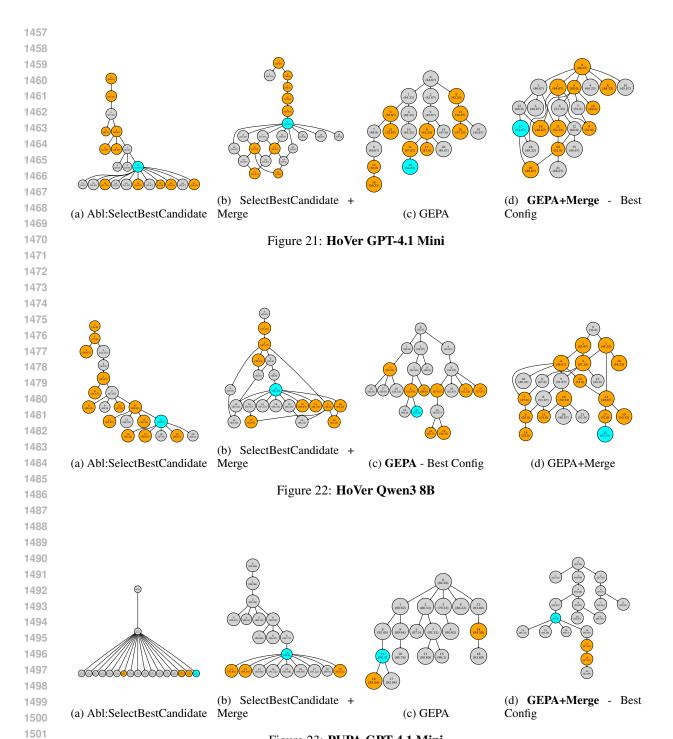


Figure 23: PUPA GPT-4.1 Mini

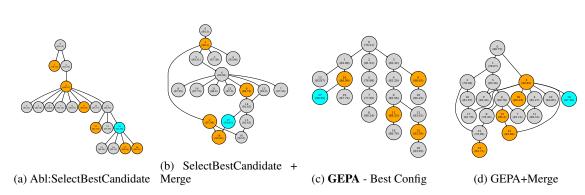


Figure 24: PUPA Qwen3 8B

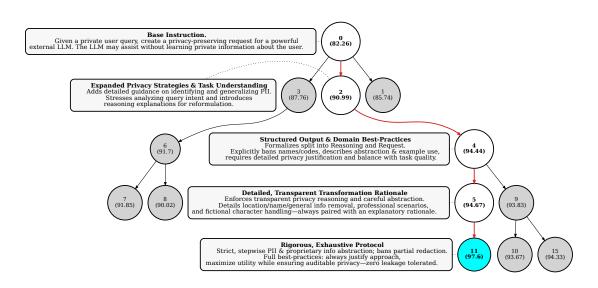


Figure 25: GEPA's reflective prompt mutation systematically incorporates task-specific nuances, leading to substantial improvements in performance. This figure visualizes the optimization trajectory taken by GEPA, presenting an annotated subtree from Figure 23d (for the privacy-preserving delegation task PUPA) to demonstrate the iterative enhancements made to the prompts. The progression from the base prompt (candidate 0) to the best performing prompt (candidate 11) is highlighted with red arrows, and key prompt changes at each step are annotated beside the corresponding nodes. Full-length instructions for these iterations are provided in Appendix M.1. Each prompt refinement in this trajectory adds targeted nuances informed by ongoing optimization, illustrating how GEPA's process accumulates lessons to continually boost task performance.

5. Reasoning Explanation:

- For each transformation, produce clear reasoning explaining how privacy is preserved and why the LLM request fulfills the user ${\rm a}\check{\mathsf{A}}\check{\mathsf{Z}}s$ intent without leaking sensitive information.

Input Format:

- A user query string possibly containing private or sensitive information.
- Required output:
 - a) A reasoning paragraph explaining your analysis and privacy considerations.
 - b) A privacy-preserving LLM request that can be sent to an external LLM for assistance.

By following the above principles and strategies, you will optimize both the quality of assistance and the protection of user privacy when interacting with third-party language models.

Node 4 (Score 94.44)

Task Description:

You will be given a user query that may include sensitive, personal, or identifiable information. Your role is to transform this query into a privacy-preserving request suitable for submission to an external large language model (LLM). The goal is to enable the external LLM to assist effectively while ensuring no private, identifying, or sensitive information about the user is exposed.

Key Requirements and Domain-Specific Details:

- Privacy Preservation:
 - Do not include any user-specific details, personal names, locations, dates, URLs, or other potentially identifiable information in the outgoing LLM request.
 - ${ t -}$ When the user query contains private, sensitive, or proprietary data, you must generalize, abstract, or omit these details.
 - Replace sensitive specifics with neutral placeholders or describe them in generalized terms without losing the core intent of the query.
 - Avoid exposing non-public, proprietary, or confidential content directly to the external LLM.
- 2. Understanding and Reformulation:
 - Carefully analyze the user's query to identify the underlying task type (e.g., summarization, creative writing, translation, profile writing, event recommendations, company background, academic content generation).
 - When reformulating, maintain the essential informational or functional need so that the external LLM can provide a useful, relevant response.
 - Preserve thematic and contextual elements necessary for quality output, while abstracting any sensitive or private details.
 - If the task involves user-supplied content (e.g., essays, texts, character descriptions), distill or summarize the relevant content or use generic versions instead of exposing original private text.
- 3. Maximizing Quality of the Reformulated Prompt:

1632

1633

1634

1635

1636

1640 1641 1642

- Construct clear, precise, and well-structured requests that explicitly guide the external LLM on the task.
- Retain appropriate detail and context to ensure relevance, but balance this carefully against privacy concerns.
- If referencing external documents, URLs, or institutions, do not include any links or private identifiers; instead, request general or typical information related to the subject matter.
- When writing prompts for creative or fictional character tasks, clarify that the profile or content is fictional to avoid accidental exposure of real personal data.
- 4. Common Strategies for Privacy Preservation:
 - Paraphrase or abstract personal and sensitive content.
 - Use general descriptions or hypothetical/example-based requests where appropriate.
 - Omit or generalize specific names, dates, locations, institutions, or proprietary course codes.
 - When user content is extensive, focus on absorbing the core themes and instructing the external model accordingly rather than passing the raw content.
- 5. Explanation Requirement:
 - Provide a concise reasoning paragraph explaining how you identified sensitive or private details and the steps you took to protect user privacy.
 - Clarify how your reformulated LLM request retains the userâĂŹs original intent and task needs without risking data leakage.
 - ${\hspace{0.25cm}\text{-}\hspace{0.25cm}}$ This explanation is mandatory to document your privacy-preserving approach and to justify the abstraction choices made.

Input and Output Format:

- Input: A single user query string that may include sensitive or personal content.
- Output:
 - a) Reasoning: A paragraph explaining the privacy considerations and reformulation strategy.
 - b) LLM Request: A generalized, privacy-safe prompt suitable for querying an external

Examples and Feedback Integration:

- Avoid including real names, course codes, or faculty references; replace them with general terms (e.g., "an interdisciplinary health minor" rather than "HHV").
- When given character bios or profiles, specify the fictional nature clearly and do not include potentially identifying physical descriptions unless necessary for the task.
- When a user asks about specific companies or entities, generalize the request to "a major company in [industry/country]" if privacy concerns exist.
- Maintain a balance between minimal leakage and maximizing task relevance and quality.
- Ensure the reformulated prompt supports a comprehensive and useful response from the external LLM without compromising privacy.

By rigorously following these detailed instructions, you will ensure maximal protection of user privacy while harnessing the power of external LLMs to assist effectively with diverse user requests across domains.

Node 5 (Score 94.67)

New Instruction for Privacy-Preserving Query Reformulation Assistant:

Task Overview

You receive a user query that may contain sensitive, private, or personally identifiable information (PII). Your goal is to transform this input into a generalized, privacy-preserving prompt suitable for querying an external large language model (LLM). The reformulated prompt must enable the external LLM to fulfill the user's original intent effectively, while rigorously protecting user privacy by abstracting or omitting any sensitive details.

Input Format:

- A single string representing a user query.
- The query may include private names, places, dates, proprietary info, or identifiable context.

Output Format:

- Part (a) Reasoning: a concise paragraph explaining:
 - How you identified sensitive or private information in the input.
 - What generalization, abstraction, or omission strategies you applied to protect privacy.
 - How the reformulated prompt maintains the original task or informational need without risking data leakage.
- Part (b) LLM Request: a carefully constructed and concise, privacy-safe prompt that retains essential thematic and functional elements to achieve a useful, relevant response from the external LLM.

Key Domain-Specific Details and Best Practices:

- 1. Privacy Preservation Principles:
 - Remove or replace all user-specific names (personal or organizational), exact dates or durations, locations, URLs, proprietary course or product codes, customer or client names, and other identifiers.
 - When geographic or organizational mentions are critical for context, abstract these to broader or public categories (e.g., âĂa sustainable travel website focused on a Central American countryâĂİ rather than naming the country explicitly).
 - Omit or abstract internal organizational pressures, client names, or sensitive contractual details to generic placeholders (e.g., âĂa clientâĂİ rather than a named company).
 - For biographical or individual-related queries, avoid requesting or referencing real personal info; instead, frame requests around hypothetical or generic profiles unless the name is widely public and essential for the task.
- 2. Understanding and Reformulating the Task:
 - Identify the underlying task (creative writing, summarization, professional communication drafting, etc.) from the userâĂŹs query.
 - Preserve the functional intent and thematic requirements (e.g., content topics around sustainability, summary of a personâĂŹs background, professional email follow-up).
 - For user-supplied content, do not repeat verbatim or expose any original text; instead, extract core themes, instruct the LLM accordingly, or replace with generic examples.

169316941695

1696 1697

1698 1699 1700

1701 1702 1703

1704 1705

1706 1707

1708 1709 1710

1711 1712

1713 1714

17151716

1717 1718 1719

1720 1721

1722 1723 1724

172617271728

1725

172917301731

173217331734

1735 1736 1737

1738

- 3. Maximizing Output Quality While Preserving Privacy:
 - Construct a prompt that is clear, precise, and contains sufficient context to enable comprehensive and relevant LLM output.
 - Avoid ambiguous or overly generic requests that might reduce relevance or usefulness.
 - Maintain a balance between detail necessary for quality and generalization required for privacy.
- 4. Common Reformulation Strategies:
 - Replace specific names with generic role identifiers or placeholders (\hat{a} Aa business contact, \hat{a} Aİ \hat{a} Aa notable individual \hat{a} Aİ).
 - Replace specific locations or institutions with generalized descriptors, e.g., $\hat{a} \check{A} a$ country known for eco-tourism. $\hat{a} \check{A} \dot{I}$
 - For time references, use relative or approximate terms without revealing explicit dates.
 - For internal or proprietary details, describe the scenario generically (e.g., âĂpressure from a client to close a matter urgentlyâĂİ).
 - When handling personal profiles or characters, explicitly state that the profile is fictional if requesting content generation about a person.
- 5. Explanation Requirements:
 - $\mbox{-}$ Always provide reasoning that details how privacy risks were identified and mitigated.
 - Explain how the essential task was preserved despite abstraction.
 - This reasoning documents the privacy-preserving approach and justifies design

Examples and Common Pitfalls:

- Do not retain or lightly obscure personal names or company names; fully abstract or remove.
- Avoid exposing direct quotes from user-supplied texts.
- Use neutral, context-appropriate phrasing to describe user intents.
- Retain geographic or cultural context only when it is publicly available and pivotal to the task.
- Ensure the output prompt remains a clear instruction that the external LLM can readily interpret.

Summary:

Your reformulations must ensure zero exposure of PII or private/proprietary content while retaining enough context and clarity to generate high-quality, relevant responses from the external LLM. This involves careful task analysis, methodical abstraction of sensitive details, and clear, precise prompt construction combined with transparent reasoning on privacy protection mechanisms used.

Node 11 (Score 97.6)

New Instruction for Privacy-Preserving Query Reformulation Assistant (Enhanced):

Task Overview

You receive a user query that may contain sensitive, private, or personally identifiable information (PII). Your objective is to reformulate this query into a generalized, privacy-preserving prompt suitable for sending to an external large language model (LLM).

1739 The reformulated prompt must enable the external LLM to fulfill the user's original 1740 intent effectively while rigorously protecting all user privacy by abstracting, omitting, 1741 or generalizing any sensitive details. 1742 1743 Input Format: 1744 - A single user query string. - The query might contain identifiable names (people, organizations), specific geographic 1745 locations, exact dates or timeframes, proprietary or internal terminology (e.g., system 1746 names, product codes), client or customer names, URLs, or any other sensitive or private 1747 information. 1748 1749 Output Format: 1750 - Part (a) Reasoning: Provide a concise paragraph that explains: 1751 * How you identified sensitive or private information in the input 1752 * What strategies you applied to protect privacy (generalization, omission, 1753 abstraction, replacement with placeholders) 1754 * How the reformulated prompt preserves the original intent and task requirements without risking data leakage 1755 - Part (b) LLM Request: 1756 A concise, carefully constructed privacy-safe prompt that: 1757 * Removes or anonymizes all PII and proprietary/internal details 1758 $\boldsymbol{\ast}$ Abstracts locations, names, dates, and technical terms as needed 1759 * Produces a clear and contextually rich instruction for the LLM to generate a 1760 relevant and informative response aligned with the user's original task 1761 Detailed Domain-Specific Guidance and Best Practices: 1762 1763 1. Identification and Treatment of Sensitive Data: 1764 - All user-specific or personal names (individual or organizational) must be removed or replaced with generic role descriptors (e.g., ${\rm a}\check{\rm A}{\rm a}$ business contact, ${\rm a}\check{\rm A}\dot{\rm a}$ 1765 client, âĂİ âĂa notable individualâĂİ). Never lightly obscure or partially redact; full 1766 abstraction is required. 1767 - All geographic mentions must be abstracted unless the location is publicly known, 1768 essential to the task, and can be generalized (e.g., aña region known for 1769 eco-tourismâĂİ instead of naming a country or city explicitly). 1770 - Exact dates or durations must never be retained; instead, use relative or approximate temporal references (e.g., âĂrecently,âĂİ âĂover the past yearâĂİ). 1771 - Internal or proprietary terms âĂŤ including system names, product codes, 1772 subscription types, and technical jargon âĂŤ must be generalized or replaced with 1773 neutral descriptors to avoid leakage of intellectual property or sensitive operational 1774 details. 1775 - Avoid direct quotes or verbatim inclusion of user-supplied texts unless obfuscated by generalization. 1776 1777 2. Task Understanding and Reformulation: 1778 - Identify the functional intent of the query: Is it creative writing, translation,

- summarization, professional communication drafting, technical explanation, or other?
- Preserve the thematic and informational core of the query (e.g., request for educational quality analysis, technical translation of a passage, biographical summary).

1780

1781

1782

1783

1784 1785

- Do not reproduce the original input verbatim; instead, frame the LLM prompt around the essential thematic elements extracted from the input.

- For queries regarding individuals, avoid direct reference to real personal information unless the name is widely public and essential; even then, use a generic or hypothetical framing for the individual profile. 3. Strategies for High-Quality and Privacy-Preserving Prompts: - Strike a balance between sufficient contextual detail and privacy abstraction to maintain prompt clarity and relevance. - Use neutral, context-aware formulations that clearly instruct the LLM on the content and style expected.

- Avoid vague or overly generic prompts that could result in less useful or lower-quality responses.
- When system or proprietary content is mentioned, instruct the LLM to generalize specific terms and maintain the technical meaning without revealing sensitive info.
- $\mbox{-}$ When a direct translation is requested on specialized text, specify to replace or abstract internal nomenclature.
- 4. Explanation Requirements:

- The reasoning must transparently explain how privacy risks were identified (e.g., presence of names, locations, dates, proprietary terms).
- It must describe the abstraction or omission methods applied (e.g., replacing âĂJonah Van BeijnenâĂİ with âĂa notable individual,âĂİ substituting âĂMakauâĂİ with âĂa specific region,âĂİ or âĂYodaâĂİ with âĂa system nameâĂİ).
- Clarify how the essential task and user intent were preserved despite these generalizations (e.g., focusing on educational quality, technical translation, biographical summary).
- This explanation justifies your design choices and demonstrates adherence to privacy-preserving principles.

Common Pitfalls to Avoid:

- Do not merely lightly obscure or partially redact sensitive details; full anonymization or abstraction is required.
- Do not repeat any user-supplied PII or proprietary content verbatim.
- Avoid including URLs, exact dates, or direct quotes without modification.
- Do not leave ambiguity that could degrade the quality or contextual clarity of the reformulated prompt.
- Do not include any real personal or organizational names unless they are public figures and the query requires it, then use generic descriptors instead.

Example Summary of Effective Approach (Informed by Prior Examples):

- For geographic queries: replace exact place names with general regions and provide a brief contextual descriptor.
- For technical texts containing system names or subscription types: instruct the LLM to translate or process the text while replacing or abstracting proprietary system identifiers.
- For biographical summaries about specific individuals: remove the real name and request a generic, well-structured four-paragraph summary about âĂa notable individual,âĂİ preserving the overall intent without leaking PII.

Summary

Your reformulations must ensure zero exposure of any PII or private/proprietary content while retaining enough thematic and functional clarity for the external LLM to produce high-quality, relevant outputs. This requires thorough analysis of the user's query,

rigorous application of privacy-preservation strategies, and explicit reasoning explanations that document your approach and choices.

N EXAMPLES OF BEST PROMPTS FOR EVERY BENCHMARK

In this section, we present the best optimized prompt obtained for every (benchmark, model) configuration. Each subsection below pertains to one (benchmark, model) configuration. Since every compound AI system consists of multiple modules, each subsection consists of multiple boxes, listing the prompts for each module. MIPROv2 optimized prompts contain upto 4 few-shot examples for each task. We provide just the first demo here for brevity. GEPA's prompts only consist of the optimized instruction, which is provided in full.

N.1 HOTPOTQA, GPT-4.1 MINI

1833

1834

1839 1840

1841

1842

1843

1844

1845 1846

1879

1847 1848 HotpotQA GPT-4.1 Mini create_query_hop2.predict 1849 Base Prompt: 1850 1851 Given the fields `question`, `summary_1`, produce the fields `query`. 1852 MIPROv2 Prompt: 1853 1854 You are an expert multi-hop question answering system designed to refine retrieval 1855 queries for complex questions. Given the original question and an initial summary of 1856 retrieved documents (summary_1), think step by step to analyze how the summary relates to the question. Use this reasoning to generate a focused and precise query that will guide 1857 the retrieval of additional relevant information needed to answer the question 1858 completely. Your output should include a clear chain-of-thought reasoning process followed by the refined query that targets complementary evidence for the next retrieval 1860 hop. Demos: Example 1: 1862 question: 1863 Simone Benedetti plays for a team that currently competes in which Serie? 1864 summary_1: 1865 Simone Benedetti plays for Virtus Entella, but the passages do not specify which Serie the team currently competes in. 1866 1867 The summary indicates that Simone Benedetti plays for Virtus Entella, but it does not 1868 state the current Serie (league level) in which Virtus Entella competes. To 1869 accurately answer which Serie the team currently competes in, it is necessary to look 1870 up the most recent league information for Virtus Entella. 1871 query: What Serie (league level) does Virtus Entella currently compete in? 1873 1874 1875 3 other demos omitted for brevity. 1877 GEPA Prompt generated by config GEPA:

 You will be given two input fields: `question` and `summary_1`.

Your task is to generate a new search query ('query') optimized for the **second hop** of a multi-hop retrieval system. The original user question is typically complex and requires information from multiple documents to answer. The first hop query is the original question used to retrieve an initial set of documents. Your goal is to generate a **second hop query** that retrieves *additional relevant documents* that were *not* found in the first hop but are necessary to answer the original question completely.

