

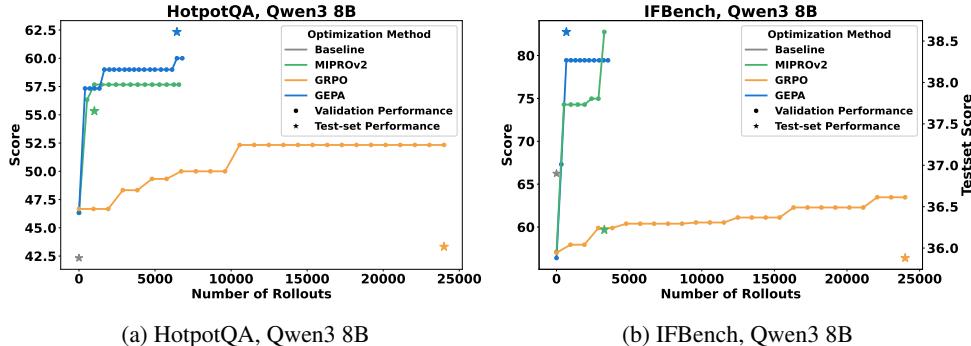
000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 GEPA: REFLECTIVE PROMPT EVOLUTION CAN OUTPER- FORM REINFORCEMENT LEARNING

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 6% 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).



(a) HotpotQA, Qwen3 8B

(b) IFBench, Qwen3 8B

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

047 with Verifiable Rewards (RLVR), e.g. with Group Relative Policy Optimization (GRPO) (Shao et al., 2024),
 048 which treats success metrics as end-of-rollout scalar rewards used to estimate policy gradients (Lambert,
 049 2025). While these RL approaches are effective, they typically require tens of thousands of rollouts in prac-
 050 tice to fit new tasks. For example, recent works leveraging GRPO typically use up to hundreds of thousands
 051 of rollouts for training (Chen et al., 2025b; Wu et al., 2025b; Zhang et al., 2025; Jin et al., 2025; Si et al.,
 052 2025; Wang et al., 2025a; Chen et al., 2025a; Sha et al., 2025; Lin et al., 2025a; Peng et al., 2025; Song
 053 et al., 2025). This sample inefficiency can quickly become a serious bottleneck: many downstream LLM
 054 applications invoke expensive tool calls, have limited inference budget for sampling from the LLM itself, or
 055 simply cannot finetune the weights of the largest or best-performing LLMs.

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

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

070 We evaluate GEPA across multi-hop reasoning (HotpotQA; Yang et al. 2018), Math (AIME, LiveBench-
 071 Math; Balunović et al. (2025); White et al. (2025)), instruction following (IFBench; Pyatkin et al. 2025b),
 072 privacy-aware delegation (PUPA; Li et al. 2025a), and retrieval-augmented verification (HoVer; Jiang et al.
 073 2020), using both open (Qwen3 8B; Yang et al. 2025; Team 2025) and proprietary (GPT-4.1 Mini; OpenAI
 074 2025) models. We find that GEPA generalizes well and is highly sample-efficient: on Qwen3 8B, it out-
 075 performs GRPO (24k rollouts) by up to 19% while using up to $35\times$ fewer rollouts, with an average gain of
 076 +10% across tasks. GEPA also surpasses the prior state-of-the-art, MIPROv2 (Opsahl-Ong et al., 2024), on
 077 all benchmarks and models, achieving +13% aggregate gains, over double MIPROv2’s +5.6%.

078 Qualitatively, GEPA-learned prompts are quite rich. Figure 2 shows excerpts from a prompt crafted for the
 079 query-creation module of a multi-hop question answering system used in HotPotQA, and Figure 26 shows
 080 that even a single *reflective* update often yields large gains. These results highlight that reflective prompt
 081 evolution with language feedback enables improved sample efficiency and robust generalization, offering a
 082 practical approach to optimizing complex AI workflows in data- or budget-constrained environments. We
 083 also present preliminary evidence of GEPA’s broader utility as an inference-time search strategy for code
 084 optimization on NPUEval (Kalade & Schelle, 2025) and KernelBench (Ouyang et al., 2025) in Section 5.

085 2 PROBLEM STATEMENT

086 We follow related work in defining a **compound AI system** as any modular system composed of one or more
 087 language model (LLM) invocations, potentially interleaved with external tool calls, orchestrated through ar-
 088 bitrary control flow. This definition subsumes a broad class of real-world LLM-based AI systems, including
 089 *agents*, *multi-agent systems*, and general-purpose scaffolding techniques like ReAct (Yao et al., 2023), Ar-
 090 chon (Saad-Falcon et al., 2025), etc. Following Soylu et al. (2024); Khattab et al. (2024); Opsahl-Ong et al.
 091 (2024); Tan et al. (2025), we formalize such a system as $\Phi = (M, C, \mathcal{X}, \mathcal{Y})$, where $M = \langle M_1, \dots, M_{|M|} \rangle$