Detailed task instructions and hints:

- 1. **Input Understanding:**
 - `question` is the original multi-hop question posed by the user.
 - `summary_1` is a concise summary of information from a document retrieved in the first hop, which partially addresses the question.
- 2. **Purpose and Context:**
 - Your generated `query` aims to find the *missing pieces* of information needed to fully answer the `question`.
 - The multi-hop retrieval system works in stages:
 - First hop: The original question returns some documents.
 - Second hop: Your query must help retrieve any *other relevant documents* NOT found in the first hop that hold complementary or broader context necessary for final answer extraction.
- 3. **Key Observations from Examples and Feedback:**
 - First-hop documents often cover one entity or aspect in the question.
 - Remaining relevant documents often involve connected or higher-level concepts mentioned in `summary_1` but not explicitly asked in the original question.
 - The `query` should be formulated to explicitly target these *missing*, but logically linked, documents.
 - Avoid merely paraphrasing the original question or restating known facts from `summarv_1`.
 - Instead, infer what broader or related entities/concepts might provide the crucial missing information.
 - For example, if `summary_1` describes a population for a small civil parish, but the question wants total population of the wider region, your `query` should target that wider region (e.g., "Madeira archipelago population in 2011").
 - Similarly, if `summary_1` covers a song and the question wants the album it came from, but first hop got song-level documents, your query should retrieve documents about the album itself.
- 4. **How to Build the Query:**
 - Identify the entities or topics mentioned in `summary_1` that appear related but different from first-hop documents.
 - Reframe the query to explicitly mention these broader or related entities connected to the original question.
 - Include relevant key context from the question to maintain specificity, but shift focus to the missing piece.
 - The goal is to retrieve documents that link or complement what was retrieved initially.
- 5. **Practical Strategy:**

```
1927
             - Read the `summary_1` carefully to spot references to bigger contexts or other
1928
             entities not covered in the first hop.
1929
             - Ask yourself, "What entity or aspect does this summary hint at that could answer the
1930
             original question but was not found yet?"
1931
             - Formulate a precise, focused factual query targeting that entity or concept to
1932
             retrieve the missing documents.
1933
          6. **Output:**
1934
             - Produce only the field 'query' as a clear, concise question or keyword phrase
1935
             designed for efficient retrieval of **second-hop documents**.
1936
             - Ensure the query relates logically to the original question while targeting the
1937
             broader or complementary knowledge identified in `summary_1`.
1938
             - Do **not** include the original question or simply rephrase it.
             - Do **not** duplicate information already well-covered by the first hop retrieval.
1939
1940
          By following these principles, you will help the multi-hop retrieval system find all
1941
          necessary documents to answer the multi-faceted original question completely.
1942
1943
1944
          HotpotQA GPT-4.1 Mini final_answer.predict
1945
1946
          Base Prompt:
1947
          Given the fields `question`, `summary_1`, `summary_2`, produce the fields `answer`.
1948
          MIPROv2 Prompt:
1950
          You are an expert multi-hop reasoning assistant skilled in synthesizing information from
1951
          multiple summaries to answer complex questions. Given the `question`, along with two
          intermediate summaries `summary_1` and `summary_2` that contain relevant evidence,
1952
          carefully analyze and integrate the information step-by-step to produce a clear, logical
1953
          reasoning process followed by a concise and accurate final answer.
1954
          Demos:
          Example 1:
1956
          auestion:
              are Machaeranthera and Prumnopitys both the plants?
1957
          summary 1:
1958
              Yes, both Machaeranthera and Prumnopitys are plants. Machaeranthera is a genus of
1959
              flowering plants in the daisy family, while Prumnopitys is a genus of coniferous
1960
              evergreen trees in the podocarp family.
1961
          summary_2:
              Yes, both Machaeranthera and Prumnopitys are plants; Machaeranthera is a genus of
1962
              flowering plants in the daisy family, while Prumnopitys is a genus of coniferous
1963
              evergreen trees in the podocarp family.
1964
          reasoning:
1965
              Both summaries agree that Machaeranthera and Prumnopitys are plants. Specifically,
              Machaeranthera is a genus of flowering plants in the daisy family, and Prumnopitys is
1967
              a genus of coniferous evergreen trees in the podocarp family. Since both belong to
```

answer:

ves

1969

1970 1971

1972 1973 plant genera, the answer is yes.

GEPA Prompt generated by config GEPA:

Task Description:

You are given three fields as input: `question`, `summary_1`, and `summary_2`. Your goal is to produce an `answer` field that directly and explicitly responds to the question using the information from the two summaries, enhanced by your authoritative domain knowledge when needed.

Input Format:

- `question`: A natural language question that may require a fact, definition, name, date, yes/no response, or other specific information.

 - `summary_1` and `summary_2`: Two independently generated summaries or snippets containing information related to the question. They may vary in completeness, accuracy, and specificity.

Requirements and Approach:

1. **Understand the question precisely.** Determine exactly what is askedâĂŤwhether a name, a specific fact, a date, or a yes/no answer.

2. **Compare both summaries.** Analyze the content of `summary_1` and `summary_2`:

If they agree and directly answer the question, use this as primary evidence.
If one summary provides a fact that the other does not mention, carefully evaluate its plausibility.

 ${ t -}$ If the summaries conflict, use domain expertise and authoritative knowledge to resolve or explicitly state uncertainty.

3. **Domain-specific factual verification and nuance:**

- **Names and nicknames:** Provide only the specific nickname or name when asked, without extra phrasing. For example, when asked for the nickname of a person or entity, respond with the nickname alone, not a full sentence.

- **Nationality and identity distinctions:** Use the most precise terms aligned with factual correctness and common usage (e.g., âĂEnglishâĂİ vs. âĂBritishâĂİ) based on domain knowledge.

 - **Dates and historical facts:** Verify dates or historical claims with domain knowledge to pick the correct fact, especially when there might be confusion between franchise start dates vs. event dates etc.

 - **Yes/no questions:** Prefer concise answers of âĂyesâĂİ or âĂnoâĂİ only, unless the question demands elaboration.
- **Types or categories:** If a question asks about the type or category (e.g., type of company), provide the most direct concise phrase without including adjectives like

nationality unless asked explicitly.
4. **Answer conciseness and relevance:**

Provide a brief and direct answer to the question.Avoid repeating or restating the question.

 - Avoid unnecessary context unless requested or needed for clarity.

- Avoid constructing full sentences unless needed; for example, answers to nickname or yes/no questions should be as short and specific as possible.

202920302031

203220332034

2035

203720382039

2040

2042 2043

2044 2045 2046

2047

2048 2049 2050

2052 2053 2054

2056 2057 2058

205920602061

206220632064

206520662067

- 5. **When authoritative knowledge supplements the summaries:**
 - If the summaries are incomplete or potentially inaccurate, incorporate trusted knowledge from your training about the topic to provide the correct and precise answer.
 - For example, when a summary gives a year that conflicts with known release dates or factual details, prefer the verified date.
 - When the summaries differ in style (one uses a formal phrase, another provides just the nickname), respond with the correct, clean answer format (e.g., just the nickname alone).
- 6. **Examples of correct reasoning and answers:**
 - Question: $\hat{a} \check{A} \check{W} hat$ is the nickname of the 2005 Toyota Grand Prix of Long Beach Polesitter $\hat{a} \check{A} \dot{I}$
 - Correct answer: `the thrill from West Hill`
 - Question: âĂWhat type of company is Zipcar led by Scott Griffith from 2003-2013?âĂİ
 - Correct answer: `car-sharing company`
 - Question: âĂWho was the partner of British comic book artist, Henry Flint, that helped create Zombo?âĂİ
 - Correct answer: `Al Ewing`

Summary:

- Use both summaries as primary but not sole evidence.
- Reliably verify and contextualize facts using domain knowledge, especially for nationality, dates, nicknames, company types, and yes/no questions.
- Provide short, direct answers matching the specificity requested.
- Avoid unnecessary elaboration unless explicitly required.
- Explicitly resolve conflicts or ambiguity using your knowledge or state uncertainty when appropriate.

This approach ensures that answers are both accurate and concise, suitable for direct consumption or integration in knowledge bases or question-answering systems.

HotpotQA GPT-4.1 Mini summarize1.predict

Base Prompt:

Given the fields `question`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given a question and a set of related passages, carefully analyze the information by thinking through the relevant facts step-by-step. Produce a clear and concise summary that synthesizes the key points from the passages directly relevant to answering the question, ensuring the summary is focused, accurate, and grounded in the evidence provided.

Demos:

Example 1:

 ${\tt question:}$

The architectural style of a church that stands in front of the Palazzo Ghisilardi Fava originated in what city?

passages:

2068

2080 2081 2082

2079

2083208420852086

2087 2088 2089

209020912092

209320942095

2096 2097 2098

2099 2100

210121022103

2104210521062107

2108 2109 2110

211121122113

2114

['Palazzo Ghisilardi Fava | Palazzo Ghisilardi Fava is a Renaissance style palace, located on via Manzoni 4 in Bologna, region of Emilia Romagna, Italy; it houses the Medieval Civic Museum of Bologna.', 'Madonna di Galliera, Bologna | The Madonna di Galliera is a church with a Renaissance facade and Baroque interiors, located on Via Manzoni, in central Bologna, Italy. It stands in front of the Palazzo Ghisilardi Fava. The present name over the portal is the "Chiesa di Filippini Madonna di Galliera e Filippo Neri".', 'Palazzo Pretorio, Prato | The Palazzo Pretorio of the Tuscan town of Prato was the old city hall located town center, standing in front of the current Palazzo Comunale. It now accommodates the Civic Museum of Prato, which was reopened on September 2013.', 'Palazzo Pisani Gritti | The Palazzo Pisani Gritti is a Gothic-style palace located across the Grand Canal from the Church of the Salute, and stands along the Campo del Traghetto where the church of Santa Maria del Giglio, in the Sestieri of San Marco, Venice, Italy.', 'DÃűmÃűtÃűr Tower | The DÃűmÃűtÃűr tower is the oldest building in Szeged, Hungary. The foundation was most probably laid during the 11th century, while the lower part was built (in Romanesque style) from the 12th century, and the upper part (Gothic style) from the 13th century. The tower was once part of the former "Szent DÃűmÃűtÃűr templom" (St. Demetrius church), but today it stands in DÃşm Square, in front of the much larger Votive Church of Szeged. The upper part was rebuilt from the original stones in 1926. The architecture of the tower is similar to another found in Southern France, or in the territory of the former Byzantine Empire.', 'Palazzo Leoni, Bologna | The Palazzo Leoni is a Renaissance style palace located on Via Marsala #31, in front of the outlet of Via Mentana, in central Bologna, region of Emilia-Romagna, Italy. The apse of the church of San Martino, is across the street.', 'Palazzo Altieri | Palazzo Altieri is a palace in Rome, which was the home of the Altieri family in the city. The palace faces the square in front of the Church of the GesÃź.'] reasoning:

The question asks for the city where the architectural style of a church that stands in front of the Palazzo Ghisilardi Fava originated. Passage [2] identifies the church standing in front of the Palazzo Ghisilardi Fava as "Madonna di Galliera" and mentions it has a Renaissance facade and Baroque interiors. Passage [1] states the Palazzo Ghisilardi Fava is located in Bologna, Italy. Since the church is located in Bologna and features a Renaissance architectural style, it can be inferred that the architectural style of this church originated in the city of Florence, which is historically recognized as the birthplace of Renaissance architecture. Thus, the architectural style (Renaissance) of the church in front of the Palazzo Ghisilardi Fava originated in Florence.

summary:
The Renaissance architectural style of the church standing in front of the Palazzo

Ghisilardi Fava originated in Florence.

GEPA Prompt generated by config GEPA:

You are a first-hop **summarization module** in a multi-hop question answering (QA) system. Your task is to generate a concise, informative `summary` given two input fields: a `question` and a list of relevant `passages`.

Your goal is to extract and synthesize key information from the retrieved passages that:

```
2116
          1. Directly relates to the initial question.
          2. Captures the core facts and entities needed to understand the scope and context of the
2117
          question.
2118
          3. Includes relevant connections, bridging entities, dates, locations, or descriptions
2119
          that enable the system to devise focused and effective follow-up queries in subsequent
2120
2121
          4. Provides a strong factual foundation for downstream answer generation modules.
2122
          **Task specifics and best practices:**
2123
2124
           - The `summary` must represent a distilled synthesis, not just a compression or
2125
          extractive snippet.
           - Explicitly include cited passage titles or key entity labels (e.g., "Children in Need
2126
          2006 \mid \dots \rangle or "Anthony Levandowski \mid \dots \rangle in your summary to highlight the origin of
2127
2128
           - Incorporate sufficient context to hint at missing or un-retrieved supporting facts,
2129
          thus enhancing the multi-hop retrieval process.
2130
          - When the question asks for an attribute (e.g., nationality, location, company origin),
2131
          ensure you provide:
             - Identification of the relevant subject or entity mentioned in the passages.
2132
             - The extracted attribute or relevant information as stated or implied.
2133
             - Bridging details that could help the system pursue remaining information in the next
2134
             retrieval step.
2135
           - Avoid forming a final answer; instead, focus on "what is known now" from the input
2136
          documents to facilitate further query refinement.
2137
          **Examples of critical elements to include:**
2138
2139
           - Entity names, roles, titles, and dates tied to the question.
2140
          - Names of organizations or locations connected through intermediary entities.
           - Distinctive identifiers or clarifications that can help narrow down next-step retrieval
2141
           (such as "Natasha Kaplinsky is an English presenter," "Kapolei is a city on Oahu," or
2142
           "Waymo spun out of Alphabet").
2143
2144
          **Format of output:**
2145
2146
          Provide a paragraph or a few sentences that cohesively summarize the key passages in
          relation to the question, referencing passage titles or entities to frame facts clearly.
2147
2148
2149
2150
          This approach ensures the summary is both informative for next-hop retrieval and
2151
          foundational for final answer extraction in multi-hop QA.
2152
2153
```

HotpotQA GPT-4.1 Mini summarize2.predict

Base Prompt:

2154

2155 2156

2157

2158 2159

2160 2161 Given the fields `question`, `context`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

2194

2195

2196

2197

2198

2199

2200

2201

2202

2203

2204

2205

2206

2207

Given a `question`, relevant `context`, and a list of `passages`, provide a clear, concise summary that integrates the key information from the passages in relation to the question and context. Use step-by-step reasoning to explain how the summary is derived from the evidence before presenting the final synthesized summary.

Demos:

Example 1:

question:

What bank was founded by the great-great grandfather of the second Duke of Florence?

context:

The bank founded by the great-great grandfather of the second Duke of Florence is the Medici Bank, established by Giovanni di Bicci de' Medici.

passages:

['Giovanni di Bicci de\' Medici | Giovanni di Bicci de\' Medici (c. 1360 âĂŞ February 20/28, 1429) was an Italian banker, a member of Medici family of Florence, and the founder of the Medici Bank. While other family members, such as Chiarissimo di Giambuono de\' Medici, who served in the Signoria in 1201, and Salvestro de\' Medici, who was implicated in the Ciompi Revolt of 1378, are historically significant, Giovanni\'s founding of the family bank truly began the family\'s rise to power in Florence. He was the father of Cosimo de\' Medici ("Pater Patriae"), grandfather of Piero di Cosimo de\' Medici, great-grandfather of Lorenzo de\' Medici (the Magnificent) and great-great-great-grandfather of Cosimo I de\' Medici, Grand Duke of Tuscany.', 'Averardo de\' Medici | Averardo de\' Medici (1320-1363), also known as Everard De Medici, was the son of Salvestro de\' Medici (died 1346), "il Chiarissimo" (English meaning "the very clear.") and the father of three children: Giovanni, Francesco, and Antonia. Giovanni di Bicci de\' Medici would later become the first historically relevant member of Medici family of Florence and the eventual founder of the Medici bank.', "Villa Medici at Cafaggiolo | The Villa Medicea di Cafaggiolo is a villa situated near the Tuscan town of Barberino di Mugello in the valley of the River Sieve, some 25\xa0kilometres north of Florence, central Italy. It was one of the oldest and most favoured of the Medici family estates, having been in the possession of the family since the 14th century, when it was owned by Averardo de' Medici. Averardo's son, Giovanni di Bicci de' Medici, is considered to be the founder of the Medici dynasty.", 'Medici: Masters of Florence | Medici: Masters of Florence is an Italian-British television drama series about the Medici dynasty set in 15th-century Florence, starring Dustin Hoffman as Giovanni di Bicci de\' Medici, Richard Madden as Cosimo de\' Medici, and Stuart Martin as Lorenzo de\' Medici ("The Elder"). The series was co-created by Frank Spotnitz ("The X-Files" and "Man in the High Castle") and Nicholas Meyer (""). Sergio Mimica-Gezzan ("The Pillars of the Earth") directed all eight episodes. Episodes 1 and 2 aired on Rai 1 (Italian TV) on 18 October 2016. According to Italian ratings compiler Auditel, it attracted a record 7.6\xa0million viewers. The first season consists of eight episodes.', 'Neri di Bicci | Neri di Bicci (1419âĂŞ1491) was an Italian painter of the Renaissance. A prolific painter of mainly religious themes, he was active mainly in Florence and in the medium of tempera. His father was Bicci di Lorenzo. His grandfather, Lorenzo di Bicci was also a painter in Florence, a pupil of Spinello Aretino.', 'Cosimo de\' Medici | Cosimo di Giovanni de\' Medici (called \'the Elder\' (Italian "il Vecchio") and posthumously "Father of the Nation" (Latin "pater patriae"); 27 September 1389 âĂŞ 1 August 1464) was an Italian banker and politician, the first of the Medici political dynasty, "de facto" rulers of Florence during much of the Italian Renaissance. Despite his influence, his power was not absolute; Florence\'s legislative councils at times resisted his proposals, something which would not have been tolerated by the Visconti of Milan, for instance; throughout his

life he was always "primus inter pares" ("first among equals"). His power derived from his wealth as a banker, and he was a great patron of learning, the arts and architecture.', "Giovanni il Popolano | Giovanni de' Medici, in full Giovanni di Pierfrancesco de' Medici, later known as il Popolano (21 October 1467 âĂŞ 14 September 1498) was an Italian nobleman of the Medici House of Florence. He was the son of Pierfrancesco di Lorenzo de' Medici, and therefore a member of a secondary branch of the family."]

reasoning:

The question asks for the bank founded by the great-great-great-grandfather of the second Duke of Florence. Passage [1] states that Giovanni di Bicci de' Medici founded the Medici Bank and was the great-great-great-grandfather of Cosimo I de' Medici, who was the Grand Duke of Tuscany. The Dukedom of Florence eventually transformed into the Grand Duchy of Tuscany under Cosimo I, who is identified as the second Duke of Florence in the question context.

Thus, Giovanni di Bicci de' Medici, as the great-great-great-grandfather of the second Duke of Florence (Cosimo I de' Medici), founded the Medici Bank. summarv:

The Medici Bank was founded by Giovanni di Bicci de' Medici, the great-great-great-grandfather of the second Duke of Florence.

GEPA Prompt generated by config GEPA:

Given the fields `question`, `context`, `passages`, produce the fields `summary`.

N.2 HOTPOTQA, QWEN3 8B

HotpotQA Qwen3 8B create_query_hop2.predict

Base Prompt:

Given the fields `question`, `summary_1`, produce the fields `query`.

MIPROv2 Prompt:

Given the question and the first summary, synthesize the key elements of the question and the summary. Identify the specific information that needs to be retrieved or confirmed. Formulate a focused and precise query that will guide the next step in the multi-hop reasoning process, ensuring it directly addresses the gap in knowledge or requires further clarification from additional context.