094	Seed Prompt for Second-Hop of Multi-Hop QA System
095	Given the fields <code>question</code> , <code>summary_1</code> , produce the fields <code>query</code> .
096	
097	
098	GEPA's Optimized Prompt for Second-Hop of Multi-Hop QA System, GPT-4.1 Mini
099	
100	You will be given two input fields: <code>question</code> and <code>summary_1</code>. Your task: Generate a new search query (query) <i>optimized for the second hop</i> of a multi-hop retrieval system.
101	<ul style="list-style-type: none"> • 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 <i>not</i> found in first hop but necessary to answer the question completely.
102	
103	
104	
105	Input Understanding: <code>question</code> is the original multi-hop question posed by the user. <code>summary_1</code> is a concise summary of information from a document retrieved in the first hop, which partially addresses the question.
106	
107	Purpose and Context:
108	<ul style="list-style-type: none"> • Your generated query aims to find the <i>missing pieces</i> of information needed to fully answer the question. ... • The query must retrieve relevant documents <i>NOT</i> found in first hop ... for final answer extraction.
109	
110	Key Observations and Lessons:
111	<ul style="list-style-type: none"> • First-hop documents often cover one entity or aspect. (... truncated for space ...)
112	How to Build the Query:
113	<ul style="list-style-type: none"> • Identify entities or topics mentioned in <code>summary_1</code> that are related but different from first-hop documents. (... truncated for space ...)
114	Practical Strategy:
115	<ul style="list-style-type: none"> • Read the <code>summary_1</code> 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 query targeting that entity or concept to retrieve the missing documents.
116	
117	Output:
118	<ul style="list-style-type: none"> • 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 <code>summary_1</code>. Do not include the original question or simply rephrase it. Do not duplicate information already well-covered by the first hop retrieval.
119	
120	
121	
122	

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 x maps 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 $\mu : \mathcal{Y} \times \mathcal{M} \rightarrow [0, 1]$ 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.:

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

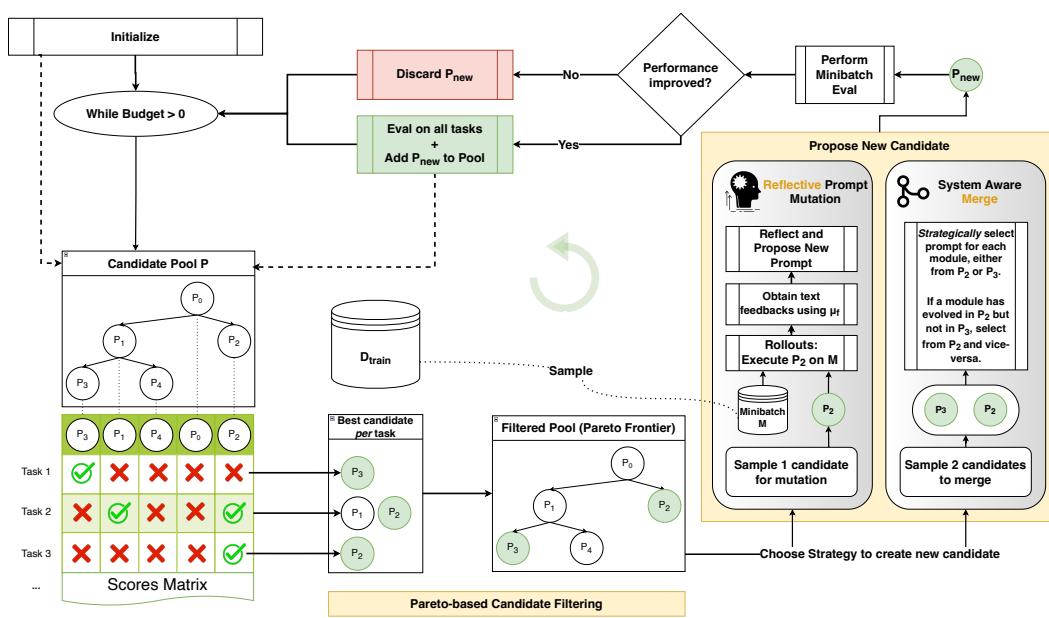


Figure 3: GEPA proposes a new candidate in every iteration by improving existing candidates using one of the two strategies (Reflective Prompt Mutation (Section 3) or System Aware Merge (Appendix D.1)), first evaluating them on a minibatch, and if improved, evaluating on a larger dataset. Instead of 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.

We adopt this general problem formulation, allowing updates to both prompts and weights of language modules, to enable comparisons between optimization algorithms that operate in different parameter spaces (e.g., GEPA vs. GRPO).

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 :

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

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 (stochastically selected from the Pareto-frontier, see Section 3.1 below), GEPA executes the selected candidate on a stochastically sampled minibatch of input queries from the trainset, tracing the program’s execution. From the execution traces, GEPA extracts the module’s inputs, outputs, and reasoning, and calls the *feedback function* μ_f , which returns a numeric score and text feedback including details about the evaluation (like compiler error messages, failed rubrics, etc.). GEPA selects the module (among the $|M|$ modules that the language program contains) to be updated based on a policy (round-robin), and a reflection LM is then shown the (current prompt, language program trajectory, score, feedback) with the task to reflectively attribute successes or failures to prompt elements and propose revised instructions. The updated module, with the rest of the language program, is evaluated again on the minibatch, and if the score improves, then the new program is added to the candidate pool. The meta-prompt for reflective prompt updates is shown in Appendix C and the full algorithm is presented in Algorithm 1.