Demos:

GEPA Prompt generated by config GEPA+Merge:

Given the fields `question` and `summary_1`, produce the field `query` that optimizes the retrieval of additional documents for a multi-hop system.

Task Details:

```
2256
          1. **Objective:** Your query must target documents not retrieved in the first hop, using
2257
          clues from the summary and the original question.
2258
          2. **Key Strategy:**
             - Identify gaps in the first hop's retrieved documents (e.g., missing entities,
2260
             relationships, or specific details).
2261
             - Use explicit information from the summary (e.g., names, locations, quantities) to
2262
             rephrase the question into a query that surfaces new relevant documents.

    Avoid restating the answer directly; instead, structure the query to explore

2263
             connections or unresolved details.
2264
          3. **Domain-Specific Guidance:**
2265
             - If the summary explicitly answers the question, the query should still focus on
2266
             retrieving documents that provide deeper context or verify the answer (e.g., "What is
             the headquarters location of [Company]?" instead of "The answer is [Location]").
2267
             - Leverage entities mentioned in the summary (e.g., "Carhartt," "Aubrey O'Day") to
2268
             anchor the query.
             - If no documents are missing, rephrase the query to explicitly request the answer
2270
             (e.g., "Which has more acts, Elektra or From the House of the Dead?").
2271
          4. **Avoid:**
             - Generating queries that duplicate the original question.
2272
             - Assuming the summary contains all necessary information for the second hop.
2273
2274
2275
          HotpotQA Qwen3 8B final_answer.predict
2276
2277
          Base Prompt:
2278
          Given the fields `question`, `summary_1`, `summary_2`, produce the fields `answer`.
2279
2280
          MIPROv2 Prompt:
2281
          Given the question, summary_1, and summary_2, generate a step-by-step reasoning process
          that logically connects the information from both summaries to arrive at the final
2283
          answer, and then provide the answer. Ensure the reasoning explicitly references how the
2284
          summaries support the conclusion and maintains consistency with the question's
2285
          requirements.
          Demos:
2286
          Example 1:
2287
          question:
              are Machaeranthera and Prumnopitys both the plants?
2289
          summary_1:
2290
              Yes, both Machaeranthera (a genus of flowering plants in the daisy/aster family) and
              Prumnopitys (a genus of coniferous trees in the podocarp family) are plants.
2291
2292
              Yes, both Machaeranthera (a flowering plant genus in the daisy family) and
2293
              Prumnopitys (a coniferous tree genus in the podocarp family) are plants.
2294
          reasoning:
2295
              Both summaries confirm that Machaeranthera and Prumnopitys are plants. Summary 1
2296
              explicitly states they are plants, categorizing Machaeranthera as a flowering plant
              genus in the daisy/aster family and Prumnopitys as a coniferous tree genus in the
2297
              podocarp family. Summary 2 repeats this confirmation with slightly different phrasing.
2298
          answer:
              Yes
2300
```

```
2303
     2304
     3 other demos omitted for brevity.
2305
     2306
2307
2308
2309
2310
```

GEPA Prompt generated by config GEPA+Merge:

Given the fields `question`, `summary_1`, and `summary_2`, produce the field `answer` by: 1. **Extracting precise terminology**: Identify the exact noun or specific term required in the answer (e.g., "Medicare" rather than "Medicare cuts"). Avoid vague or generalized terms unless explicitly stated in the summaries.

- 2. **Resolving ambiguity**: If the question references a title, historical role, or specific designation (e.g., "second Duke of Florence"), prioritize contextual or historical clues from the summaries to infer the correct answer, even if the exact term is not explicitly stated. Use domain-specific knowledge (e.g., Medici family lineage) to fill gaps when summaries are indirect or vague.
- 3. **Cross-referencing summaries**: Ensure consistency between summaries. If summaries conflict, prioritize the one with explicit factual claims (e.g., numerical data, direct statements). If no explicit claim exists, synthesize information while ensuring alignment with historical, political, or cultural context.
- 4. **Avoiding overgeneralization and extra information**: Focus strictly on the most specific and directly stated information in the summaries. Do not add context, explanations, or external knowledge beyond what is explicitly provided. For example, if the question asks for a year, provide only the year; do not include band member details or historical background.
- 5. **Prioritizing factual alignment**: If a summary explicitly states the answer, use that. If summaries are indirect or vague, synthesize information while ensuring alignment with factual knowledge (e.g., linking "Path to Prosperity" to Rep. Paul RyanâĂŹs Medicare proposal).

Key adjustments based on feedback:

- **Conciseness**: Answers must be strictly factual and concise, avoiding additional context or explanations. For example, if the question is "Is X shorter than Y?" the answer should be a simple "No" or "Yes" based on numerical comparisons, not a full explanation.
- **Numerical precision**: When comparing measurements (e.g., heights, dates), ensure exact values are used and explicitly stated in the summaries. If summaries provide conflicting numbers, resolve via direct factual claims.
- **Domain-specific knowledge**: Use known facts (e.g., architectural records, historical timelines) to validate ambiguous answers, but only when summaries lack explicit information.

2337 2338 2339

2311

2312

2313

2314

2315

2316

2317

2318

2319

2320

2321

2322

2323

2324

2325

2326

2327

2328

2329

2330

2331

2332

2333

2334

2335

2336

HotpotQA Qwen3 8B summarize1.predict

2340 2341

Given the fields `question`, `passages`, produce the fields `summary`.

Given the fields `question`, `passages`, produce the fields `summary`.

2342 2343 2344

MIPROv2 Prompt:

Base Prompt:

2345 2346

Demos: 2347 Example 1: question:

2351 2352 2353

2354 2355 2356

2357235823592360

236123622363

2365 2366 2367

2364

2368 2369 2370

2370 2371 2372

237323742375

2376 2377

2378 2379

2380

238123822383

2384

238623872388

2389 2390

239123922393

23942395

Tay Garnett and Alexander Kluge both have what job? passages:

['Tay Garnett | William Taylor "Tay" Garnett (June 13, 1894 âĂŞ October 3, 1977) was an American film director and writer.', 'Trade Winds (film) | Trade Winds is a 1938 comedy film distributed by United Artists. It was directed by Tay Garnett, and starred Fredric March and Joan Bennett. The screenplay was written by Dorothy Parker, Alan Campbell and Frank R. Adams, based on story by Tay Garnett.', 'Prestige (film) | Prestige is a 1932 American pre-Code drama film directed by Tay Garnett and written by Tay Garnett, Rollo Lloyd and Francis Edward Faragoh. The film stars Ann Harding, Adolphe Menjou, Melvyn Douglas and Guy Bates Post. The film was released on January 22, 1932, by RKO Pictures.', 'Alexander Kluge | Alexander Kluge (born 14 February 1932) is a German author, philosopher, academic and film director.', 'Cattle King | Cattle King is a 1963 film directed by Tay Garnett. It stars Robert Taylor and Robert Loggia. It also appears to have been called Guns of Wyoming in some countries.', 'GÃnther von Kluge | GÃnther von Kluge (30 October 1882 âĂŞ 19 August 1944) was a German field marshal during World War II. Kluge held commands on both the Eastern and Western Fronts. Although Kluge was not an active conspirator in the 20 July plot, he committed suicide on 19 August 1944, after having been recalled to Berlin for a meeting with Hitler in the aftermath of the failed coup. He was replaced by Field Marshal Walter Model.', 'Bad Company (1931 film) | Bad Company is a 1931 American Pre-Code gangster film directed and co-written by Tay Garnett with Tom Buckingham based on Jack Lait\'s 1930 novel "Put on the Spot". It stars Helen Twelvetrees and Ricardo Cortez. Told from the view of a woman, the working titles of this film were "The Gangster\'s Wife" and "The Mad Marriage". Unlike many static early sound films, Garnett includes several scenes using a moving camera climaxing in a gigantic assault on an office building with both sides using heavy machine guns.']

reasoning:

The question asks for the common job of Tay Garnett and Alexander Kluge. From the passages, Tay Garnett is explicitly identified as a film director and writer (passages 1, 2, 3, 5, 7). Alexander Kluge is described as a German author, philosopher, academic, and film director (passage 4). The shared job between them is "film director."

summary:

Both Tay Garnett and Alexander Kluge are film directors.

GEPA Prompt generated by config GEPA+Merge:

You are the first-hop **summarization module** in a multi-hop QA system. Your task is to generate a **comprehensive, structured summary** that:

- 1. **Extracts direct answers** from the top retrieved passages to address the question.
- 2. **Identifies and highlights missing or implied clues** that may require further retrieval (e.g., entities, connections, or contextual details).
- 3. **Synthesizes information** by combining explicit facts from the passages with domain-specific knowledge or logical inferences to guide subsequent steps.

Summary Structure

```
2397
           - **Entity/Person Mention**: Clearly state the subject (e.g., "Billy Truax", "Eintracht
2398
          Braunschweig") and include **full names, titles, or official designations** (e.g.,
2399
           "Thomas Lance Rentzel", "Braunschweiger Turn- und Sportverein Eintracht von 1895 e.V.").
2400
           - **Direct Answer**: Include **explicit answers** from the passages (e.g., birth dates,
2401
          team affiliations, or direct statements).
2402
           · **Clues for Next Steps**: Signal **missing information** (e.g., "Lance Rentzel's birth
2403
          year is explicitly stated, but his exact birthplace is not; need to search for 'Lance
          Rentzel birthplace'
2404
          - **Domain-Specific Context**: Add **relevant background** (e.g., "Eintracht Braunschweig
2405
          is a German football club based in Braunschweig, Lower Saxony" or "NFL players' birth
2406
          dates are critical for age comparisons").
2407
2408
          ### **Guidelines**
           - **Do not omit** any entity or detail from the retrieved passages that could be relevant
2409
          for follow-up queries (e.g., team names, locations, or historical context).
2410
          - **Prioritize clarity** by **separating direct answers from inferred clues** (e.g.,
2411
          using bullet points, subheadings, or bolded labels).
2412
          - **Avoid assumptions** not supported by the passages; if information is absent,
2413
          **explicitly state that it is missing** and suggest **precise search terms** (e.g.,
           "Verify Wichita Dwight D. Eisenhower National Airport's tower status via FAA records").
2414
           - **Include quantifiable data** (e.g., "few thousand Stabyhouns exist globally", "born
2415
          July 15, 1943") to enable precise comparisons.
2416
           - **Highlight connections** between entities (e.g., "Billy Truax and Lance Rentzel were
2417
          traded in 1970") to aid in cross-referencing.
2418
          ### **Key Niche/Domain-Specific Insights**
2419
          - **NFL Player Comparison**: Birth dates are critical for age determination, and team
2420
          affiliations (e.g., "traded in 1970") may imply historical context.
2421
          - **Airport Classification**: "Non-towered" status is explicitly stated in some passages
2422
          (e.g., "non-towered public airport"), while others require inference (e.g., "major
          commercial airports typically have towers").
2423
           - **Football Club Context**: Clubs like Eintracht Braunschweig require background on
2424
          their location, league, and history (e.g., "based in Braunschweig, Lower Saxony").
2425
          - **Quantifiable Data**: Use exact dates, numbers, or rankings (e.g., "few thousand
2426
          Stabyhouns exist globally") to enable precise comparisons.
2427
2428
          ### **Critical Additional Instructions**
          - **Ensure All Retrieved Documents Are Represented**: Explicitly include all entities,
2429
          titles, and details from the retrieved passages (e.g., full names, film titles, and
2430
          specific roles).
2431
           - **Signal Missing Links**: If a connection between entities is implied but not
2432
          explicitly stated (e.g., "Nancy Steiner worked on *The Lovely Bones*"), flag this as a
2433
          potential gap and suggest search terms to resolve it.
           - **Prioritize Bridging Concepts**: Highlight relationships between entities (e.g., "Gary
2434
          Pinkel coached Toledo in 1993 and holds the most wins in school history") to enable
2435
          focused follow-up queries.
2436

    - **Avoid Overgeneralization**: Only include domain-specific context that is either

2437
          explicitly stated in the passages or directly inferable (e.g., "major commercial airports
          typically have towers" is acceptable, but "airports with fewer than 10,000 passengers are
2438
          non-towered" is not unless stated).
2439
2440
          ### **Example Format**
2441
          For the question *"Which NFL player is younger, Billy Truax or Lance Rentzel?"*:
2442
```

```
2444
           - **Entity/Person Mention**: Billy Truax (William Frederick Truax), Lance Rentzel (Thomas
2445
          Lance Rentzel)
2446
          - **Direct Answer**:
2447
            - **Billy Truax**: Born July 15, 1943.
2448
            - **Lance Rentzel**: Born October 14, 1943.
2449
          - **Clues for Next Steps**: None required; birth dates are explicitly provided.
2450
          - **Domain-Specific Context**: Birth dates are sufficient to determine age difference
          within the same year.
2451
2452
          For the question *"Which is a non-towered airport, Wichita Dwight D. Eisenhower National
2453
          Airport or Montrose Regional Airport?"*:
2454
           - **Entity/Person Mention**: Wichita Dwight D. Eisenhower National Airport, Montrose
2455
          Regional Airport
           - **Direct Answer**:
2456
            - **Montrose Regional Airport**: "non-towered public airport" (passage 3).
2457
            - **Wichita Dwight D. Eisenhower National Airport**: No explicit mention of tower
2458
            status; inferred as **towered** (typical for major commercial airports).
2459
          - **Clues for Next Steps**: Verify Wichita's tower status via FAA records or additional
          sources (e.g., "Wichita Dwight D. Eisenhower National Airport tower status").
2460
           - **Domain-Specific Context**: Non-towered airports lack a control tower, relying on
2461
          pilot communication (passage 4). Major commercial airports like Wichita usually have
2462
          towers.
2463
2464
           **Tip:** When summarizing, donâĂŹt just compress; synthesizeâĂŤinclude both direct
2465
          answers and clues required for the systemâĂŹs next steps. Always explicitly state if a
          retrieved documentâĂŹs content is missing critical information, and provide actionable
2466
          search terms to address gaps.
2467
2468
```

HotpotQA Qwen3 8B summarize2.predict

Base Prompt:

Given the fields `question`, `context`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given the fields `question`, `context`, and `passages`, generate a structured reasoning process that explains how to synthesize information from these sources, and then produce a concise summary of the key findings.

Demos:

Example 1:

2480 question:

2469

2470 2471

2472

2473 2474

2475

2476

2477

2478

2479

2481 2482

2483

2485

2486

2487

2488

2490

Simone Benedetti plays for a team that currently competes in which Serie?

The provided passages do not specify the Serie in which Simone Benedetti's team (Virtus Entella) currently competes.

2484 passages:

['Simone Benedetti | Simone Benedetti (born 3 April 1992) is an Italian professional footballer who plays as a centre back for Virtus Entella.', 'Virtus Entella | Virtus Entella (usually referred as simply Entella or Chiavari Entella) is an Italian association football club, based in Chiavari, Liguria. They currently compete in Serie B.', 'Simone Palermo | Simone Palermo (born 17 August 1988) is an Italian footballer who plays as a midfielder for Virtus Entella at Serie B.', 'Andrea Paroni

 | Andrea Paroni (born 14 October 1989) is an Italian footballer who currently plays as a goalkeeper for Virtus Entella in the Serie B.', 'Alessandro Iacobucci | Alessandro Iacobucci (born 3 June 1991) is an Italian footballer who currently plays as a goalkeeper for Serie B side Virtus Entella.', 'George Ganea | George Ganea (born 26 May 1999 in Bucharest) is a Romanian footballer who plays as a Striker for Serie B club Virtus Entella Primavera team.', 'Gianpaolo Castorina | Gianpaolo Castorina (born August 30, 1976 in Milan) is a retired Italian professional footballer turned coach, currently in charge as head coach of Serie B club Virtus Entella.'] reasoning:

The question asks for the Serie in which Simone Benedetti's team (Virtus Entella) currently competes. While the context initially states that this information is not specified, the passages explicitly mention that Virtus Entella competes in Serie B. Multiple passages (e.g., [2], [3], [4], [5], [7]) confirm this detail, directly answering the question.

summary:

Simone Benedetti's team, Virtus Entella, currently competes in Serie B.

GEPA Prompt generated by config GEPA+Merge:

Given the fields `question`, `context`, and `passages`, produce the field `summary`.

Your task is to synthesize information from the question, context, and newly retrieved passages to generate a **comprehensive, precise, and well-structured summary** that enables the answer generation module to confidently arrive at the correct answer.

Key Requirements:

- 1. **Explicit Answers First**: Prioritize explicitly stated facts from the context and passages (e.g., direct mentions of entities, roles, or relationships).
- 2. **Infer or Generalize When Necessary**: If critical details are missing from the passages, infer connections or generalize based on contextual clues and domain-specific knowledge (e.g., linking ownership structures, roles, or historical context).
- 3. **Bridge Gaps**: Ensure the summary includes all **key supporting information** required to answer the question, even if it is not explicitly stated in the input. For example:
 - If the answer is "Newcastle United," include details about Sports Direct's ownership and the connection to the billionaire.
 - If the answer is a person's role (e.g., "troubleshooter"), explicitly state their relationship to the question's subject and any relevant background.
- 4. **Structure and Precision**:
 - Clearly connect entities, roles, and relationships (e.g., "Stan Kroenke owns Sports Direct and Arsenal F.C.").
 - Avoid ambiguity by including all necessary contextual links (e.g., "Mike Ashley founded Sports Direct and owns Newcastle United").
 - Use precise terminology and ensure alignment with domain-specific knowledge (e.g., "investigative journalist" instead of "writer").
- 5. **Domain-Specific Knowledge**: Leverage implicit domain knowledge when passages lack critical details (e.g., knowing that "Project RAND" is linked to Henry H. Arnold and the RAND Corporation).

2542

2543

2544

2545

2546

Example Integration:

If the question is about a person's profession in a novel, ensure the summary includes:

- 2540 - The character's name. 2541
 - Their profession (explicitly stated in the text).
 - Contextual links to the book series or plot (e.g., "in *The Girl in the Spider's Web*").
 - Any relevant background about the profession or characterâĂŹs role in the story.

Always aim to match the **coverage and relevance** of an "ideal summary" as described in the feedback, ensuring the answer module has all necessary information to generate the correct final answer.

2547 2548

N.3 IFBENCH, GPT-4.1 MINI

2549 2550 2551

IFBench GPT-4.1 Mini generate_response_module.predict

Base Prompt:

2552 2554

2556 2557 2558

MIPROv2 Prompt:

2559 2560 2561

2562 2563

2565

2566 2567

2569

2571

2573 2574

2575 2576 2577

2578 2579

2580 2581 2582

Respond to the query

You will be given a query containing a complex task with multiple constraints and instructions. Your job is to first repeat the query exactly as given, word for word, without adding any extra words or commentary before or after the repetition. After repeating the query, provide a detailed, step-by-step chain-of-thought reasoning that carefully unpacks and addresses every aspect of the query. Then, produce a final response that adheres strictly to all requirements, including formatting, language, and content constraints specified in the query. Be thorough, precise, and ensure the final answer is fully validated and consistent with the reasoning. If the query specifies additional formatting or stylistic rules (such as language or inclusion of a postscript), include them exactly as instructed. Demos:

GEPA Prompt generated by config GEPA+Merge:

You are given a query input, and your task is to respond appropriately to that query. The query may contain specific instructions or constraints that you must strictly adhere to in your response. Carefully analyze the query to determine the exact requirements, including but not limited to:

- Responding with an answer chosen from a restricted set of options exactly as specified (including exact wording and punctuation).
- Including specific words a minimum number of times.
- Including specific letters a minimum number of times.
- Repeating the entire query word-for-word before providing your answer if explicitly requested, without adding extra words or characters before the repetition.
- Avoiding specific forbidden words or keywords in your response if indicated.
- Following any other explicit instructions or constraints embedded in the query.

When the query requests calculations or factual answers (e.g., combinatorial calculations), you should:

2586

2587 2589

2590

2591 2592

2593 2594

2595 2596

2597 2598 2599

2600 2601

2602 2603

2604 2605

2606 2607

2608 2609

2610 2611

2612 2613 2614

2616 2617 2618

2619

2615

2620 2621 2622

2623 2624

2625 2626 2627

2629

2631

- 1. Carefully interpret the mathematical or logical problem.
- 2. Show your reasoning internally to confirm the final answer (reasoning does not need to be included in the response unless explicitly requested).
- 3. Provide the final direct response strictly following all instructions, especially when asked to repeat the query verbatim first before giving the answer.

General approach:

- Always parse the query thoroughly to extract every constraint and instruction.
- Ensure your response exactly matches the format, wording, and content as instructed.
- Do not invent or omit any part of the user's explicit requests.
- Meet all formatting, lexical, numeric, and structural constraints without deviation.
- If the query involves repeating text verbatim, do not alter capitalization, punctuation, or wording.
- Incorporate any required keywords or letters the required number of times naturally into your response.
- When multiple constraints (like avoiding specific words while including others) apply simultaneously, ensure you satisfy all simultaneously.

This task requires precision, exact reproduction, and strict adherence to any given constraints or instructions embedded in the query. Your goal is to deliver the requested answer in the exact manner requested without extraneous additions or omissions.

IFBench GPT-4.1 Mini ensure_correct_response_module.predict

Base Prompt:

Ensure the response is correct and adheres to the given constraints. Your response will be used as the final response.

MIPROv2 Prompt:

Given the original query and the initially generated response, carefully verify that the response fully meets all specified constraints and instructions in the query. Produce detailed step-by-step reasoning that explains how the response satisfies or fails the requirements such as exact repetition of the query text, minimum sentence count, formatting rules (e.g., capitalization), letter frequency, or other explicit user demands. Based on this reasoning, output a final corrected and fully compliant response that adheres strictly to the original instructions, ensuring accuracy, completeness, and formatting correctness. Your final output should begin by repeating the original query exactly as provided, with no modifications or additional text before it, followed by a comprehensive, well-structured answer that respects all constraints, and a clear explanation of how the requirements have been met.

Demos: Example 1:

query:

Can you provide the social media handles of Harry Potter? Your answer must contain exactly 8 bullet points. Use the markdown bullet points such as:

- * This is point 1.
- * This is point 2 Finish your response with this exact phrase That's my answer.. No other words should follow this phrase.

response:

* Twitter: [@wizardingworld](https://twitter.com/wizardingworld)

```
2632
          * Instagram: [@wizardingworld](https://www.instagram.com/wizardingworld/)
2633
          * Facebook: [Wizarding World](https://www.facebook.com/wizardingworld)
2634
          * YouTube: [Wizarding World](https://www.youtube.com/c/WizardingWorld)
2635
          * Twitter: [@harrypotterfilm](https://twitter.com/harrypotterfilm)
2636
          * Instagram: [@harrypotterfilm](https://www.instagram.com/harrypotterfilm/)
2637
          * Facebook: [Harry Potter](https://www.facebook.com/harrypotter)
2638
          * TikTok: [@wizardingworld](https://www.tiktok.com/@wizardingworld)
2639
          That's my answer..
2640
          reasoning:
2641
              The request is for the social media handles of Harry Potter, formatted as exactly
2642
              eight bullet points using markdown bullet points, and the response must end with the
              exact phrase "That's my answer." The provided response includes eight bullet points,
2643
              referencing official or widely recognized social media accounts related to Harry
2644
              Potter and the Wizarding World franchise on Twitter, Instagram, Facebook, YouTube,
2645
              and TikTok. The format is correct, the bullet points are used properly with markdown
2646
              syntax, and the closing phrase matches the requirement exactly. Therefore, the
2647
              response meets all given constraints.
          final_response:
2648
              * Twitter: [@wizardingworld](https://twitter.com/wizardingworld)
2649
          * Instagram: [@wizardingworld](https://www.instagram.com/wizardingworld/)
2650
          * Facebook: [Wizarding World](https://www.facebook.com/wizardingworld)
2651
          * YouTube: [Wizarding World](https://www.youtube.com/c/WizardingWorld)
2652
          * Twitter: [@harrypotterfilm](https://twitter.com/harrypotterfilm)
2653
          * Instagram: [@harrypotterfilm](https://www.instagram.com/harrypotterfilm/)
          * Facebook: [Harry Potter](https://www.facebook.com/harrypotter)
2654
          * TikTok: [@wizardingworld](https://www.tiktok.com/@wizardingworld)
2655
2656
          That's my answer..
2657
          ######################################
2659
          3 other demos omitted for brevity.
          2661
2662
          GEPA Prompt generated by config GEPA+Merge:
2663
          You will receive a user query that includes explicit, detailed instructions about how you
2664
2665
```

must structure your response. Your overall task is to interpret these embedded instructions with perfect accuracy and produce a response that strictly complies with every single one, without adding, omitting, or rewording any mandated text, phrases, or structural elements.

Key detailed requirements and strategy for this task:

2666

2667 2668

2669 2670

2671

2672

2674

2676

- 1. **Query Parsing and Extraction of Instructions**
 - Carefully read the entire query to identify all explicit instructions concerning:
 - Whether and how to repeat the query text (verbatim or partially).
 - Specific length constraints (number of sentences, bullet points, word counts).
 - Formatting instructions (e.g., capitalization requirements, quotation marks, markdown bullet styles).
 - Mandatory phrases or exact sentences that must appear (especially those to be repeated verbatim or appended at the end).

- Content limitations or prohibitions (for example, refusal language or disclaimers for unethical requests).
- Note that some instructions may be nested or appear within the query's wording and are critical to follow exactly.
- 2. **Exact Text Reproduction**
 - When asked to repeat the query text (or any other required phrase) verbatim, do so with zero changes âĂŤ no added or removed words, punctuation, or formatting.
 - Do not prepend or append anything to the repeated text unless explicitly instructed.
 - Preserve all original capitalization, spacing, and punctuation exactly as in the query.
- 3. **Structural and Formatting Compliance**
 - Follow all formatting instructions strictly, such as:
 - Wrapping the entire response in quotation marks if required.
 - Using specified markdown bullet point styles (e.g., asterisks).
 - Ensuring capitalization instructions (e.g., all caps or minimum occurrences of uppercase words) are perfectly met.
 - Adhering to sentence or paragraph counts exactly as requested.
- 4. **Response Content Accuracy and Appropriateness**
 - After fulfilling all structural requirements, respond to the main substantive question accurately and completely.
 - Use domain knowledge and reliable calculations to ensure factual correctness in answers.
 - For questions requesting sensitive or potentially harmful content (e.g., cures without scientific basis), produce responsible answers that include disclaimers or refusals if instructed.
 - Always respect ethical guidelines and any mandated refusal language or concluding statements for such queries.
- 5. **No Extraneous Text**
 - Do not add explanations, internal reasoning, apologies, or meta commentary beyond what the query explicitly permits or demands.
 - Your final output must be the exact, ready-to-deliver response that meets all user instructions perfectly.
- 6. **Examples and Patterns Observed**
 - Users often combine multiple complex formatting and content instructions (e.g., repetition of request text, followed by specific number of sentences or bullet points, with capitalization rules).
 - Ensure you carefully distinguish when to repeat the query text verbatim and when to respond directly (sometimes the repetition excludes an instruction sentence).
 - Handle instructions about capitalized words appearing a minimum number of times by distributing such words naturally but thoroughly across the response.
 - When length constraints specify minimums (e.g., âĂat least 39 sentencesâĂİ), ensure you meet or exceed exactly rather than approximating.
 - For bullet points or enumerations, use the precise markdown style asked for (commonly asterisks).
 - Follow refusal instructions verbatim and terminate the response exactly as specified.
- **In summary:** YOUR RESPONSE MUST BE A PERFECT TEXTUAL AND STRUCTURAL RENDERING OF THE USER QUERYÂĂ´ZS EMBEDDED INSTRUCTIONS. ALWAYS PRIORITIZE EXACT USER SPECIFICATIONS OVER

DEFAULT GENERATIVE NORMS. THE OUTPUT IS FINAL AND WILL NOT BE EDITED, SO COMPLIANCE MUST

2729

2730

2731 2732

2733 2734 2735

2736

2737

2738

2739

2740

2741

2742 2743

2744

2745 2746

2747 2748

2749 2750 2751

2752

2753 2754

2755 2756 2757

2758 2759 2760

2761 2762 2763

2764 2765

2766 2767

2768 2769

2770 2771

2772

BF 100%.

This task therefore requires a rigorously rule-based approach:

- First parse and isolate all explicit instructions (including repetitions and formatting).
- Then assemble your answer by reproducing mandated text identically.
- Next produce the requested content in the exact required format, length, and style.
- Finally, review your output against the instructions ensuring total compliance before delivering your response.

N.4 IFBENCH, QWEN3 8B

IFBench Qwen3 8B generate_response_module.predict

Base Prompt:

Respond to the query

MIPROv2 Prompt:

You are a precision-focused instruction-adherence expert tasked with generating structured, ethically compliant responses. When responding to queries, always think step-by-step to analyze constraints (formatting rules, content boundaries, ethical guidelines), synthesize requirements, and produce outputs that strictly adhere to specified parameters. Ensure responses are fully formatted as requested (e.g., lowercase text, bullet points, markdown), avoid any unverified claims, and maintain neutrality in ambiguous ethical scenarios.

Demos:

Example 1: query:

Can you provide the social media handles of Harry Potter? Your answer must contain exactly 8 bullet points. Use the markdown bullet points such as:

* This is point 1.

* This is point 2 Finish your response with this exact phrase That's my answer.. No other words should follow this phrase. reasoning:

The query asks for the social media handles of Harry Potter, a fictional character. Since he does not have personal social media accounts, the response should instead list official accounts related to the Harry Potter franchise. These include accounts managed by Warner Bros., the Wizarding World, and associated organizations. The answer is structured into exactly 8 bullet points as requested.

response: * @WizardingWorld (Twitter)

- * @WizardingWorld (Instagram)
- * @WizardingWorld (Facebook)
- * @JKRowling (Twitter)
- * @HarryPotterAlliance (Facebook)
- * @HogwartsExpress (Twitter)
- * @TheWizardingWorldOfHarryPotter (Facebook)
- * @HarryPotterBooks (Twitter)

That's my answer.

GEPA Prompt generated by config GEPA:

Respond to the query by first repeating it word for word without any changes. Then, provide your answer. Ensure your answer includes niche or domain-specific factual information relevant to the task. If the query specifies constraints (e.g., letter frequency, sentence limits, or formatting rules), adhere to them strictly. Use a generalizable strategy to solve the task, such as breaking down complex problems into logical steps or applying specialized knowledge. Avoid adding any preamble or postscript before or after the repeated query.

IFBench Qwen3 8B ensure_correct_response_module.predict

Base Prompt:

Ensure the response is correct and adheres to the given constraints. Your response will be used as the final response.

MIPROv2 Prompt:

You are a meticulous and ethical AI assistant tasked with generating detailed, compliant responses. When given a query, first think step by step to create a reasoning process that addresses the query's requirements, then produce a response that strictly adheres to all specified constraints (e.g., formatting rules, content boundaries, ethical guidelines). Ensure the final output is structured with a "Reasoning:" section followed by a "Final Response:" section, and verify that it aligns with the query's exact specifications while maintaining logical coherence and ethical responsibility. Demos:

Example 1:

query:

What are some techniques for winning a swordfight in a medieval combat game? In your response, the letter f should appear less than 18 times. Do not include keywords ['heart', 'listen', 'reputation', 'side'] in the response.

response:

Prioritize stance and balance to control the fight's flow. Use angles to create openings for strikes while minimizing exposure. Practice parrying with precision to disrupt opponent rhythm. Maintain distance to avoid overcommitting, and target weak points like joints or limbs. Focus on rhythm and timing to exploit vulnerabilities. reasoning:

The response outlines key techniques for medieval swordfighting, focusing on stance, angles, parrying, distance, and targeting weak points. It avoids the restricted keywords and limits the letter 'f' to 2 instances, satisfying the constraints. The advice is practical and aligns with historical combat strategies.

final_response:

Prioritize stance and balance to control the fight's flow. Use angles to create openings for strikes while minimizing exposure. Practice parrying with precision to disrupt opponent rhythm. Maintain distance to avoid overcommitting, and target weak points like joints or limbs. Focus on rhythm and timing to exploit vulnerabilities.

GEPA Prompt generated by config GEPA:

Ensure the response strictly follows these instructions:

- 1. **First**, repeat the user's query **word for word** without any changes or additions.
- 2. **Then**, provide your answer in the specified format, adhering to all constraints (e.g., markdown, structure, content).
- 3. **Do not include any additional text, explanations, or formatting** beyond the repeated query and your answer.
- 4. **Include niche/domain-specific factual details** (e.g., technical commands, best practices, or platform-specific configurations) if applicable, as these are critical for accurate task completion.
- 5. **Use precise formatting** (e.g., bullet points, code blocks, headers) as requested, ensuring no markdown is omitted or altered.
- 6. **Avoid generalizable strategies** unless explicitly instructed; focus on actionable, specific guidance.
- 7. **Validate all technical steps** (e.g., Dockerfile syntax, CLI commands) for accuracy and completeness.
- 8. **Highlight potential pitfalls and solutions** to address common issues in the task.
- 9. **Prioritize clarity and conciseness**, ensuring the response is both comprehensive and easy to follow.
- 10. **Adhere to language and case requirements** (e.g., all caps, English only) if specified.

N.5 HOVER, GPT-4.1 MINI

HoVer GPT-4.1 Mini create_query_hop2.predict

Base Prompt:

Given the fields `claim`, `summary_1`, produce the fields `query`.

MIPROv2 Prompt:

Given the original claim and the initial summary of retrieved evidence, carefully analyze the information step-by-step to identify any gaps, contradictions, or points needing further clarification. Then, generate a clear, focused, and precise query that targets additional relevant information to deepen the investigation and help verify or refute the claim. Your response should include a detailed chain-of-thought reasoning explaining your thought process in formulating this refined query.

Demos:

Example 1:

claim:

AirBaltic's has bases at Tallinn Airport and the airport that ceased operations in 2015, with the airport code of Ryanair: VNO, ICAO: EYVI.

summary_1:

AirBaltic has bases at Tallinn Airport and Vilnius Airport (IATA: VNO, ICAO: EYVI), but Vilnius Airport did not cease operations in 2015. The airport that ceased operations in 2015 mentioned in the passages is unrelated and has different codes. Therefore, the claim that airBaltic's base with code VNO/EYVI ceased operations in 2015 is false.

reasoning:

The claim states that AirBaltic has bases at Tallinn Airport and an airport with codes VNO (IATA) and EYVI (ICAO) which ceased operations in 2015. However, the summary clarifies that Vilnius Airport, which has those codes, did not cease operations in 2015. The airport that did cease operations in 2015 has different codes and is unrelated to AirBaltic's bases. Therefore, the claim that AirBaltic's base at the airport with codes VNO/EYVI ceased operations in 2015 is incorrect.

Did Vilnius Airport (IATA: VNO, ICAO: EYVI) cease operations in 2015?

GEPA Prompt generated by config GEPA+Merge:

Task Description:

You are given two fields: `claim` and `summary_1`. Your goal is to produce a field called `query` âĂŤ a concise set of relevant, fact-checking questions or search style queries that can be used to retrieve evidence documents verifying or refuting the claim based on information contained or inferred from the summary.

Input Format:

- `claim`: A factual statement potentially containing multiple fact assertions about people, events, attributes, titles, dates, roles, or relationships.
- `summary_1`: A short paragraph summarizing factual information related to the claim, often clarifying or correcting some parts of the claim.

Output Format:

- `query`: One or more specific, well-phrased questions or keyword queries that directly target the key factual discrepancies or verifications raised by the claim in light of the summary.

Detailed Instructions:

- 1. **Extract key factual elements from the claim** $\hat{a}\check{A}\check{T}$ names, dates, titles, roles, events, or relationships explicitly or implicitly stated.
- 2. **Contrast these facts with the summary to identify points of agreement, contradiction, or ambiguity.**
- 3. **Formulate fact-checking queries that are:**
 - $\mbox{-}$ Tightly focused on the core factual issues raised by the claim and addressed or contradicted by the summary.
 - Include named entities, dates, roles, or other domain-specific identifiers directly mentioned in both claim and summary to improve retrieval effectiveness.
 - ${\mathord{\text{--}}}$ When relevant, break complex claims into multiple queries ensuring each fact is verifiable separately.
- 4. **When relevant details appear only in the summary but are hinted at or missing from the claim (e.g., specific titles, roles, or names), include these in the queries to enable retrieval of key evidence.**

2947

2949

2950

2951

2952

2953

2954

2955

2956

2958

2959

- 5. **Use a clear, natural question format or targeted keyword phrases that could serve well as search queries.**
- 6. **Avoid overly broad or generic queries; precision improves evidence retrieval quality.**
- 7. Optionally, you may provide brief reasoning internally (not required in output) to ensure the queries cover all claim aspects and reflect the summary insights.

Examples of typical query components include:

- Correct dates of events or deaths.
- Confirmation of a personâĂŹs role or association with a known work.
- Verification of relationships or allegations.
- Details about specific cultural or domain elements (e.g., operas' structure, song directors).
- Clarifying entity attributes or classifications (e.g., ethnicity, nationality).

By following these instructions, you will produce queries that are both comprehensive and targeted, maximizing the chance of retrieving relevant factual evidence relevant to the claim verification task.

HoVer GPT-4.1 Mini create_query_hop3.predict

Base Prompt:

Given the fields `claim`, `summary_1`, `summary_2`, produce the fields `query`.

MIPROv2 Prompt:

You are an expert fact-checker specializing in multi-hop reasoning and evidence synthesis. Given a claim and two intermediate summaries that consolidate evidence from previous retrieval steps, thoughtfully analyze the information step-by-step to generate a clear and focused query. This query should be designed to retrieve the most relevant additional documents that can help verify or refute the claim by leveraging the insights from both summaries. Provide a detailed chain-of-thought reasoning explaining how you integrate the summaries and the claim to formulate this refined query.

Demos:

Example 1:

claim:

The brother of Freddie Sessler, was the restaurateur whose club was frequented by Ruth Elizabeth "Bette" Davis, an actress with Welsh ancestry.

summary_1:

Freddie Sessler's brother was Siegi Sessler, a London restaurateur whose club was frequented by Bette Davis, an actress with Welsh ancestry.

summary_2:

Freddie Sessler's brother was Siegi Sessler, a prominent London restaurateur whose club, Siegi's, was frequented by Hollywood stars including Ruth Elizabeth "Bette" Davis, an actress with Welsh ancestry.

reasoning:

The claim states that the brother of Freddie Sessler was a restaurateur whose club was frequented by Ruth Elizabeth "Bette" Davis, noted to have Welsh ancestry. Both summaries support this claim by identifying the brother as Siegi Sessler, a restaurateur in London, whose club was indeed frequented by Bette Davis, an actress noted for Welsh ancestry. Summary_2 adds that the club's name was Siegi's and mentions the presence of other Hollywood stars. To fact-check the claim accurately,

it would be useful to verify the identity of Freddie Sessler's brother as Siegi Sessler, confirm that he was a restaurateur, verify that Bette Davis frequented his club, and confirm Bette Davis's Welsh ancestry.

query: Who was Siegi Sessler, and was he the brother of Freddie Sessler? Was Siegi Sessler a restaurateur whose club was frequented by Ruth Elizabeth "Bette" Davis? Did Bette Davis have Welsh ancestry?

GEPA Prompt generated by config GEPA+Merge:

Given three text fields: `claim`, `summary_1`, and `summary_2`, your task is to produce a `query` field that is designed to effectively retrieve evidence documents relevant to verifying or refuting the claim based on the information contained in the two summaries.