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). In practice, there are domains where human-graders are able to rate the AI system’s responses, along with providing detailed feedback justifying their scalar ratings. When available, D_{train} can be augmented with such human-written explanations for each instance; during reflection, and 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.

235 3.1 PARETO-BASED CANDIDATE SELECTION
236237 GEPA is a highly modular algorithm that supports various strategies for candidate selection, with the choice
238 of strategy governing the exploration–exploitation tradeoff. A naive approach is to always select the best-
239 performing candidate, but this often traps the optimizer in a local optimum: once a dominant strategy is
240 found, it becomes difficult to surpass, and the optimizer exhausts its budget without learning new, potentially
241 better strategies. Figure 4a illustrates this behavior: after finding one new strategy (the first child node), the
242 search repeatedly attempts to refine it, fails to improve, and ultimately depletes the budget.243 To address this, GEPA employs a Pareto-based “illumination” strategy (Mouret & Clune, 2015), shown in
244 Algorithm 2. For each training instance, GEPA records the highest score across all candidates, forming
245 a Pareto frontier. Candidates that achieve the best score on at least one task are retained, while strictly
246 dominated ones are pruned. From this pruned set, GEPA stochastically samples a candidate, weighting
247 probabilities by how many tasks each candidate leads. This strategy helps GEPA escape local optima without
248 inflating the search, efficiently balancing exploration and exploitation by focusing resources on candidates
249 that embody “winning” strategies within the optimization budget.250 4 EVALUATION
251252 We adopt a standard train/validation/test split. Optimizers have full access to the train split, including text
253 and labels, for program tuning. Although optimizers may monitor the performance of candidate parameters
254 (like model checkpoints) by tracking scores on the validation set (to implement early stopping, for example),
255 direct access to the content of validation instances is restricted. We evaluate on six benchmarks—AIME-
256 2025 (Balunović et al., 2025), LiveBench-Math (White et al., 2025), HotpotQA (Yang et al., 2018), IF-
257 Bench (Pyatkin et al., 2025b), HoVer (Jiang et al., 2020), and PUPA (Li et al., 2025a)—each paired with
258 existing compound AI systems and feedback functions. Experiments use Qwen3 8B (Yang et al., 2025) and
259 GPT-4.1 Mini (OpenAI, 2025) with standardized inference settings, and compare against state-of-the-art
260 optimizers MIPROv2 (Opsahl-Ong et al., 2024), [Trace \(with its OptoPrime optimizer\) \(Cheng et al., 2024\)](#),
261 [TextGrad \(Yuksekgonul et al., 2025\)](#), and GRPO¹ (Shao et al., 2024). Appendix E provides further details
262 on benchmarks, systems, and feedback functions (Subsection E.1); models and inference settings (Subsec-
263 tion E.2); [monetary cost to run the experiments \(Subsection E.3\)](#); and optimizer configurations (Subsec-
264 tion E.4). Table 1, Table 2 and Figure 7 summarize our main results, from which we derive the following
265 observations:266 **Observation 1: Reflective Prompt Evolution is highly sample-efficient and can outperform weight-
267 space reinforcement learning:** Across four benchmarks, GEPA adapts rapidly and generalizes robustly
268 in compound AI systems—beating GRPO (24,000 rollouts) by up to 19% while using up to 35× fewer
269 rollouts. It reaches optimal test performance with 4–35× fewer rollouts and exceeds GRPO on [5 out of 6](#)
270 tasks by 19.0%, 2.73%, 13.66%, 5.19% and 0.7%. GEPA matches GRPO’s best validation after only 243,
271 402, 330, 1143, 1179, and 306 rollouts—up to 78× greater sample efficiency. GEPA+Merge widens the
272 gap, outperforming GRPO by 21% at a comparable rollout budget to GEPA.273 The majority of GEPA’s rollout budget is spent on validation, where scores are utilized solely for candidate
274 selection and not for producing learning signals. If we restrict the analysis to train set rollouts, GEPA
275 requires only 79 to 737 rollouts to reach optimal performance. To match GRPO’s best validation scores,
276 GEPA achieves this with only 102, 32, 6, and 179 train rollouts for four tasks, respectively, underscoring the
277 high sample efficiency of learning based on reflective prompt evolution.278 ¹We use LoRA for GRPO due to its low cost and successful adoption with GRPO (Wang et al., 2025b; Xu et al.,
279 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;
280 Zhao et al., 2024; Sidahmed et al., 2024). Additionally, we explore full-parameter finetuning. Figure 10 shows a similar
281 result comparing GEPA to GRPO with full finetuning.

282 Table 1: Benchmark results for different optimizers with Qwen3 8B. GEPA and GEPA+Merge achieve better
 283 performance than GRPO with far fewer rollouts on all benchmarks except AIME. For example, for IFBench,
 284 GEPA found optimal prompts after just 678 rollouts achieving 38.61%, outperforming GRPO’s test set score
 285 of 35.88% with 24,000 rollouts.

Qwen3 8B	HotpotQA	IFBench	HoVer	PUPA	AIME-2025	LiveBench-Math	Aggregate	Improvement
Baseline	42.33	36.90	35.33	80.82	27.33	48.70	45.23	—
GRPO	43.33	35.88	38.67	86.66	38.00	51.26	48.91	+3.68
MIPROv2	55.33	36.22	47.33	81.55	20.00	46.60	47.84	+2.61
GEPA	62.33	38.61	52.33	91.85	32.00	51.95	54.85	+9.62
GEPA+Merge	64.33	28.23	51.67	86.26	32.00	51.95	52.40	+7.17
Total optimization budget (# rollouts)								
GEPA (+Merge)	6871	3593	7051	2426	1839	1839	3936	—
GRPO	24000	24000	24000	24000	24000	24000	24000	—

295 Table 2: Benchmark results for different optimizers evaluated on GPT-4.1 Mini. As a prompt-optimization
 296 system, GEPA works off-the-shelf on *closed-source* models as well, outperforming state-of-the-art prompt
 297 optimizers including MIPROv2 (in 2 settings: **Instruction-only optimization** (“MIPROv2-No-Demos”) as
 298 well as **joint instruction and few-shot optimization** (“MIPROv2”)), Trace (with its OptoPrime optimizer), and
 299 TextGrad. Additionally, GEPA-optimized prompts demonstrate strong cross-model generalization: “GEPA-
 300 Qwen-Opt”—optimized entirely for (and using) the weaker Qwen3-8B—achieves a +9% gain when eval-
 301 uated on GPT-4.1-Mini without modification, notably outperforming all baselines (MIPROv2, TextGrad,
 302 Trace) that optimized directly for (and using) GPT-4.1-Mini.

GPT-4.1 Mini	HotpotQA	IFBench	HoVer	PUPA	AIME-2025	LiveBench-Math	Aggregate	Improvement
Baseline	38.00	47.79	46.33	78.57	49.33	58.20	53.03	—
Trace (OptoPrime)	60.33	51.19	46.00	74.18	45.33	60.74	56.30	+3.27
MIPROv2-No-Demos	38.00	52.04	51.33	91.85	48.67	60.97	57.14	+4.11
MIPROv2	58.00	49.15	48.33	83.37	51.33	61.84	58.67	+5.64
TextGrad	62.33	48.64	47.67	85.68	46.67	63.84	59.14	+6.11
GEPA	69.00	52.72	51.67	94.47	59.33	64.13	65.22	+12.19
GEPA+Merge	65.67	55.95	56.67	96.46	59.33	64.13	66.36	+13.33
Optimized with Qwen3-8B, evaluated on GPT-4.1-Mini								
GEPA-Qwen-Opt	65.67	49.83	54.67	90.05	52.67	59.31	62.03	+9.00

512 Since tracking candidates’ validation performance accounts for majority of GEPA’s rollout budget, sample
 513 efficiency can be further improved by evaluating on a smaller validation set or by tracking scores on dynam-
 514 ically selected validation subsets instead of the full set—both of which we propose as directions for future
 515 work. Figures 1a, 1b, 13c and 14c show the full performance-vs-rollouts curve for all optimizers over
 516 benchmarks HotpotQA, IFBench, HoVer and PUPA, respectively.

517 **Observation 2: Reflective prompt evolution enables instruction-optimization alone to outperform joint
 518 instruction and few-shot optimization:** We compare GEPA with MIPROv2, a state-of-the-art instruction
 519 and few-shot optimizer, using two leading models across six diverse tasks, and observe that GEPA con-
 520 sistently outperforms MIPROv2 in all settings, achieving margins as high as 11.1% for GPT-4.1 mini and
 521 10.3% for Qwen3 8B. Further, GEPA and GEPA+Merge more than double the aggregate gains over baseline
 522 seen with MIPROv2 across all benchmarks and models (+13.33% and +12.19% vs +5.64% for MIPROv2).

523 While prior works such as Opsahl-Ong et al. (2024) and Wan et al. (2024) have provided compelling ev-
 524 idence for the effectiveness of few-shot example optimization—often outperforming instruction-based ap-
 525 proaches—our findings suggest an exciting shift in this trend. We attribute this primarily to recent advances
 526 in the instruction-following and self-reflective abilities of LLMs, as well as the design choices in GEPA
 527 that capitalize on these improved capabilities. To further contextualize our findings, we redo the study on
 528 *generalization gap* (the difference between validation and test set performance for optimized prompts) as

Table 3: Comparing candidate selection strategies across different tasks with Qwen3 8B while keeping the evolution harness fixed. At each step, `SelectBestCandidate` (used by TextGrad Yuksekgonul et al. (2025)) evolves only from the top-scoring candidate. `BeamSearch` maintains a pool of the top-N candidates (used by APO Pryzant et al. (2023)), but is still prone to local optima. In comparison, GEPA’s Pareto-based selection yields a +12.44% improvement, significantly outperforming the +6.05% and +5.11% gains of greedy and beam-search strategies respectively.