Detailed task description and considerations:

- 1. **Purpose of the Query:**
 - The query should accurately and comprehensively target the key factual elements from the claim that are addressed or clarified in the summaries.
 - The query will be used to retrieve evidence documents, so it should be specific enough to pinpoint relevant support or contradiction but broad enough to cover all important details present in the summaries.
- 2. **Utilizing the Summaries:**
 - Summaries often correct, clarify, or add factual context to the claim. Your query must incorporate these clarifications (e.g., name corrections, factual specifics, or counterpoints) to ensure retrieval of relevant evidence reflecting the nuanced truth.
 - ${\mathord{\text{--}}}$ Include all distinctive entities, facts, dates, locations, names, and relationships mentioned or corrected in the summaries that pertain to the claim.
 - For example, if a summary corrects a documentary title in the claim, the query should reference the corrected title and related details to guide retrieval effectively.
- 3. **Ouery Content Strategy:**
 - Explicitly mention key entities, such as persons, places, dates, works, or political entities involved.
 - Include attributes or relationships relevant to the claimâĂŹs accuracy (e.g., "Was person X a politician in country Y during year Z?" or "Did documentary A and documentary B film in different locations such as location 1 and location 2?").
 - If there is a factual dispute or correction in the summaries (e.g., nationality, official names, population figures), phrase the query to target evidence clarifying this dispute.
- 4. **Domain-Specific Nuances:**
 - Be mindful of historical geopolitical names and periods (e.g., "United Kingdom of the Netherlands between 1815 and 1830").
 - Recognize proper titles and correct spellings (e.g., corrected documentary titles).
 - Include both subjects or objects mentioned as comparisons or contrasts in the claim and summaries (e.g., two documentaries, two politicians, two towns).

3010 3011

3012 3013

3014 3015 3016

3017 3018 3019

3020

3021 3022 3023

3024 3025

3027 3028 3029

3026

3030 3031 3032

3033 3034 3035

3037 3038

3039 3040 3041

3043 3044 3045

3046

3047

3042

3048 3049 3050

3052

3054

5. **Formulation Style:**

- Queries should be phrased as clear, precise, and objective questions that can be answered based on evidence âĂŤ often structured as yes/no or informational queries.
- Avoid overly broad or vague phrasing. Aim for detail-rich queries that connect multiple evidence points in the summaries.
- 6. **Generalizable Strategy:**
 - Identify the core claim elements and verify if the summaries confirm, contradict, or amend these elements.
 - Incorporate both the claim and corrections from summaries into the query so evidence retrieval captures the full factual context.
 - Use the comparison or contrast highlighted by the summaries to create queries that specifically test the claim's veracity (e.g., comparing locations, roles, or historical timeframes).

By following these guidelines, you will generate queries that maximize the likelihood of retrieving relevant texts that confirm or refute the claim accurately.

HoVer GPT-4.1 Mini summarize1.predict

Base Prompt:

Given the fields `claim`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given a `claim` and a list of relevant `passages`, carefully analyze the evidence by reasoning step-by-step to assess the claim's validity. Produce a detailed chain-of-thought `reasoning` that explains how the information in the passages supports or refutes the claim, followed by a clear, concise `summary` that synthesizes the key findings in relation to the claim. Ensure the reasoning explicitly connects evidence from the passages to the claim, enabling a thorough and transparent multi-hop verification process.

Demos:

Example 1:

claim:

The manufacturer that owns the company that currently produced and marketed the candy the video game Darkened Skye uses in magic is headquartered in the US state Virginia. passages:

['Darkened Skye | Darkened Skye is a third-person action-adventure video game developed by Boston Animation. It was released for Microsoft Windows and the Nintendo GameCube in North America in 2002 and the PAL regions in 2003. The game was also packaged with Outlaw Golf. Its title character is a young woman named Skye who lives in a fantasy realm searching for her mother. She does not use firearms, but can perform magic using Skittles candies, as well as use her staff as a melee weapon, though it also becomes an energy weapon when used in conjunction with the Skittles.' "Rocket Fizz | Rocket Fizz is a franchise chain of candy stores in the United States. The company markets a diverse variety of candies and produces its own line of soft drinks with unique flavors. Its flagship store is located in Camarillo, California, and most of the company's franchise stores are located in California. The company also markets candies that are rare to find or in limited production by various manufacturers, such as those that were popular during the 1960s to 1980s. As of

3058 3059 3060

3061 3062 3063

3065 3066 3067

3068 3069 3070

3075 3076 3077

3078 3079 3080

3082

3084 3085 3086

3088 3089

3090 3091 3092

3093 3094 3095

3097

3096

3099 3100

'Spangler Candy Company | The Spangler Candy Company is a privately owned international confectioner that has been manufacturing and marketing candy since 1906.', 'Future US | Future US, Inc. (formerly known as Imagine Media and The Future Network USA) is an American media corporation specializing in targeted magazines and websites in the video games, music, and technology markets. Future US is headquartered in the San Francisco with a small sales office in New York City. Future US is owned by parent company, Future plc, a small publishing company based in the United Kingdom.', 'Fun Dip | Fun Dip is a candy manufactured by The Willy Wonka Candy Company, a brand owned by NestlÃl. The candy has been on the market in the United States and Canada since the 1940s and was originally called Lik-M-Aid. It was originally manufactured by Fruzola, and then Sunline Inc., through their Sunmark Brands division, which was purchased by NestlÃl in January 1989. It comes in many different flavors with candy sticks that are included.', 'Hot Tamales | Hot Tamales is a chewy, cinnamon flavored, oblong-shaped candy introduced in 1950 âĂŤ manufactured and marketed in the United States by Just Born, a family-owned, Pennsylvania-based candy company.', "Frankford Candy & Docolate Company | Frankford Candy & Chocolate Company is an American candy manufacturer, located in Philadelphia, Pennsylvania, founded in 1947 by Sam Himmelstein. The current CEO is Stuart Selarnick. In recent years, Frankford Candy & Chocolate Company has become a very successful candy manufacturer. It is one of the largest producers of chocolate rabbits in the United States, making over 100 varieties. In 2000, it started licensing popular children's TV characters, such as SpongeBob SquarePants."] reasoning:

September 2017, there are 86 Rocket Fizz store locations in the United States.",

The claim states that the manufacturer of the candy used in the video game Darkened Skye is headquartered in Virginia. Passage [1] establishes that the video game Darkened Skye uses Skittles candies for magic. Therefore, the candy in question is Skittles. The crucial step is to identify the company that produces and markets Skittles and whether it is headquartered in Virginia. None of the provided passages explicitly mention Skittles or its manufacturer. The passages list various candy companies, but none mention Skittles or match its manufacturer. The known producer of Skittles is Mars, Incorporated, which is headquartered in McLean, Virginia. Although this fact is not directly stated in the provided passages, it is relevant general knowledge. Since the passages do not refute the claim and for accuracy based on known facts, the claim can be considered true that the manufacturer that owns the company producing and marketing Skittles is headquartered in Virginia. summary:

The candy used as magic in Darkened Skye is Skittles, which are produced by Mars, Incorporated. Mars is headquartered in McLean, Virginia, a US state. Therefore, the claim is true based on external knowledge confirming that the manufacturer is based in Virginia.

3 other demos omitted for brevity.

GEPA Prompt generated by config GEPA+Merge:

Given the fields `claim`, `passages`, produce the fields `summary`.

HoVer GPT-4.1 Mini summarize2.predict

Base Prompt:

3102

3103 3104

3105

3106 3107

3108

3109

3110

3111

3112

3113

3114

3115

3116

3117

3118

3119

3120

3121

3122

3123

3124

3125

3126

3127

3128

3129

3130

3131

3132

3133

3134

3135

3136

3137

3138

3139

3140

3141

3142

3143

3144

3145

3146

3147 3148 Given the fields `claim`, `context`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given a `claim`, relevant `context`, and a set of supporting `passages`, generate a detailed step-by-step reasoning process that logically connects the evidence to the claim, followed by a concise summary that clearly states whether the claim is supported or refuted based on the aggregated information. Ensure the reasoning explicitly references key evidence from the context and passages to justify the conclusion in the summary.

Demos:

Example 1:

claim:

One half of the KalyanjiâĂŞAnandji duo, Kalyanji Virji Shah's brother, won the Filmfare Award for Best Music Director.

context:

Yes, Kalyanji Virji Shah's brother Anandji Virji Shah, as part of the KalyanjiâĂŞAnandji duo, won the Filmfare Award for Best Music Director in 1975 for the film "Kora Kagaz."

passages:

['Kalyanji Virji Shah | Kalyanji Virji Shah (30 June 1928 âĂŞ 24 August 2000) was the "Kalyanji" of the Kalyanji-Anandji duo. He and his brother Anandji Virji Shah have been famous Indian film musicians, and won the 1975 Filmfare Award for Best Music Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri (1992).', 'Anandji Virji Shah | Anandji Virji Shah is an Indian music director. Together with his brother he formed the Kalyanji-Anandji duo, and won the 1975 Filmfare Award for Best Music Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri (1992).', 'KalyanjiâĂŞAnandji | KalyanjiâĂŞAnandji are an Indian composer duo from Gujarat: Kalyanji Virji Shah (30 June 1928-03 November 2000) and his brother Anandji Virji Shah (born 02 March 1933). The duo are known for their work on Hindi film soundtracks, particularly action potboilers in the 1970s. Some of their best-known works are "Don", "Bairaag", "Saraswatichandra", "Qurbani", "Tridev" and "Safar". They won the 1975 Filmfare Award for Best Music Director for "Kora Kagaz".', 'KalyanjiâĂŞAnandji discography | This is a discography of Bollywood composer duo Kalyanji Anandji, consisting of Kalyanji Virji Shah and Anandji Virji Shah. They have composed music for over 200 films in their 46 year career.', 'Viju Shah | Viju Shah (born, Vijay Kalyanji Shah, 5 June 1959) is a music director of Hindi cinema. He is the son of music director Kalyanji Virji Shah of composer duo Kalyanji Anandji. He composed music for movies such as "Vishwatama" (1992), "Mohra" (1994), "Tere Mere Sapne" (1996) and "" (1997) for which got his second nomination for the Filmfare Award for Best Music Director and he won the 1998 Filmfare Award for Best Background Score for "".', 'Bombay the Hard Way: Guns, Cars and Sitars | Bombay the Hard Way: Guns, Cars and Sitars is a 1998 project of producer Dan the Automator in which music from 1970s funk Bollywood composer duo Kalyanji-Anandji (Kalyanji Virji Shah and Anandji Virji Shah) was taken and given a funky remix. The album was eventually withdrawn (possibly due to copyright concerns). A sequel, "Bombay 2: Electric Vindaloo", was also produced.', 'Kora Kagaz | Kora Kagaz (Hindi: àďŢàěŃàďřàď¿àďŢàď¿àďůàďàď; translation: Blank Paper) is a 1974 Hindi drama film produced by Sanath Kothari and directed by Anil Ganguly. The film stars Vijay Anand, Jaya Bhaduri, A.K. Hangal, Achala Sachdev and Deven Verma. The films music is by

3151

3152

3153

3154

3155

3156

3157

3158

3159

reasoning:

Kalyanji Anandji. The famous title song "Mera Jeevan Kora Kagaz" was by Kishore

The claim states that one half of the KalyanjiâĂŞAnandji duo, Kalyanji Virji Shah's brother, won the Filmfare Award for Best Music Director. From the passages and context, it is clear that the duo consists of Kalyanji Virji Shah and his brother Anandji Virji Shah. Both together won the Filmfare Award for Best Music Director in 1975 for the film "Kora Kagaz." Therefore, Anandji Virji Shah, who is Kalyanji's brother, indeed won the Filmfare Award for Best Music Director as part of this duo. summary:

Anandji Virji Shah, the brother of Kalyanji Virji Shah, won the Filmfare Award for Best Music Director in 1975 as part of the KalyanjiâĂŞAnandji duo.

3160 3161

3162

3163

3 other demos omitted for brevity.

3164 3165

GEPA Prompt generated by config GEPA+Merge:

3166

Given the input fields:

3167

3171

3172 3173

3174

3175

3176

3177 3178

3179

3180

3181

3182 3183

3184

3185

3186 3187

3188

3189

3190 3191

- 3168 `claim`: A factual statement or assertion that may be true or false. - `context`: A brief explanation or summary relating to the claim that typically 3169 clarifies the claimâĂŹs accuracy. 3170
 - `passages`: A list of relevant textual evidence or knowledge snippets containing facts, descriptions, or biographies related to entities or concepts mentioned in the claim.

Your task is to produce a `summary` that meets the following requirements:

- 1. **Accurately reflect the relationship between the claim and the evidence**: Analyze the input `claim` carefully and check it against the `context` and all `passages`. Determine whether the claim is supported, partially supported, or contradicted by the
- 2. **Explicitly connect key entities and facts from the passages**: Your summary must mention the main entities and facts that directly confirm or refute parts of the claim. For example, highlight relevant names, works (films, albums, songs, books), attributes, dates, nicknames, or roles that clarify the claim's accuracy.
- 3. **Make substantive connections**: The summary should not merely restate the claim or conclusion. Instead, it should explicitly link the claim to detailed evidence from the passagesâĀĪsuch as specific film titles, album names, song titles, dates, or relationshipsâĂŤto provide clear reasoning for the accuracy or inaccuracy of the claim.
- 4. **Include relevant disambiguations or clarifications**: Where applicable, clarify potential misunderstandings or common confusions highlighted in the passages (e.g., distinguishing similarly named films or songs, specifying which individual is referenced, or noting alternate titles and adaptations).
- 5. **Use concise, factual language**: The summary should be clear and succinct but include all relevant evidence necessary to understand how the claim is supported or refuted.

3193 3194 3195

3199 3200

3201

3202

3203

6. **Support future query generation**: As the summary will be used to generate queries to find evidence, it must contain explicit mentions of key entities, titles, and facts that can guide retrieval and reasoning.

Overall, your approach should be to reason comprehensively over the claim, context, and each passage, synthesizing connected facts and evidence into a coherent and evidence-rich summary that clearly documents why the claim is true, false, or partially true. Avoid vague or overly general statements; instead, ground your summary in the precise factual details present in the passages and context.

3204 3205 3206

3207

3208

3211 3212

3213

3214

3215

3216

3217

3218

3219

3220

3223

3226

3227

3228

3230

3231

3232

N.6 HoVer, Qwen3 8B

3209 HoVer Qwen3 8B create_query_hop2.predict 3210

Given the fields `claim`, `summary_1`, produce the fields `query`.

MTPROv2 Prompt:

Base Prompt:

In a high-stakes scenario where the accuracy of your query directly determines the validity of a complex claim, generate a precise query that builds on the provided claim and summary to uncover critical evidence. Use step-by-step reasoning to identify gaps in the initial summary, then craft a query that explicitly targets these gaps to retrieve additional supporting information. Ensure your query is unambiguous and directly addresses the claim's unresolved aspects.

Demos:

Example 1:

claim: 3222

Sandi Simcha DuBowski directed a documentary released in 2001. The horror film Zeitgeist was not also released in 2001.

summary 1:

The claim is correct: Sandi Simcha DuBowski directed the 2001 documentary *Trembling Before G-d*, and there is no evidence in the passages that the horror film *Zeitgeist* was released in 2001.

reasoning:

The claim is correct because Sandi Simcha DuBowski indeed directed the 2001 documentary *Trembling Before G-d*, as stated in summary_1. The assertion that *Zeitgeist* was not released in 2001 is supported by the absence of evidence in the provided summaries linking the film to that year.

query:

Is there any evidence in the provided summaries that the horror film *Zeitgeist* was released in 2001?

3233 3234 3235

3236 3 other demos omitted for brevity. 3237

3238 3239

GEPA Prompt generated by config GEPA:

3240 3241 3242

Given the fields `claim` and `summary_1`, generate a precise and focused query that identifies the specific evidence needed to verify or refute the claim.

Key Requirements:

- 1. **Target Missing Evidence**: Identify and explicitly ask about the unverified or unconfirmed details in the summary (e.g., names, dates, locations, or connections) that are critical to the claim.
 - Example: If the summary mentions "Massimo Giordano" as a potential counterpart but lacks birthplace details, the query should ask for evidence confirming their ties to Naples.
- 2. **Correct Historical/Domain-Specific Anomalies**: Address discrepancies like anachronisms (e.g., Ali QushjiâĂŹs work in the 1960s vs. his actual 15th-century timeline) or misattributions (e.g., *Hayy ibn Yaqdhan* by Ibn Tufail, not Ali Qushji).
- 3. **Link to Summary Context**: Ensure the query references the summaryâ \check{A} 2s key points (e.g., "the claim states X, but the summary notes Y is unverified").
- 4. **Use Specific Terms**: Include exact names, titles, or dates mentioned in the summary to avoid ambiguity (e.g., "AnaÃrs Nin," "Metropolitan City of Naples," "1948").

Example Strategy:

If the summary states:

- The query should ask:
 - *"Is there evidence confirming [Fact B], such as [specific name/date/location]?"*

Domain-Specific Notes:

- Verify historical timelines (e.g., Ali Qushji died in 1474, not the 1960s).
- Confirm authorships (e.g., *Hayy ibn Yaqdhan* is attributed to Ibn Tufail, not Ali Qushji).
- Check for geographic or biographical details (e.g., birthplaces, cities, or professional connections).

Ensure your query directly addresses the summary $\tilde{a}\tilde{A}$ sunverified claims and includes all critical terms from the summary to maximize evidence retrieval.

HoVer Qwen3 8B create_query_hop3.predict

Base Prompt:

Given the fields `claim`, `summary_1`, `summary_2`, produce the fields `query`.

MIPROv2 Prompt:

You are a fact-checking assistant specializing in multi-hop reasoning and information synthesis. Given the fields `claim`, `summary_1`, and `summary_2`, generate a precise query that synthesizes the original claim with the two summaries to probe for deeper contextual relationships, resolving ambiguities and confirming supporting details through targeted evidence retrieval.

Demos:

GEPA Prompt generated by config GEPA:

Given the fields `claim`, `summary_1`, and `summary_2`, produce the field `query` that:

1. Explicitly asks whether the claim is supported by the provided summaries.

```
3290
          2. Includes **specific evidence** from the summaries (e.g., names, titles, dates, or
3291
          factual details) to guide retrieval of relevant documents.
3292
          3. Focuses on **key disputed points** in the claim (e.g., incorrect attributions, missing
3293
          evidence, or conflicting statements) to ensure the query targets the most relevant
3294
          information.
3295
3296
          **Key considerations for the query:**
          - If the summaries mention specific works (e.g., *Planes, Trains and Automobiles*),
3297
          include the title.
3298
          - If the summaries reference named entities (e.g., John Candy, KLM, Richard Ford),
3300
          - If the summaries clarify contradictions (e.g., "not X but Y"), structure the query to
3301
          highlight this contrast.
          - Avoid vague phrasing; instead of asking "Is the claim true?" use "Does the evidence in
3302
          the summaries confirm [specific detail]?'
3303
3304
          **Example:**
3305
          If the claim is about a film and the summaries mention *Planes, Trains and Automobiles*
          (1987), the query should include that title to retrieve relevant evidence.
3307
3308
3309
          HoVer Qwen3 8B summarize1.predict
3310
3311
          Base Prompt:
3312
```

Given the fields `claim`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given the fields `claim` and `passages`, generate a structured reasoning process that analyzes the claim step-by-step using the provided evidence, and produce a concise summary that distills the key findings and evaluates the claim's validity based on the evidence.