Qwen3 8B	HotpotQA	IFBench	Hover	PUPA	Aggregate	Improvement
Baseline	42.33	36.90	35.33	80.82	48.84	—
SelectBestCandidate	58.33	30.44	45.33	85.45	54.89	+6.05
BeamSearch	57.33	36.39	41.00	81.08	53.95	+5.11
GEPA	62.33	38.61	52.33	91.85	61.28	+12.44

proposed by Wan et al. (2024). The results presented in Figure 15 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.

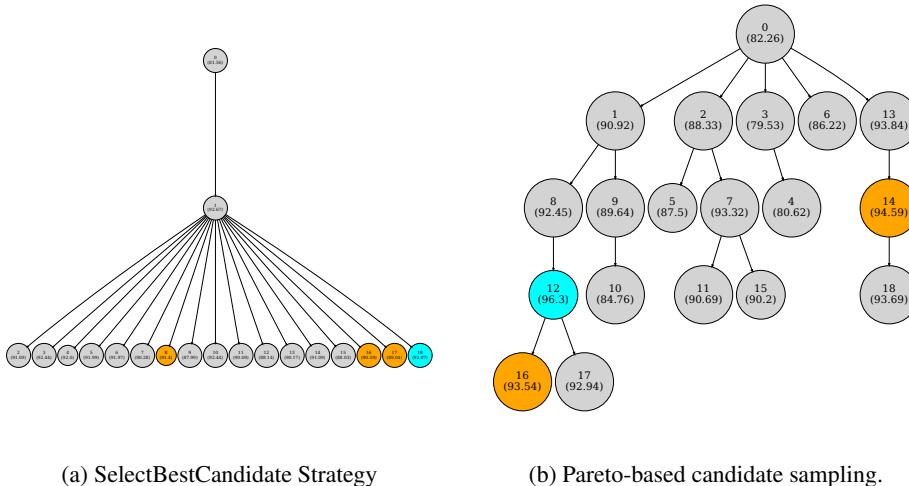


Figure 4: Comparing the impact of different candidate selection strategies. (Left) As can be seen, selecting the best-performing candidate in every iteration led to a local-optima after one iteration, leading to suboptimal search performance. (Right) On the other hand, using pareto-based candidate selection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

376 GEPA refines prompts iteratively with rollout feedback; to test our Pareto-based selection, we compare
 377 against a baseline that always picks the best-performing candidate in the `SelectBestCandidate` strategy
 378 (which is similar to the strategy used by [TextGrad](#) [Yuksekgonul et al. \(2025\)](#)), and `BeamSearch`(N=4) (used
 379 by [APO](#) [Pryzant et al. \(2023\)](#)). As shown in Table 3, these baselines often yield suboptimal exploration
 380 of the prompt search space, leading to poor performance. GEPA with Pareto-based sampling outperforms
 381 the `BeamSearch` strategy by upto 11.33%, and `SelectBestCandidate` strategy by up to 8.17%, with an
 382 aggregate margin of +7.33% and +6.4% across all benchmarks, respectively. Figure 4 highlights the differ-
 383 ence in optimization trajectories: always choosing the current best candidate gives immediate improvement
 384 but quickly stalls, wasting rollouts on a single candidate. In contrast, our Pareto-based method expands
 385 the search by considering all Pareto-optimal candidates (all “winning” strategies found so far), balancing
 386 exploration and exploitation and converging to a higher-performing solution within the same rollout budget.

387 We also find that **instruction-optimized prompts are more efficient and generalize better than few-
 388 shot demonstrations (Observation 4)**. Further, **System-aware crossover strategies such as Merge can
 389 improve performance, but their effectiveness depends on budget allocation between mutation and
 390 crossover and on timing (Observation 5)**. Finally, **GEPA-optimized prompts demonstrate cross-model
 391 generalization (Observation 6)**. See Appendix F for further discussion.

392 5 EXTENDED APPLICATIONS OF GEPA

393 **GEPA For Inference-Time Search** GEPA can be used as an inference-time search technique by setting
 394 the validation set equal to the training set ($D_{\text{val}} = D_{\text{train}}$), under which GEPA iteratively proposes better sol-
 395 lutions for each target task. We evaluate this approach on hardware-specific code generation: writing AMD
 396 NPU kernels (NPUEval (Kalade & Schelle, 2025)) and generating NVIDIA GPU’s CUDA kernels (Kernel-
 397 Bench (Ouyang et al., 2025)) using GPT-4o OpenAI (2024). GEPA raises mean kernel vector utilization on
 398 NPUEval from 4.25% (sequential refinement baseline) to 30.52%, and increases the fraction of KernelBench
 399 tasks with CUDA code surpassing baseline-PyTorch speed from close to 0% to over 20%. Full experimental
 400 details are provided in Appendix G.

401 **GEPA for Adversarial Prompt Search** GEPA can discover adversarial prompts by inverting the re-
 402 ward—finding instructions that minimize task performance. On AIME-2025 with GPT-5 Mini, starting from
 403 a simple instruction, GEPA evolved a trivia-style distractor that reduced pass@1 from 76% to 10%. Adver-
 404 sarial search drew on AIME 2022–2024 problems, with evaluation on AIME-2025 using 5 runs per problem
 405 (30 problems; 150 generations). This robustness probe surfaces brittle instruction-following interactions
 406 (e.g., trivia with strict formatting) and yields reusable stress tests and regression suites for deployment and
 407 show that reflective prompt evolution is also effective for stress-testing them by locating universal, query-
 408 agnostic perturbations that sharply degrade accuracy. These findings parallel recent results on query-agnostic
 409 adversarial triggers, which report two- to seven-fold increases in error rates across model families; in our
 410 AIME-2025 setting, GEPA’s learned universal instruction yields a comparable 3.8× increase (errors from
 411 24% to 90%) (Rajeev et al., 2025). Full experimental details are provided in Appendix H.

412 6 RELATED WORK

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

423 **Evolutionary algorithms** have been used to optimize prompts, e.g., EvoPrompt (Guo et al., 2024), which
 424 evolves prompt populations. Rainbow Teaming (Samvelyan et al., 2024) applies quality-diversity evolution
 425 to generate diverse adversarial prompts. GEPA additionally uses domain-specific feedback for targeted mu-
 426 tations achieving higher sample efficiency. AlphaEvolve (Novikov et al., 2025) and OpenEvolve (Sharma,
 427 2025) apply evolutionary search directly to code rewriting, excelling when problem solution can be codi-
 428 fied. While AlphaEvolve targets a single hard problem, GEPA brings evolution to prompts across domains,
 429 combining Pareto-frontier optimization and prompt evolution to transfer tactics from related problems.

430 **Feedback-driven improvement** often uses reinforcement learning, such as majority voting signals (Zuo
 431 et al., 2025), but RL can be sample-inefficient when rewards are slow to compute. An alternative is learning
 432 in the language space: in-context bandit/self-bootstrapping (Shinn et al., 2023; Madaan et al., 2023) (Monea
 433 et al., 2025; Xu et al., 2025a; Feng et al., 2025; Cheng et al., 2024), workflow memory and skills (Wang
 434 et al., 2024; 2025c), and test-time strategy synthesis via Dynamic Cheatsheet (Suzgun et al., 2025), reasoning
 435 cache (Chen et al., 2025c). GEPA instead uses examples to propose new *instructions*, yielding task-specific
 436 rules.

437 To **optimize compound AI systems and agents** (Lin et al., 2025b), DSPy (Khattab et al., 2022; 2024)
 438 searches/bootstrap few-shot examples, TextGrad (Yuksekgonul et al., 2025) backpropagates textual feed-
 439 back, and MIPROv2 (Opsahl-Ong et al., 2024) jointly aligns instructions and examples via Bayesian opti-
 440 mization; these largely rely on global rewards. **Agent-Pro** (Zhang et al., 2024) evolves agent policies through
 441 **dynamic belief generation and reflection on interactive experiences**. Optimas (Wu et al., 2025a) introduces
 442 globally aligned local rewards per module. GEPA combines global rewards with environment textual feed-
 443 back per module and maintains a Pareto frontier over individual data instances, matching prompts/agent
 444 design to specific examples. The Pareto-guided evolution lets GEPA explore diverse prompt/code/agent
 445 design strategies before converging to a robust, generalizable set.

446 7 CONCLUSION

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

457 458 REFERENCES

460 Advanced Micro Devices. Amd xdna™ architecture. <https://www.amd.com/en/technologies/xdna.html#xdna2>, 2025. Accessed on: 2025-07-23.

463 Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi.
 464 Promptwizard: Task-aware prompt optimization framework, 2024. URL <https://arxiv.org/abs/2405.18369>.

467 Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran
 468 Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

470 Meta AI. llama-prompt-ops: An open-source tool for seamless migration from other llms to llama, and for
 471 general prompt optimization. <https://github.com/meta-llama/llama-prompt-ops>, 2025. Accessed:
 472 2025-07-11.

473

474 Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Aime-2025, Febru-
 475 ary 2025. URL https://huggingface.co/datasets/MathArena/aime_2025. Dataset from Math-
 476 Arena: Evaluating LLMs on Uncontaminated Math Competitions.

477

478 Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang,
 479 Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia,
 480 Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement
 481 learning, 2025.

482 Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen
 483 Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to Reason
 484 with Search for LLMs via Reinforcement Learning, March 2025a. URL <http://arxiv.org/abs/2503.19470> [cs].

485

486 Peter Chen, Xiaopeng Li, Ziniu Li, Xi Chen, and Tianyi Lin. Spectral Policy Optimization: Coloring your In-
 487 correct Reasoning in GRPO, May 2025b. URL <http://arxiv.org/abs/2505.11595> [cs].