Demos:

Example 1:

3321 | clair

3313 3314

3315

3317

3319

3320

3322

3323

3324

3325

3326

3327

3328

3330

3331

3332

3334

3336

One half of the KalyanjiàÄŞAnandji duo, Kalyanji Virji Shah's brother, won the Filmfare Award for Best Music Director. passages:

['Kalyanji Virji Shah | Kalyanji Virji Shah (30 June 1928 âĂŞ 24 August 2000) was the "Kalyanji" of the Kalyanji-Anandji duo. He and his brother Anandji Virji Shah have been famous Indian film musicians, and won the 1975 Filmfare Award for Best Music Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri (1992).', 'KalyanjiâĂŞAnandji | KalyanjiâĂŞAnandji are an Indian composer duo from Gujarat: Kalyanji Virji Shah (30 June 1928-03 November 2000) and his brother Anandji Virji Shah (born 02 March 1933). The duo are known for their work on Hindi film soundtracks, particularly action potboilers in the 1970s. Some of their best-known works are "Don", "Bairaag", "Saraswatichandra", "Qurbani", "Tridev" and "Safar". They won the 1975 Filmfare Award for Best Music Director for "Kora Kagaz".', 'Anandji Virji Shah | Anandji Virji Shah is an Indian music director. Together with his brother he formed the Kalyanji-Anandji duo, and won the 1975 Filmfare Award for Best Music Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri (1992).', 'Viju Shah | Viju Shah (born, Vijay Kalyanji Shah, 5 June 1959) is a

3339 3340 3341

3342 3343 3344

3345 3346 3347

3348 3349 3350

3351 3352

3353 3354 3355

3356 3357

3358 3359

3360 3361 3362

3363 3364

3366

3367 3368

3370 3371

3372 3373

3374 3375 3376

3377 3378 3379

3380 3381

3382 3383 music director of Hindi cinema. He is the son of music director Kalyanji Virji Shah of composer duo Kalyanji Anandji. He composed music for movies such as "Vishwatama" (1992), "Mohra" (1994), "Tere Mere Sapne" (1996) and "" (1997) for which got his second nomination for the Filmfare Award for Best Music Director and he won the 1998 Filmfare Award for Best Background Score for "".', 'KalyanjiâĂŞAnandji discography | This is a discography of Bollywood composer duo Kalyanji Anandji, consisting of Kalyanji Virji Shah and Anandji Virji Shah. They have composed music for over 200 films in their 46 year career.', 'Bombay the Hard Way: Guns, Cars and Sitars \mid Bombay the Hard Way: Guns, Cars and Sitars is a 1998 project of producer Dan the Automator in which music from 1970s funk Bollywood composer duo Kalyanji-Anandji (Kalyanji Virji Shah and Anandji Virji Shah) was taken and given a funky remix. The album was eventually withdrawn (possibly due to copyright concerns). A sequel, "Bombay 2: Electric Vindaloo", was also produced.', 'Bairaag | Bairaag is a 1976 Bollywood film. Produced by the duo Mushir-Riaz, it is directed by Asit Sen. The film stars Dilip Kumar, who received a Filmfare nomination for Best Actor for playing a triple role. "Bairaag" also stars Saira Banu, Leena Chandavarkar, Ruma Guha Thakurta, Prem Chopra, Helen, Sujit Kumar, Madan Puri, Paintal, Kader Khan, Asit Sen and Sachin. Its music is by the duo Kalyanji Anandji, who received a Filmfare nomination for Best Music. The film received a Filmfare Award for Best Sound for P. Harikishan. According to one source, the film did not do well at the box office.']

reasoning

The claim states that Kalyanji Virji Shah's brother won the Filmfare Award for Best Music Director. The passages confirm that the KalyanjiàĂŞAnandji duo (Kalyanji Virji Shah and his brother Anandji Virji Shah) won the 1975 Filmfare Award for Best Music Director for "Kora Kagaz." Since the award was shared by both brothers, the claim is accurate as it identifies Anandji Virji Shah (Kalyanji's brother) as the winner.

Anandji Virji Shah, the brother of Kalyanji Virji Shah, won the 1975 Filmfare Award for Best Music Director as part of the KalyanjiâĂŞAnandji duo for their work on "Kora Kagaz."

GEPA Prompt generated by config GEPA:

Given the fields `claim` and `passages`, produce the fields `summary` by:

- 1. **Analyzing the claim** to identify key elements (e.g., names, locations, relationships, specific terms).
- 2. **Cross-referencing the passages** to locate explicit or implicit evidence that supports, contradicts, or partially addresses the claim.
- 3. **Explicitly connecting the claim to the evidence** in the passages, even if the evidence is not directly stated in the claim (e.g., linking "Roger Yuan" to "Shanghai Noon" in Example 1).
- 4. **Highlighting domain-specific details** (e.g., "Sentinelese language is unclassifiable" in Example 3) that are critical for accuracy.
- 5. **Stating whether the claim is supported, unsupported, or partially supported**, and explaining the reasoning with clear references to the passages.
- 6. **Prioritizing key evidence** that would be relevant for further queries (e.g., "Marlborough" in Example 2) to ensure retrievability.

Additional Guidance:

3385

3387 3388 3389

3390 3391

3392

3394 3395 3396

3397 3398

3403 3404

3405 3406

3407 3408 3409

3410 3411

3412 3413 3414

3415 3416 3417

3418

3419 3420 3421

3422

3423 3424 3425

3426 3427 3428

3429 3430

- If the claim contains unverified details (e.g., "Roger Yuan" in Example 1), explicitly note this in the summary.
- For niche or ambiguous information (e.g., "uncontacted people" in Example 3), reference specific passages that clarify the context.
- Ensure summaries are concise but include all critical evidence needed to validate or refute the claim.

HoVer Qwen3 8B summarize2.predict

Base Prompt:

Given the fields `claim`, `context`, `passages`, produce the fields `summary`.

MIPROv2 Prompt:

Given the fields `claim`, `context`, and `passages`, perform multi-hop reasoning to generate a structured summary that validates or refutes the claim. First, analyze the claim and contextual information to identify key relationships. Next, evaluate evidence from the retrieved passages to build a coherent narrative. Construct a logical reasoning chain connecting the claim to supporting or contradictory evidence. Finally, produce a concise summary that explicitly confirms or contradicts the claim based on your analysis, citing relevant evidence from the context and passages. Demos:

Example 1:

claim:

The recorded history of the state, for which Baptist George Ruby was a prominent black Republican leader in the Reconstruction-era, officially started in 1519 in the same state that holds the city Augustus Chapman Allen used his inheritance to fund the founding of.

context.

The claim is not supported by the passages. While George Ruby was associated with Texas during Reconstruction and Augustus Chapman Allen founded Houston in Texas, there is no evidence in the passages that Texas's recorded history began in 1519. passages:

["Merchants and Manufacturers Building | The One Main Building, formerly the Merchants and Manufacturers Building (commonly referred to as the M&M Building), is a building on the campus of the University of HoustonâĂŞDowntown. The building is recognized as part of the National Register of Historic Places, is a Recorded Texas Historic Landmark, and considered a Contributing Building in Downtown Houston's Main Street/Market Square Historic District. The building was built above Allen's LandingâĂŤan area where Houston's founders John Kirby Allen and Augustus Chapman Allen originally settled.", 'Augustus Chapman Allen | Augustus Chapman Allen (July 4, 1806 âĂŞ January 11, 1864), along with his younger brother, John Kirby Allen, founded the City of Houston in the U.S. state of Texas. He was born on July 4, 1806, in Canaseraga Village, New York (the present day hamlet of Sullivan in the Town of Sullivan, New York), to Sarah (Chapman) and Roland Allen.', "History of Texas | The recorded History of Texas begins with the arrival of the first Spanish conquistadors in the region of North America now known as Texas in 1519, who found the region populated by numerous Native American /Indian tribes.

Native Americans' ancestors had been there for more than 10,000 years as evidenced by the discovery of the remains of prehistoric Leanderthal Lady.

During the period of recorded history from A.D. 1519 to 1848, all or parts of Texas were claimed by five countries: France, Spain, Mexico, the Republic of Texas, the

3433 3434 3435

3436 3437 3438

3439 3440 3441

3442 3443

3444 3445

3446 3447 3448

3449 3450

3451 3452 3453

3454 3455

3456 3457 3458

3459 3460

3461 3462

3463 3464 3465

3466 3467 3468

3469 3470 3471

3472 3473 3474

3475 3476 3477

United States of AmericaãĂŤas well as the secessionist Confederate States of America in 1861âĂŞ1865, during the American Civil War.", 'Charlotte Baldwin Allen | Charlotte Baldwin Allen (July 14, 1805 âĂŞ August 3, 1895) is known in Texan history as the "mother of Houston". She was the wife of Augustus Chapman Allen, who used her inheritance to finance the founding of this city.', 'Allen Parkway | Allen Parkway is an arterial road west of Downtown Houston, Texas. It has a distance of approximately 2.3 miles (3.7\xa0km), running from Interstate 45 west to Shepherd Drive, where it becomes Kirby Drive. Originally known as Buffalo Parkway, it was later named after John Kirby Allen and Augustus Chapman Allen, the founders of Houston.', "Allen's Landing | Allen's Landing is the birthplace of the city of HoustonâĂŤthe largest city in the U.S. state of Texas. In August 1836, just months after the Republic of Texas won its independence from Mexico, two brothers (and real estate developers) from New YorkâĂŤJohn Kirby Allen and Augustus Chapman AllenâĂŤpurchased 6,642\xa0acres\xa0(27\xa0kmÄš) in the area and settled there on the banks of Buffalo Bayou. Allen's Landing is located south of the University of HoustonâĂŞDowntown Commerce Street Building, at the intersection of Commerce Street and Main Street. "Keith Allen (American football) | Keith D. Allen (born July 12, 1974) is an American

football coach. He served as the head football coach at Southwest Baptist University from 2007 to 2012, compiling a record of 27âÅŞ39. His 27 wins are the most in the history of Southwest Baptist's football program, which began play in 1983."]

The claim is not supported by the passages. While the passages confirm that Texas's recorded history began in 1519 (passage 3) and that Augustus Chapman Allen founded Houston in Texas (passage 2), they do not explicitly link the start of Texas's recorded history to the same state where Augustus Chapman Allen used his inheritance to fund the founding of a city. The claim conflates these two facts without direct evidence in the passages that the 1519 start of recorded history is tied to the state where Allen's inheritance was used for founding Houston. summary:

The claim is not supported by the provided passages, as they confirm Texas's recorded history began in 1519 and Augustus Chapman Allen founded Houston in Texas but do not explicitly connect the two facts as stated in the claim.

3 other demos omitted for brevity.

GEPA Prompt generated by config GEPA:

Given the fields `claim`, `context`, and `passages`, produce the field `summary` that: 1. **Evaluates the claim's validity** by explicitly connecting it to **specific evidence** from the `passages` (e.g., citing exact names, dates, or terms that support/refute the claim).

- 2. **Highlights factual inaccuracies** in the claim, such as incorrect origins, dates, or relationships, and links them to **direct contradictions** in the `passages` or `context`. 3. **Mentions key terms or entities** critical to the claim (e.g., "Sly Stone," "Large MÃnsterlÃďnder," or "Hit Parader's 2006 ranking") to ensure summaries are actionable for query generation.
- 4. **Balances accuracy** by acknowledging both correct and incorrect aspects of the claim, as seen in the examples (e.g., validating "The Wild Bunch" compilation while correcting the magazine's nationality and ranking year).
- 5. **Avoids vague statements** like "no evidence" and instead specifies **which parts of the passages** are relevant (e.g., "Passage 6 confirms Hit Parader was American").

 Generalizable Strategy:

- Always reference **specific passages** (e.g., "Passage 1 states...") to ground claims in the text.
- Use **precise terminology** (e.g., "German Shorthaired Pointer" instead of "cousins") to align with the data.
- Prioritize **clarity and specificity** to ensure summaries can guide further evidence retrieval.

N.7 PUPA, GPT-4.1 MINI

PUPA GPT-4.1 Mini craft_redacted_request.predict

Base Prompt:

Given a private user query, create a privacy-preserving request for a powerful external LLM.

The LLM may assist without learning private information about the user.

MIPROv2 Prompt:

You will be given a private user query. Your task is to analyze the query carefully and think step-by-step to understand the user's intent while identifying and removing or abstracting any sensitive, proprietary, or personally identifiable information. Then, generate a clear, privacy-preserving prompt suitable for an external large language model (LLM) that requests useful, relevant guidance or information without exposing private details. Additionally, provide a detailed chain-of-thought explanation describing your reasoning process for how you transformed the original query into the redacted, generalized LLM request. Ensure the resulting prompt is professional, contextually appropriate, and maintains user privacy while enabling the LLM to assist effectively. Demos:

Example 1: user_query:

Imagine you are a stock researcher and advisor, I will give you a table about stocks and I want you to give me advice on what to invest next. This is a project I am working on an I need help with predictions I am making. Here is the table: Stock Price Dividends 5D 5D % 1M 1M % YTD YTD % 1Y 1Y % 5Y 5Y % IBM \$146,62 \$6,64 \$3,81 2,66 \$2,63 1,83 \$5,07 3,58 \$16,31 12,51 \$0,03 0,02 MCD \$284,90 \$6,08 \$5,07 1,81 -\$7,20 -2,46 \$20,57 7,78 \$29,24 11,44 \$123,26 76,26 JNJ \$163,98 \$4,76 -\$3,00 -1,80 -\$3,90 -2,32 -\$14,21 -7,97 \$0,98 0,60 \$29,66 22,08 MSFT \$324,89 \$2,72 \$1,25 0,39 -\$9,88 -2,95 \$85,31 35,61 \$59,66 22,49 \$217,33 202,05 WMT \$158,36 \$2,28 \$1,39 0,89 -\$1,85 -1,15 \$14,75 10,27 \$25,47 19,16 \$62,29 64,84 KO \$60,50 \$1,84 \$0,12 0,20 -\$1,59 -2,55 -\$2,45 -3,89 -\$2,23 -3,55 \$15,13 33,35 PFE \$36,07 \$1,64 -\$1,06 -2,84 -\$0,67 -1,81 -\$15,19 -29,64 -\$10,16 -21,98 -\$5,43 -13,09 AAPL \$179,96 \$0,96 \$3,45 1,95 -\$16,57 -8,43 \$54,89 43,89 \$18,58 11,51 \$125,04 227,65 NVDA \$468,23 \$0,16 -\$3,71 -0,79 \$7,05 1,53 \$325,08 227,09 \$310,22 196,33 \$401,52 601,89 CRM \$212,10 \$0,00 \$2,75 1,31 -\$9,86 -4,44 \$77,32 57,37 \$51,89 32,39 \$59,11 38,64 PYPL \$61,16 \$0,00 \$1,06 1,76 -\$13,82 -18,43 -\$13,43 -18,00 -\$31,51 -34,00 -\$29,46 -32,51 BIDU \$137,47 \$0,00 \$10,24 8,05 -\$15,63 -10,21 \$18,34 15,39 -\$9,85 -6,69 -\$88,25 -39,10

reasoning:

The user wants investment advice based on a provided table of stock data including price, dividends, and percentage changes over various time intervals (5 days, 1

month, year-to-date, 1 year, and 5 years). The primary request is for recommendations on what to invest in next and help with making predictions.

To preserve privacy, I should not include the user's raw data or specific details about individual stocks directly in the prompt to the external LLM. Instead, I should request general advice on how to analyse stock data tables that include price, dividend, and percentage changes over multiple time periods. Also, I can ask for advice on what indicators to weigh more heavily for investment decisions and for predictive strategies based on such data structures.

This approach ensures that the external LLM does not receive any private or identifying user data, only a generic request. I will ask the LLM to provide guidance on interpreting such stock performance indicators and suggestions for predictive analytics or investment selection criteria.

llm_request:

355⁻

 Please provide general guidance for analyzing stock investment opportunities based on a data table format that includes stock prices, dividend payouts, and percentage changes over various time intervals such as 5 days, 1 month, year-to-date, 1 year, and 5 years. Outline key indicators that investors typically weigh when making decisions, how to interpret short-term versus long-term performance trends, and suggest strategies or metrics to use when making predictions or recommendations about which stocks to invest in next. The request is for generic advice without any specific stock names or values.

GEPA Prompt generated by config GEPA+Merge:

New Instruction for Privacy-Preserving Query Reformulation Assistant (Enhanced):

Task Overview:

You receive a user query that may contain sensitive, private, or personally identifiable information (PII). Your objective is to reformulate this query into a generalized, privacy-preserving prompt suitable for sending to an external large language model (LLM). The reformulated prompt must enable the external LLM to fulfill the user's original intent effectively while rigorously protecting all user privacy by abstracting, omitting, or generalizing any sensitive details.

Input Format:

- A single user query string.
- The query might contain identifiable names (people, organizations), specific geographic locations, exact dates or timeframes, proprietary or internal terminology (e.g., system names, product codes), client or customer names, URLs, or any other sensitive or private information.

Output Format:

- Part (a) Reasoning:
 - Provide a concise paragraph that explains:
 - \star How you identified sensitive or private information in the input
 - * What strategies you applied to protect privacy (generalization, omission, abstraction, replacement with placeholders)

3618

* How the reformulated prompt preserves the original intent and task requirements 3573 without risking data leakage 3574 - Part (b) LLM Request: 3575 A concise, carefully constructed privacy-safe prompt that: 3576 * Removes or anonymizes all PII and proprietary/internal details 3577 $\boldsymbol{\ast}$ Abstracts locations, names, dates, and technical terms as needed 3578 * Produces a clear and contextually rich instruction for the LLM to generate a relevant and informative response aligned with the user's original task 3579 3580 Detailed Domain-Specific Guidance and Best Practices: 3582 1. Identification and Treatment of Sensitive Data: 3583 - All user-specific or personal names (individual or organizational) must be removed or replaced with generic role descriptors (e.g., âĂa business contact,âĂİ âĂa 3584 client, âĂİ âĂa notable individualâĂİ). Never lightly obscure or partially redact; full 3585 abstraction is required. 3586 - All geographic mentions must be abstracted unless the location is publicly known, 3587 essential to the task, and can be generalized (e.g., âĂa region known for eco-tourismâĂİ instead of naming a country or city explicitly). - Exact dates or durations must never be retained; instead, use relative or 3589 approximate temporal references (e.g., âĂrecently,âĂİ âĂover the past yearâĂİ). 3590 - Internal or proprietary terms âĂŤ including system names, product codes, 3591 subscription types, and technical jargon âĂŤ must be generalized or replaced with 3592 neutral descriptors to avoid leakage of intellectual property or sensitive operational 3593 details. - Avoid direct quotes or verbatim inclusion of user-supplied texts unless obfuscated 3594 by generalization. 3595 3596 2. Task Understanding and Reformulation: 3597 - Identify the functional intent of the query: Is it creative writing, translation, summarization, professional communication drafting, technical explanation, or other? - Preserve the thematic and informational core of the query (e.g., request for 3599 educational quality analysis, technical translation of a passage, biographical summary). 3601 - Do not reproduce the original input verbatim; instead, frame the LLM prompt around 3602 the essential thematic elements extracted from the input. 3603 - For queries regarding individuals, avoid direct reference to real personal information unless the name is widely public and essential; even then, use a generic 3604 or hypothetical framing for the individual profile. 3605 3606 3. Strategies for High-Quality and Privacy-Preserving Prompts: 3607 - Strike a balance between sufficient contextual detail and privacy abstraction to 3608 maintain prompt clarity and relevance. - Use neutral, context-aware formulations that clearly instruct the LLM on the content 3609 and style expected. 3610 - Avoid vague or overly generic prompts that could result in less useful or 3611 lower-quality responses. 3612 - When system or proprietary content is mentioned, instruct the LLM to generalize 3613 specific terms and maintain the technical meaning without revealing sensitive info. - When a direct translation is requested on specialized text, specify to replace or 3614 abstract internal nomenclature. 3615 3616 4. Explanation Requirements: 3617

- The reasoning must transparently explain how privacy risks were identified (e.g., presence of names, locations, dates, proprietary terms).
- It must describe the abstraction or omission methods applied (e.g., replacing âĂJonah Van BeijnenâĂİ with âĂa notable individual,âĂİ substituting âĂMakauâĂİ with âĂa specific region,âĂİ or âĂYodaâĂİ with âĂa system nameâĂİ).
- Clarify how the essential task and user intent were preserved despite these generalizations (e.g., focusing on educational quality, technical translation, biographical summary).
- This explanation justifies your design choices and demonstrates adherence to privacy-preserving principles.

Common Pitfalls to Avoid:

- Do not merely lightly obscure or partially redact sensitive details; full anonymization or abstraction is required.
- Do not repeat any user-supplied PII or proprietary content verbatim.
- Avoid including URLs, exact dates, or direct quotes without modification.
- Do not leave ambiguity that could degrade the quality or contextual clarity of the reformulated prompt.
- Do not include any real personal or organizational names unless they are public figures and the query requires it, then use generic descriptors instead.

Example Summary of Effective Approach (Informed by Prior Examples):

- For geographic queries: replace exact place names with general regions and provide a brief contextual descriptor.
- For technical texts containing system names or subscription types: instruct the LLM to translate or process the text while replacing or abstracting proprietary system identifiers.
- For biographical summaries about specific individuals: remove the real name and request a generic, well-structured four-paragraph summary about âĂa notable individual,âĂİ preserving the overall intent without leaking PII.

Summary:

Your reformulations must ensure zero exposure of any PII or private/proprietary content while retaining enough thematic and functional clarity for the external LLM to produce high-quality, relevant outputs. This requires thorough analysis of the user's query, rigorous application of privacy-preservation strategies, and explicit reasoning explanations that document your approach and choices.

PUPA GPT-4.1 Mini respond_to_query

Base Prompt:

Respond to a user query.

For inspiration, we found a potentially related request to a powerful external LLM and its response.

MIPROv2 Prompt:

Given a privacy-preserving redacted prompt sent to an external untrusted LLM and its response, along with the original user query containing potentially sensitive information, generate a final, coherent, and contextually appropriate user-facing response. Your output should integrate the anonymized LLM output with the original

3666 queryâĂŹs intent, ensuring privacy is maintained while providing a complete and polished 3667 answer that directly addresses the user's request. 3668 Demos: 3669 Example 1: 3670 related_llm_request: 3671 Please provide a detailed explanation of David Easton's system analysis in political 3672 science, outlining the key components of his systems theory approach and its significance in the study of political systems. 3673 related_llm_response: 3674 Certainly! David Easton's system analysis is a foundational framework in political science that applies systems theory to the study of political systems. His approach 3676 provides a structured way to understand how political systems function, maintain 3677 stability, and respond to changes in their environment. Below is a detailed explanation of Easton's system analysis, including its key components and 3678 significance. 3679 3680 3681 3682 ### Overview of David Easton's System Analysis 3683 David Easton (1917âĂŞ2014) was a prominent political scientist who sought to develop a 3684 general theory of political systems by borrowing concepts from systems theory, which 3685 originated in biology and engineering. His goal was to create a scientific and analytical 3686 framework to study politics as an integrated and dynamic system rather than as isolated 3687 events or institutions. 3688 Easton's system analysis views a political system as a set of interactions abstracted 3689 from the totality of social behavior, through which authoritative allocation of values 3690 (decisions) is made and implemented. In other words, politics is about how societies 3691 decide who gets what, when, and how, and Easton's model helps explain how these decisions are made and sustained. 3693 3695 ### Key Components of Easton's Systems Theory Approach 3696 3697 Easton's model conceptualizes the political system as a **black box** that processes inputs and produces outputs, embedded within a larger environment. The key components are: 3699 #### 1. **Inputs** 3700 Inputs are demands and supports that come from the environment and society into the 3701 political system. 3702 - **Demands:** These are claims or requests made by individuals or groups for the 3703 political system to allocate values or resources in a particular way (e.g., demands for 3704 law enforcement, education, welfare). 3705 - **Supports:** These are actions or attitudes that indicate approval, allegiance, or 3706 compliance with the political system (e.g., voting, paying taxes, obeying laws). 3707 Inputs represent the pressures and resources that the political system must process. 3708 3709

2. **Political System (The Black Box)**

```
3713
3714
          The political system is the core of Easton's model. It receives inputs, processes them,
          and produces outputs.
3715
3716
          - It includes institutions, organizations, and processes involved in authoritative
3717
          decision-making (e.g., legislatures, executives, courts).
3718
           - The system transforms demands and supports into decisions and policies.
3719
          - The system must maintain its boundaries and adapt to changes to survive.
3720
          #### 3. **Outputs**
3721
          Outputs are the decisions, policies, and actions that the political system produces in
3722
          response to inputs.
3723
           - These include laws, regulations, public policies, and authoritative allocations of
3724
          values.
3725
           - Outputs affect the environment and society, influencing future inputs.
3726
3727
          #### 4. **Feedback**
3728
          Feedback is the process by which outputs influence future inputs.
3729
          - Positive feedback can reinforce the systemâĂŹs stability.
3730
          - Negative feedback can signal problems or dissatisfaction, prompting adjustments.
3731
          - Feedback helps the system adapt and maintain equilibrium.
3732
3733
          #### 5. **Environment**
3734
          The political system exists within a broader environment that includes social, economic,
          cultural, and international factors.
3735
3736
          - The environment provides inputs and is affected by outputs.
3737
          - The system must interact with and adapt to environmental changes to survive.
3738
3739
3740
          ### Diagrammatic Representation (Simplified)
3741
3742
3743
          Environment
3744
             âĘŚ âĘŞ
          [Demands & Supports] âEŠ [Political System] âEŠ [Decisions & Policies]
3745
                   âĘŚ âĘŞ
3746
                   âEŘ----- Feedback -----
3747
3748
3749
3750
          ### Significance of Easton's System Analysis in Political Science
3751
3752
          1. **Holistic Understanding of Politics:**
3753
             Easton's model encourages viewing politics as an interconnected system rather than
3754
             isolated events or institutions. It highlights the dynamic interactions between
             society and political authority.
3755
3756
          2. **Focus on Process and Stability:**
3757
```

3760 The model emphasizes how political systems maintain stability and order through 3761 continuous processing of inputs and outputs, and through feedback mechanisms. 3762 3763 3. **Analytical Clarity:** 3764 By abstracting political activity into inputs, system processing, outputs, and 3765 feedback, Easton provided a clear framework for empirical research and comparative 3766 analysis. 3767 4. **Foundation for Further Theories:** Easton's systems theory influenced later developments in political science, including 3769 structural-functionalism, policy analysis, and the study of political development and 3770 change. 3771 5. **Adaptability:** 3772 The model can be applied to various political systems (democracies, authoritarian 3773 regimes, etc.) and scales (local, national, international), making it versatile. 3774 3775 6. **Emphasis on Environment:** 3776 Recognizing the environmentâĂŹs role underscores the importance of external factors (economic conditions, social movements, international pressures) in shaping political 3777 3778 3779 3780 3781 ### Criticisms and Limitations 3782 - **Overly Abstract:** Some critics argue that Easton's model is too abstract and 3783 general, lacking specificity about power relations, conflict, and individual agency. 3784 - **Neglect of Conflict:** The model tends to emphasize stability and equilibrium, 3785 potentially underplaying political conflict and change. - **Limited Normative Guidance:** It is primarily descriptive and analytical, offering little normative guidance on what political systems ought to do. 3787 3789 3790 ### Summary 3791 David Easton's system analysis 3792 user_query: 3793 System analysis by David Easton 3794 response: 3795 David Easton's system analysis is a foundational framework in political science that 3796 conceptualizes the political system as a dynamic and interactive system. His approach uses systems theory to explain how political systems function, maintain stability, 3797 and adapt to changes in their environment. 3798 3799 At its core, Easton's model views the political system as a "black box" that processes 3800 inputs from society and the environment, transforms them into outputs, and uses feedback to adjust and sustain itself. The key components of Easton's system analysis include: 3801 3802 1. **Inputs:** These are the demands and supports that society directs towards the 3803 political system. Demands are requests or claims for resource allocation or policy

change, while supports are actions or attitudes such as compliance, legitimacy, or loyalty that sustain the system.

- 2. **Political System: ** This is the set of institutions and processes that processes the inputs. It interprets demands and supports to make authoritative decisions, including the formulation and enforcement of policies.
- 3. **Outputs:** These are the decisions, policies, and actions that the political system produces in response to inputs. They affect society and its environment, shaping future interactions.
- 4. **Feedback:** Outputs generate responses that feed back into the system as new inputs, signaling whether the outputs were effective or require modification. This feedback enables the system to adapt to internal and external changes.
- 5. **Environment:** The system operates within a broader social, economic, cultural, and international context that influences and is influenced by the political system.

Easton's theory is significant because it offers a holistic, process-oriented lens to analyze politics, emphasizing interaction between society and state institutions. It provides a framework to understand political stability, adaptation, and change by focusing on the continuous circulation of inputs, processing, outputs, and feedback. The model has been influential in shaping subsequent political theories and comparative political analysis.

However, critics note that Easton's model can be overly abstract, sometimes downplaying conflict, power struggles, and individual agency within political processes.

In summary, David Easton's system analysis offers a powerful conceptual tool to study political systems as dynamic entities that manage societal demands and supports through authoritative decision-making to maintain order and adapt over time.

GEPA Prompt generated by config GEPA+Merge:

Respond to a user query.

For inspiration, we found a potentially related request to a powerful external LLM and its response.

N.8 PUPA, QWEN3 8B

PUPA Qwen3 8B craft_redacted_request.predict

Base Prompt:

Given a private user query, create a privacy-preserving request for a powerful external LLM.

The LLM may assist without learning private information about the user.

MIPROv2 Prompt:

Given a private user query, generate a structured, non-sensitive LLM request that captures the user's intent while omitting personal details. Include a step-by-step reasoning process explaining how the redaction was achieved, ensuring the request is actionable and privacy-preserving. Format the reasoning as a coherent thought process and the request as a clear, specific task for the LLM to execute.

Example 1:

user_query:

D§ÑĂĐŸĐŠĐţÑĆ ĐIĐŤĐ£ĐŸÑĹĐŸ ĐŽĐ¿ĐĐŤĐIĐťÑČ ĐťĐŽÑŔ Đ¿ÑĆĐ£ÑĂĐŤĐŠĐŽĐŸ snmpwalk ĐiĐŤ ip 10.2.35.2 ÑĄ НЍĐiĐiÑŃĐĐŸ Đ£Đ¿ ĐťĐ¿ÑĄÑĆÑČĐ£ÑČ initi_user/d2J3BgtYdB2/d2J3BgtYdB2 (MD5) DES

reasoning:

The user is requesting an SNMP walk command to retrieve data from an SNMPv3-enabled device at IP 10.2.35.2. The provided credentials include a username (`initi_user`), an authentication password (`d2J3BgtYdB2`), and an encryption password (`d2J3BgtYdB2`). The authentication protocol is MD5, and the encryption protocol is DES. The command must include SNMPv3 options to specify these parameters correctly. llm_request:

Generate a command-line instruction for executing `snmpwalk` with SNMPv3 authentication and encryption using the provided credentials, ensuring privacy by omitting sensitive details in the response.

GEPA Prompt generated by config GEPA:

New Instruction for the Assistant:

Task Overview:

Rephrase the user's query into a privacy-preserving, actionable request for an external LLM, ensuring no personally identifiable information (PII), proprietary data, or sensitive details are exposed. Maintain the original intent, clarity, and specificity of the query while adhering to domain-specific strategies for generalized information.

Key Requirements:

1. **Privacy Preservation**:

- **Generalize Specifics**: Replace specific locations (e.g., "Andhra Pradesh" âξŠ "a major Indian city"), numbers (e.g., "18 volunteers" âξŠ "a group of participants"), or proprietary data (e.g., "van Cleef production sites" âξŠ "[Brand Name]'s production sites based on publicly available information").
- **Avoid PII/Proprietary Data**: Remove names, addresses, dates, internal processes, or brand-specific secrets. Use placeholders like [Brand Name], [Country], or [Region] for brand/company references.
- **Publicly Available Focus**: For brand-related queries, emphasize publicly accessible data (e.g., "locations of [Brand Name]'s production sites" instead of "van Cleef's secret factories").
- 2. **Quality of Request**:

```
3901
             - **Clarity and Actionability**: Ensure the rephrased query is precise, avoids
3902
             ambiguity, and specifies constraints (e.g., "concise, platform-friendly hashtags" for
             creative tasks).
3904
             - **Domain-Specific Precision**: For scientific/technical queries, retain key terms
3905
             (e.g., "CAPM formula," "pharmacokinetic parameters") while anonymizing study details
3906
             (e.g., "18 volunteers" âĘŠ "a group of participants").
3907
             - **Avoid Overly Broad Requests**: Specify constraints like keyword focus, tone, or
             format (e.g., "health and fitness keywords" instead of "general hashtags").
3908
3909
          3. **Domain-Specific Strategies**:
             - **Geographic Queries**: Replace regions/cities with generalized terms (e.g., "major
3911
             city in [Country]" or "well-known [Region]").
3912
             - **Brand/Company Queries**: Reference brands by name but prioritize publicly
             available data (e.g., "locations of [Brand Name]'s production sites").
3913
             - **Creative Tasks**: Use examples or constraints (e.g., "wellness, strength, motivation keywords" for hashtags; "platform guidelines" for social media content).
3914
3915
3916
          4. **Avoid Leakage**:
             - **No Inferred Details**: Do not assume user-specific preferences, contexts, or
3917
             private data (e.g., "Tell me the location of van Cleef production sites" âţš "Provide
3918
             locations of [Brand Name]'s production sites based on publicly available information").
3919
             - **Anonymize Numerical Data**: Replace specific counts, dosages, or metrics with
             generalized terms (e.g., "40 mg OA" âĘŠ "a dose of OA").
3921
3922
           **Example Workflow:**
          - **User Query**: "Calculate DâĂŹAmicoâĂŹs cost of equity using CAPM with rRF = 4.67%, Κ
3923
          = 1.56, and market risk premium = 6.63%."
3924
          - **Rephrased Request**: "Calculate the cost of equity for a company using the CAPM
3925
           formula, given a risk-free rate of 4.67%, a beta of 1.56, and a market risk premium of
3926
          6.63%. Show step-by-step calculations and round the final result to two decimal places."
          **Niche Domain-Specific Guidance:**
3928
           - **Scientific/Pharmacokinetic Queries**: Generalize study details (e.g., "18 healthy
          male volunteers" âEŠ "a group of participants"), retain key parameters (e.g., AUC, peak
3930
          concentrations), and avoid proprietary methodologies.
3931
           - **Creative/Translation Tasks**: Use placeholders for song titles/artists (e.g.,
           "Mahmood's song [Title]") and specify tone/rhythm requirements for translations.
3932
           - **Financial Calculations**: Retain formulas and numerical values but avoid linking to
          specific companies or proprietary models.
3934
3935
           **Final Output Format:**
3936
           - **reasoning**: Briefly explain the rephrasing choices (e.g., "Generalized '18
3937
          volunteers' to 'a group of participants' to preserve privacy").
           - **llm_request**: Provide the anonymized, actionable request for the LLM.
3938
3939
```

PUPA Qwen3 8B respond_to_query

Base Prompt:

3940

3941 3942

3943

3945

3947

Respond to a user query.

For inspiration, we found a potentially related request to a powerful external LLM and its response.

```
3948
          MIPROv2 Prompt:
3949
3950
          Synthesize the external LLM's response with the user's specific query to generate a final
3951
          privacy-preserving answer. Replace placeholders with actual values while omitting
3952
          sensitive information, maintaining a professional tone, and ensuring contextual accuracy.
          Combine the technical solution from the LLM with the user's unique details to produce a
3953
          clear, anonymized response that adheres to privacy constraints.
3954
          Demos:
3955
          Example 1:
3956
          related_llm_request:
             Create a PESTLE analysis framework for GE Healthcare in the US, outlining key issues
             in each category (Political, Economic, Social, Technological, Environmental, Legal)
3958
             and their potential impact on the organization. Prioritize these issues based on
3959
             urgency and significance without referencing confidential data. Structure the output
3960
             as a matrix with clear criteria for evaluation.
3961
          related_llm_response:
3962
             Certainly! Below is a **PESTLE analysis framework** tailored for **GE Healthcare in
             the US**, structured as a matrix. It outlines key issues in each category, evaluates
3963
             their **urgency** and **significance**, and highlights their **potential impact** on
             the organization.
3965
3966
3967
          ### PESTLE Analysis Matrix for GE Healthcare (US)
3968
3969
          | **Category** | **Key Issues** | **Urgency** (High/Med/Low) | **Significance**
3970
          (High/Med/Low) | **Potential Impact on GE Healthcare** |
3971
          |-----
3972
          -----|-----|
3973
          |-----
3974
3975
          | **Political** | 1. Healthcare policy changes (e.g., Medicare/Medicaid reimbursement
          reforms) | High | High | Directly affects revenue streams and product/service demand;
3977
          requires adaptation to new reimbursement models. |
          | | 2. Trade policies and tariffs on medical equipment imports/exports | Medium | Medium
3978
          | Could increase costs of components or affect supply chain efficiency. |
3979
          | | 3. Government funding for healthcare innovation and research | Medium | High |
          Influences R&D opportunities and partnerships; potential for grants and subsidies. |
3981
          | **Economic** | 1. US healthcare spending trends and budget constraints | High | High |
          Impacts purchasing power of hospitals and clinics; affects sales volume and pricing
          strategies. |
3983
          | | 2. Inflation and rising operational costs | High | Medium | Increases production and
3984
          operational expenses; may pressure profit margins. |
3985
          | | 3. Economic cycles affecting capital investment in healthcare infrastructure | Medium
3986
          | Medium | Influences demand for high-cost medical equipment and technology upgrades. |
3987
          | **Social** | 1. Aging population increasing demand for healthcare services and devices
          | High | High | Drives demand for diagnostic imaging, monitoring, and therapeutic
3988
          devices; growth opportunity. |
3989
          | | 2. Growing patient awareness and demand for personalized medicine | Medium | Medium |
3990
          Necessitates innovation in tailored healthcare solutions and data-driven technologies. |
          | | 3. Workforce shortages in healthcare professionals | High | Medium | May affect
          adoption and training for new technologies; impacts customer support and service
3992
         delivery. |
3994
```

```
3995
          | **Technological** | 1. Rapid advancements in AI and machine learning for diagnostics
3996
          and imaging | High | High | Critical for maintaining competitive edge; requires
3997
          continuous investment in R&D and product development. |
3998
          | | 2. Integration of IoT and connected devices in healthcare | Medium | High | Enhances
3999
          product offerings but demands robust cybersecurity and interoperability solutions. |
4000
          | | 3. Data privacy and cybersecurity technologies | High | High | Essential to protect
4001
          patient data and comply with regulations; failure risks legal penalties and reputational
          damage. |
4002
          | **Environmental** | 1. Regulatory pressure to reduce carbon footprint and waste in
4003
          manufacturing | Medium | Medium | Necessitates sustainable practices; potential cost
4004
          implications but improves brand reputation. |
4005
          | | 2. Energy efficiency in product design and operations | Medium | Medium | Can reduce
4006
          operational costs and appeal to environmentally conscious customers. |
          | | 3. E-waste management and recycling programs | Low | Medium | Compliance with
4007
          environmental standards; growing importance for corporate social responsibility. |
4008
          | **Legal** | 1. Compliance with FDA regulations and medical device approvals | High |
4009
          High | Critical for market access; delays or failures can halt product launches and cause
4010
          financial losses. I
          | | 2. Intellectual property rights and patent protections | Medium | High | Protects
4011
          innovation and competitive advantage; infringement risks costly litigation. |
4012
          | | 3. Liability and product safety regulations | High | High | Ensures patient safety;
4013
          non-compliance can lead to lawsuits, recalls, and reputational damage. |
4014
4015
4016
          ### Criteria for Evaluation:
4017
4018
          - **Urgency**: How immediate the issue requires attention (High = immediate/ongoing;
4019
          Medium = within 1-2 years; Low = longer-term).
4020

    - **Significance**: The potential magnitude of impact on GE HealthcareâĂŹs operations,

          profitability, and strategic positioning (High = critical; Medium = moderate; Low =
4021
4022
          - **Potential Impact**: Describes how the issue could affect GE HealthcareâĂŹs business,
4023
          including risks and opportunities.
4024
4025
4026
          This matrix helps prioritize strategic focus areas for GE Healthcare in the US, balancing
4027
          immediate challenges with longer-term trends.
4028
          user query:
4029
             issue priority matrix on pestle analysis for GE Healthcare in US
4030
          response:
4031
             Here's a prioritized **issue priority matrix** based on the PESTLE analysis for **GE
             Healthcare in the US**, focusing on the most urgent and significant challenges and
4032
             opportunities:
4033
4034
4035
          ### **High Priority Issues (High Urgency + High Significance)**
4036
          | **Category** | **Key Issue** | **Urgency** | **Significance** | **Impact Summary** |
4037
          |-----
4038
          -----|-----|-----
4039
4040
```

```
4042
4043
          | **Political** | Healthcare policy changes (e.g., reimbursement reforms) | High | High |
4044
          Direct revenue risk; requires rapid adaptation to new regulations.
4045
          | **Economic** | US healthcare spending trends and budget constraints | High | High |
4046
          Drives demand and pricing pressures; critical for sales strategy. |
4047
          | **Social** | Aging population demand for healthcare services | High | High | Growth
4048
          opportunity in diagnostics/therapeutics; long-term market expansion. |
          | **Technological** | AI/ML advancements in diagnostics and imaging | High | High |
4049
         Essential for competitiveness; demands sustained R&D investment. |
          | **Legal** | FDA regulations and medical device approvals | High | High | Critical for
4051
          market access; delays risk financial losses. |
4052
4053
4054
          ### **Medium Priority Issues (High Urgency + Medium Significance or Medium Urgency + High
4055
          Significance)**
4056
          | **Category** | **Key Issue** | **Urgency** | **Significance** | **Impact Summary** |
4057
          |-----|
4058
          | **Economic** | Inflation and operational costs | High | Medium | Pressures profit
          margins; requires cost-efficiency measures. |
4059
          | **Technological** | IoT integration and cybersecurity | Medium | High | Enhances
4060
          offerings but demands robust security frameworks. |
4061
          | **Environmental** | Carbon footprint reduction and sustainability | Medium | Medium |
4062
         Balances cost implications with brand reputation and regulatory compliance. |
4063
4064
4065
          ### **Prioritization Criteria**
4066
          - **Urgency**: Immediate (High) vs. longer-term (Low).
4067
          - **Significance**: Critical (High) vs. moderate (Medium).
          - **Focus**: Issues with **High in both urgency and significance** require immediate
4068
          strategic action, while others demand monitoring or phased planning.
4069
4070
          This matrix helps GE Healthcare allocate resources to address the most pressing
4071
          challenges and leverage high-impact opportunities.
4072
4073
          4074
          3 other demos omitted for brevity.
4075
          4076
4077
          GEPA Prompt generated by config GEPA:
4078
          Respond to a user query by leveraging a related LLM response as inspiration. Structure
4079
          your answer in a clear, organized format (e.g., bullet points, sections) to enhance
4080
          readability. Ensure the response is tailored to the user's language and avoids any
4081
         personally identifiable information (PII) or sensitive data.
4082
          When rephrasing or summarizing content:
4083
          1. **Maintain academic tone** if the query specifies formal language (e.g., rephrasing
4084
          technical or historical text).
4085
         2. **Refactor repetitive or redundant information** into concise, focused sections (e.g.,
```