488

489 Peter Baile Chen, Yi Zhang, Dan Roth, Samuel Madden, Jacob Andreas, and Michael Cafarella. Log-
 490 augmented generation: Scaling test-time reasoning with reusable computation, 2025c. URL <https://arxiv.org/abs/2505.14398>.

491

492 Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization
 493 with rich feedback, execution traces, and llms, 2024. URL <https://arxiv.org/abs/2406.16218>.

494

495 Xidong Feng, Bo Liu, Yan Song, Haotian Fu, Ziyu Wan, Girish A. Koushik, Zhiyuan Hu, Mengyue Yang,
 496 Ying Wen, and Jun Wang. Natural language reinforcement learning, 2025. URL <https://arxiv.org/abs/2411.14251>.

497

498 Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rockt"aschel.
 499 Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

500

501 Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz,
 502 Kourosh Hakhamaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica.
 503 Evolving skyrl into a highly-modular rl framework, 2025. Notion Blog.

504

505 Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yu-
 506 jiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt op-
 507 timizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=ZG3RaNI08>.

508

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

511

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

514

515

516

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

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

525 Sarunas Kalade and Graham Schelle. Npueval: Optimizing npu kernels with llms and open source compilers,
 526 2025. URL <https://arxiv.org/abs/2507.14403>.

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

530 Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan
 531 A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and
 532 Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In
 533 *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=sY5N0zY50d>.

534 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez,
 535 Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with
 536 pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626,
 537 2023.

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

541 Siyan Li, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. PAPILLON:
 542 Privacy preservation from Internet-based and local language model ensembles. In Luis Chiruzzo, Alan
 543 Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of
 544 the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*,
 545 pp. 3371–3390, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics.
 546 ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.nacl-long.173. URL <https://aclanthology.org/2025.nacl-long.173/>.

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

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

552 Matthieu Lin, Jenny Sheng, Andrew Zhao, Shenzhi Wang, Yang Yue, Victor Shea Jay Huang, Huan Liu, Jun
 553 Liu, Gao Huang, and Yong-Jin Liu. Training of scaffolded language models with language supervision:
 554 A survey, 2025b. URL <https://arxiv.org/abs/2410.16392>.

555 Shu Liu, Sumanth Hegde, Shiyi Cao, Alan Zhu, Dacheng Li, Tyler Griggs, Eric Tang, Akshay Malik,
 556 Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Sto-
 557 ica. Skyrl-sql: Matching gpt-4o and o4-mini on text2sql with multi-turn rl, 2025.

564 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon,
 565 Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,
 566 Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refine-
 567 ment with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine
 568 (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran
 569 Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

571 Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. Llms are in-context bandit reinforce-
 572 ment learners, 2025. URL <https://arxiv.org/abs/2410.05362>.

574 Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015. URL <https://arxiv.org/abs/1504.04909>.

577 Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang,
 578 Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Ab-
 579 bas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex
 580 Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A cod-
 581 ing agent for scientific and algorithmic discovery. Technical report, Google DeepMind,
 582 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.

585 OpenAI. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>, 2024. Accessed: 2025-09-08.

587 OpenAI. GPT-4.1 series, 2025. Large language model series, released April 2025. <https://openai.com/index/gpt-4-1/>.

589 Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and
 590 Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs.
 591 In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on
 592 Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November
 593 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL <https://aclanthology.org/2024.emnlp-main.525/>.

596 Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Re, and Azalia Mirho-
 597 seini. Kernelbench: Can LLMs write efficient GPU kernels? In *Scaling Self-Improving Foundation
 598 Models without Human Supervision*, 2025. URL <https://openreview.net/forum?id=k6V4jb8jkX>.

600 Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. VerIF: Verification Engineering
 601 for Reinforcement Learning in Instruction Following, June 2025. URL <http://arxiv.org/abs/2506.09942> [cs].

603 Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt
 604 optimization with “gradient descent” and beam search. In Houda Bouamor, Juan Pino, and Kalika
 605 Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Pro-
 606 cessing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi:
 607 10.18653/v1/2023.emnlp-main.494. URL <https://aclanthology.org/2023.emnlp-main.494/>.

608 Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan
 609 Lambert, and Hannaneh Hajishirzi. IF-RLVR-Train, July 2025a. URL https://huggingface.co/datasets/allenai/IF_multi_constraints_upto5.

611 Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan
 612 Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025b. URL <https://arxiv.org/abs/2507.02833>.

613

614 Meghana Rajeev, Rajkumar Ramamurthy, Prapti Trivedi, Vikas Yadav, Oluwanifemi Bamgbose, Sathwik Te-
 615 jaswi Madhusudan, James Zou, and Nazneen Rajani. Cats confuse reasoning llm: Query agnostic adver-
 616 sarial triggers for reasoning models, 2025. URL <https://arxiv.org/abs/2503.01781>.

617

618 Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha,
 619 E. Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, and Azalia Mirhoseini. Archon: An ar-
 620 chitecture search framework for inference-time techniques, 2025. URL <https://arxiv.org/abs/2409.15254>.