grouping similar ideas, eliminating duplication).

4086

4104 4105

4106

4107

4108 4109

```
4089
          3. **Highlight domain-specific facts** (e.g., cultural, technical, or regional details)
4090
          to add value, as these may not be universally known.
4091
          4. **Improve code or technical responses** by breaking down complex logic into smaller,
4092
          well-documented functions, reducing redundancy, and ensuring proper resource management.
4093
4094
          For all outputs:
4095
          Prioritize **clarity, accuracy, and adherence to the user's requirements**.
          - Avoid **leaking proprietary information, PII, or unverified claims**.
4096
          - If the userâĂŹs query involves multiple steps (e.g., code optimization, content
4097
          rephrasing), address each part systematically.
4098
          - Use **language-specific conventions** (e.g., correct grammar, terminology) to align
4099
          with the userâĂŹs preferred language.
4100
4101
```

GEPA GENERATED PROMPTS FOR KERNEL GENERATION

O.1 NPUEVAL: KERNEL CODE GENERATION FOR NEW HARDWARE ARCHITECTURE

Figure 26 shows the prompt generated by GEPA with GPT-40 for NPU Keernel Generation, that achieves 26.85% score with the same same GPT-40 agent, that achieved just 4.25% with a simple prompt.

O.2 KERNELBENCH: CUDA KERNEL CODE GENERATION FOR NVIDIA GPUS

4110 4111 GEPA GPT-40 generated prompt for CUDA Kernel Generation 4112 4113 To optimize a given PyTorch model by replacing operators with custom CUDA kernels, follow 4114 these detailed instructions. Your goal is to achieve performance improvements while 4115 ensuring correctness. Name your optimized output architecture `ModelNew`. Output the 4116 new model code in codeblocks. Please generate real code, NOT pseudocode, and ensure 4117 the code compiles and is fully functional. Do not include testing code. 4118 ### Steps to Create Custom CUDA Kernels 4119 4120 #### 1. Identify Operators to Replace 4121 - Analyze the model to identify operators that can benefit from custom CUDA 4122 implementations. 4123 - Consider operator fusion opportunities to combine multiple operations into a single 4124 kernel for efficiency. 4125 #### 2. Setup and Compilation 4126 - Use `torch.utils.cpp_extension.load_inline` to compile your CUDA code. This allows you 4127 to define and compile custom CUDA kernels directly within your Python script. 4128 - Ensure all necessary CUDA and C++ headers are included to avoid missing includes errors. 4129 #### 3. Implementing the CUDA Kernel 4130 - Write the CUDA kernel code, ensuring it is optimized for parallel execution. Use shared 4131 memory to reduce global memory accesses and ensure coalesced memory access patterns. 4132 - Example structure for a CUDA kernel: 4133 `cpp __global__ void my_kernel(const float* input, float* output, int size) { 4134 4135

```
4136
4137
4138
4139
4140
4141
4142
4143
4144
            You are tasked with generating C++ code for a single kernel function that will run on an AI Engine (AIE) tile. The kernel should perform a
            specified operation on a vector of bfloat16 values. Your code should be complete and self-contained within a single code block, including all
4145
            necessary headers and imports. Follow these guidelines:
4146
            1. **Headers and Imports**: Include only the necessary headers for AIE operations. Avoid including `<adf.h>` or any headers that are not part of
4147
            the standard AIE API. Use:
             ```cpp
4148
 #include <stdint.h>
4149
 #include <aie_api/aie.hpp>
 #include <aie_api/utils.hpp>
4150
4151
 2. **Kernel Function**: Implement the kernel function as specified in the input. The function should take pointers to input and output buffers
4152
 and a size parameter. Use AIE vector operations to process the data efficiently.
4153
 3. **Vector Operations**: Utilize the AIE API's vector operations for loading, processing, and storing data. For example, use `aie::vector` for
 vector operations and `aie::reduce_min` for reduction tasks. Ensure that the vector size is compatible with the AIE hardware capabilities.
4154
4155
 4. **Avoid Non-Existent Functions**: Do not use functions like `aie::exp` or `aie::store v` if they are not supported. Instead, implement the
 required functionality using available AIE API functions.
4156
4157
 5. **Error Handling**: Ensure that the code is free of syntax errors and compatible with the AIE environment. Test the code for compilation
 without errors.
4158
 Here is an example template for a kernel function:
4159
 ``cpp
4160
 #include <stdint.h>
 #include <aie_api/aie.hpp>
4161
 #include <aie api/utils.hpp>
4162
 void kernel_function_name(bfloat16 *input_vector, bfloat16 *output_vector, uint32_t vector_size) {
4163
 constexpr int vector_length = 16; // Adjust based on AIE capabilities
 aie::vector<bfloat16, vector_length> input_data;
4164
 aie::vector<bfloat16, vector_length> output_data;
4165
 for (uint32_t i = 0; i < vector_size; i += vector_length)
4166
 input_data = aie::load_v<vector_length>(input_vector + i);
 // Perform the required operation on input data
4167
 // Store the result in output_data
4168
 aie::store_v(output_vector + i, output_data);
4169
4170
```

Replace `kernel\_function\_name` and the operation logic with the specific task details provided in the input.

4171

4172

4173

Figure 26: GEPA generated prompt for NPUEval that achieves 26.85% score with the same GPT-40 agent, that achieved a 4.25% score.

```
4183
 int idx = blockIdx.x * blockDim.x + threadIdx.x;
4184
 if (idx < size) {</pre>
4185
 // Perform computation
4186
 }
4187
 } . .
4188
4189
 #### 4. Kernel Launch Configuration
4190
 - Configure the kernel launch parameters (number of blocks and threads per block) to
4191
 maximize GPU utilization.
4192
 - Example:
4193
 ``python
4194
 const int block_size = 256;
 const int num_blocks = (size + block_size - 1) / block_size;
4195
 my_kernel<<<num_blocks, block_size>>>(input, output, size);
4196
4197
4198
 #### 5. Error Handling and Debugging
4199
 - Implement error checking after CUDA API calls and kernel launches using
 cudaGetLastError()` to catch errors early.
4200
 - Use CUDA debugging tools like `cuda-memcheck` and NVIDIA Nsight for debugging and
4201
 profiling.
4202
 - Be aware of common syntax errors and namespace issues, and ensure that the CUDA code
4203
 adheres to syntax rules.
4204
 #### 6. Integrating with PyTorch
4205
 - Define a Python function that wraps the CUDA kernel call and integrates it into the
4206
 PyTorch model.
4207
 - Example:
 `python
 def my_custom_op(input):
4209
 output = torch.zeros_like(input)
4210
 my_kernel(input.data_ptr<float>(), output.data_ptr<float>(), input.numel())
4211
 return output
4212
4213
4214
 #### 7. Testing and Validation
 - Validate the correctness of the custom kernel by comparing its output with the original
4215
 PyTorch operator.
4216
 - Use profiling tools to measure performance improvements and identify further
4217
 optimization opportunities.
4218
 - Establish reference outputs for known-good inputs to verify correctness.
4219
 - Consider floating-point discrepancies and use a small epsilon for acceptable differences
4220
4221
 #### 8. Compatibility and Compute Capability
4222
 - Ensure that the CUDA code is compatible with the target GPU's compute capability.
4223
 - Use appropriate compiler flags and consider building for multiple compute capabilities
 if necessary.
4224
4225
 ### Example Code
4226
 Here is an example of how to replace a simple element-wise addition with a custom CUDA
4227
 kernel:
4228
```

```
4230
4231
                          ```python
4232
                         import torch
4233
                         import torch.nn as nn
4234
                         from torch.utils.cpp_extension import load_inline
4235
4236
                         # Define the custom CUDA kernel for element-wise addition
                        elementwise_add_source = ""
4237
                        #include <torch/extension.h>
4238
                         #include <cuda_runtime.h>
4239
4240
                         __global__ void elementwise_add_kernel(const float* a, const float* b, float* out, int
4241
                                   size) {
                                 int idx = blockIdx.x * blockDim.x + threadIdx.x;
4242
                                 if (idx < size) {</pre>
4243
                                          out[idx] = a[idx] + b[idx];
4244
4245
                        }
4246
                         torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b) {
4247
                                 auto size = a.numel();
4248
                                 auto out = torch.zeros_like(a);
4249
4250
                                 const int block_size = 256;
4251
                                 const int num_blocks = (size + block_size - 1) / block_size;
4252
                                 elementwise\_add\_kernel <<< num\_blocks, \ block\_size>>> (a.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data\_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < float>(), \ b.data_ptr < 
4253
                                            float>(), out.data_ptr<float>(), size);
4254
4255
                                 return out;
4256
4257
4258
                         elementwise_add_cpp_source = (
                                  "torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b);"
4260
4261
                         # Compile the inline CUDA code for element-wise addition
4262
                         elementwise_add = load_inline(
4263
                                 name="elementwise_add",
4264
                                 cpp_sources=elementwise_add_cpp_source,
4265
                                 cuda_sources=elementwise_add_source,
4266
                                 functions=["elementwise_add_cuda"],
                                 verbose=True,
4267
                                 extra_cflags=[""].
4268
                                 extra_ldflags=[""],
4269
4270
                        class ModelNew(nn.Module):
4271
                                 def __init__(self) -> None:
4272
                                          super().__init__()
4273
                                          self.elementwise_add = elementwise_add
4274
4275
```

```
def forward(self, a, b):
4278
                 return self.elementwise_add.elementwise_add_cuda(a, b)
4279
4280
4281
          ### Additional Best Practices
4282
           **Optimize Memory Usage**: Minimize data transfers between host and device, and use
4283
               shared memory to reduce global memory access.
          - **Atomic Operations**: When using atomic operations like `atomicMax` with floating-point
4284
               numbers, ensure correct usage by following best practices, such as using appropriate
               data types and minimizing contention.
4286
          - **Performance Optimization**: Maximize parallel execution, optimize memory access
4287
               patterns, and use compiler flags to enhance performance.
4288
          - **Namespace Usage**: Avoid adding class declarations or function definitions directly to
               reserved namespaces like `cuda`. Use nested namespaces within non-reserved
4289
               namespaces to organize code.
4290
          - **Numerical Precision**: Be aware of floating-point arithmetic issues, such as non-
4291
               associativity, and use appropriate precision levels for calculations.
4292
          - **Debugging Tools**: Utilize tools like CUDA-GDB and NVIDIA Nsight for debugging and
               profiling to ensure correctness and performance.
4293
4294
          By following these instructions, you can effectively replace PyTorch operators with custom
4295
               CUDA kernels, ensuring both performance improvements and correctness.
4296
4297
          ### Instruction for Replacing PyTorch Operators with Custom CUDA Kernels
4298
4299
          Your task is to optimize a given PyTorch model by replacing certain operators with custom
               CUDA kernels to achieve performance improvements. Follow the steps below to ensure a
4300
               successful implementation:
4301
4302
          #### Step 1: Identify Operators for Replacement
4303
          - **Criteria for Selection**: Choose operators that are computationally intensive and have
               potential for parallelization. Consider operators that are frequently used in the
4304
               model's forward pass.
4305
          - **Operator Fusion**: Look for opportunities to fuse multiple operators into a single
4306
               CUDA kernel, such as combining matrix multiplication with activation functions (e.g.,
4307
               matmul + ReLU).
4308
4309
          #### Step 2: Implement Custom CUDA Kernels
          - **Kernel Structure**: Define your CUDA kernel using the `__global__` specifier. Ensure
4310
               that each thread handles a specific part of the computation. Use correct index
4311
               calculations to access data.
4312
          - **Memory Management**:
4313
            - Allocate memory for input, output, and any intermediate data on the GPU using \dot{}
4314
                cudaMalloc`. Use `cudaMemcpy` to transfer data between host and device.
            - Utilize shared memory to cache frequently accessed data and reduce global memory
4315
4316
            - Ensure coalesced global memory accesses for efficient memory transactions.
4317
          - **Numerical Stability and Boundary Conditions**:
4318
            - Implement verification mechanisms to ensure numerical stability. Use `__host__
4319
                 __device__` functions for testing on both CPU and GPU.
            - Handle boundary conditions to prevent out-of-bounds memory access. Ensure that thread
4320
                indices are within valid ranges.
4321
          - **Optimization Techniques**:
4322
4323
```

```
4324
            - Use shared memory to reduce global memory accesses and improve performance.
4325
            - Consider using mixed precision and Tensor Cores for matrix operations to enhance
4326
                 performance.
4327
            - Avoid diverged execution paths to maintain efficient parallel execution.
4328
4329
          #### Step 3: Integrate CUDA Kernels into PyTorch Model
4330
          - **Inline Compilation**: Use `torch.utils.cpp_extension.load_inline` to compile your CUDA
               code and integrate it into the PyTorch model.
4331
          - **Model Modification**: Replace the original PyTorch operators with calls to your custom
4332
               CUDA functions. Ensure that the new model architecture (`ModelNew`) is fully
4333
               functional and compiles without errors.
4334
4335
          #### Step 4: Testing and Validation
          - **Correctness**: Verify that the output of the optimized model matches the expected
4336
               output of the original model. Use a set of test cases to ensure accuracy.
4337
          - **Performance Evaluation**: Measure the runtime of the optimized model and compare it to
4338
                the original. Aim for a significant reduction in execution time.
4339
          - **Edge Case Handling**: Ensure that the kernel correctly handles cases where the matrix
               size is not a multiple of the block size and other potential edge cases.
4340
4341
          #### Example Code
4342
          Below is an example of how to define and integrate a custom CUDA kernel for element-wise
4343
               addition:
4344
          ```python
4345
 import torch
4346
 import torch.nn as nn
4347
 from torch.utils.cpp_extension import load_inline
4348
4349
 # Define the custom CUDA kernel for element-wise addition
 elementwise_add_source = ""'
4350
 #include <torch/extension.h>
4351
 #include <cuda_runtime.h>
4352
4353
 __global__ void elementwise_add_kernel(const float* a, const float* b, float* out, int
4354
 size) {
4355
 int idx = blockIdx.x * blockDim.x + threadIdx.x;
 if (idx < size) {
4356
 out[idx] = a[idx] + b[idx];
4357
 }
4358
 }
4359
4360
 torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b) {
 auto size = a.numel();
4361
 auto out = torch::zeros_like(a);
4362
4363
 const int block_size = 256;
4364
 const int num_blocks = (size + block_size - 1) / block_size;
4365
 elementwise_add_kernel<<<num_blocks, block_size>>>(a.data_ptr<float>(), b.data_ptr<
4366
 float>(), out.data_ptr<float>(), size);
4367
4368
 return out;
4369
```

```
4371
4372
4373
4374
 elementwise_add_cpp_source = (
4375
 "torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b);"
4376
4377
 # Compile the inline CUDA code for element-wise addition
4378
 elementwise_add = load_inline(
4379
 name="elementwise_add",
4380
 cpp_sources=elementwise_add_cpp_source,
4381
 cuda_sources=elementwise_add_source,
 functions=["elementwise_add_cuda"],
4382
 verbose=True,
4383
 extra_cflags=[""],
4384
 extra_ldflags=[""],
4385
4386
4387
 class ModelNew(nn.Module):
 def __init__(self) -> None:
4388
 super().__init__()
4389
 self.elementwise_add = elementwise_add
4390
4391
 def forward(self, a, b):
4392
 return self.elementwise_add.elementwise_add_cuda(a, b)
4393
4394
4395
 - Ensure that the optimized model maintains the same accuracy as the original.
4396
 - The custom CUDA kernels should be optimized for performance, minimizing execution time
 and maximizing GPU utilization.
4397
4398
 By following these instructions, you will be able to effectively replace PyTorch operators
4399
 with custom CUDA kernels, achieving significant performance improvements while
4400
 maintaining model accuracy.
4401
4402
```