621

622 Mikayel Samvelyan, Sharath Chandra Raparthi, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish
 623 Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Tim Rocktäschel, and Roberta
 624 Raileanu. Rainbow teaming: open-ended generation of diverse adversarial prompts. In *Proceedings of*
 625 *the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY,
 626 USA, 2024. Curran Associates Inc. ISBN 9798331314385.

627

628 Zeyang Sha, Shiwen Cui, and Weiqiang Wang. SEM: Reinforcement Learning for Search-Efficient Large
 629 Language Models, May 2025. URL <http://arxiv.org/abs/2505.07903>. arXiv:2505.07903 [cs].

630

631 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Huawei Zhang, Mingchuan
 632 Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in
 633 open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

634

635 Asankhaya Sharma. Openevolve: Open-source implementation of alphaevolve. <https://github.com/codelion/openevolve>, 2025. GitHub.

636

637 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: lan-
 638 guage agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on*
639 Neural Information Processing Systems, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

640

641 Shuzheng Si, Haozhe Zhao, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Bofei Gao, Kangyang Luo, Wenhao Li,
 642 Yufei Huang, Gang Chen, Fanchao Qi, Minjia Zhang, Baobao Chang, and Maosong Sun. Teaching Large
 643 Language Models to Maintain Contextual Faithfulness via Synthetic Tasks and Reinforcement Learning,
 644 May 2025. URL <http://arxiv.org/abs/2505.16483>. arXiv:2505.16483 [cs].

645

646 Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral
 647 Chaudhary, Roman Komarytsia, Christiane Ahlheim, Yonghao Zhu, Bowen Li, Saravanan Ganesh, Bill
 648 Byrne, Jessica Hoffmann, Hassan Mansoor, Wei Li, Abhinav Rastogi, and Lucas Dixon. Parameter Effi-
 649 cient Reinforcement Learning from Human Feedback, September 2024. URL <http://arxiv.org/abs/2403.10704>. arXiv:2403.10704 [cs].

650

651 Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and
 652 Ji-Rong Wen. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning,
 653 March 2025. URL <http://arxiv.org/abs/2503.05592>. arXiv:2503.05592 [cs].

654

655 Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps
 656 that work better together, 2024. URL <https://arxiv.org/abs/2407.10930>.

657

658 Zhongxiang Sun, Qipeng Wang, Haoyu Wang, Xiao Zhang, and Jun Xu. Detection and Mitigation
 659 of Hallucination in Large Reasoning Models: A Mechanistic Perspective, May 2025. URL <http://arxiv.org/abs/2505.12886>. arXiv:2505.12886 [cs].

658 Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet:
 659 Test-time learning with adaptive memory, 2025. URL <https://arxiv.org/abs/2504.07952>.
 660

661 Shangyin Tan, Lakshya A Agrawal, Arnav Singhvi, Liheng Lai, Michael J Ryan, Dan Klein, Omar Khattab,
 662 Koushik Sen, and Matei Zaharia. Langprobe: a language programs benchmark, 2025. URL <https://arxiv.org/abs/2502.20315>.
 663

664 Qwen Team. Qwen/qwen3-8b. <https://huggingface.co/Qwen/Qwen3-8B>, 2025. Accessed: 2025-07-11.
 665

666 Ryan Teknium, Jeffrey Quesnelle, and Chen Guang. Hermes 3 Technical Report, August 2024. URL
 667 <http://arxiv.org/abs/2408.11857>. arXiv:2408.11857 [cs].
 668

669 Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Arik. Teach better or show smarter? on instructions
 670 and exemplars in automatic prompt optimization. *Advances in Neural Information Processing Systems*,
 671 37:58174–58244, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6b031defd145b02bed031093d8797bb3-Abstract-Conference.html.
 672

673 Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi
 674 Wang, Kam-Fai Wong, and Heng Ji. Acting Less is Reasoning More! Teaching Model to Act Efficiently,
 675 May 2025a. URL <http://arxiv.org/abs/2504.14870>. arXiv:2504.14870 [cs].
 676

677 Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger.
 678 Tina: Tiny Reasoning Models via LoRA, April 2025b. URL <http://arxiv.org/abs/2504.15777>.
 679 arXiv:2504.15777 [cs].
 680

681 Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL
 682 <https://arxiv.org/abs/2409.07429>.
 683

684 Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for
 685 agentic tasks, 2025c. URL <https://arxiv.org/abs/2504.06821>.
 686

687 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le,
 688 and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL
 689 <https://arxiv.org/abs/2201.11903>.
 690

691 Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv,
 692 Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu,
 693 Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A
 694 challenging, contamination-limited ILM benchmark, 2025. URL <https://arxiv.org/abs/2406.19314>.
 695

696 Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenic Grobelnik, Narendra
 697 Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, Diyi Yang, James Zou, and Jure Leskovec.
 698 Optimas: Optimizing compound ai systems with globally aligned local rewards, 2025a. URL <https://arxiv.org/abs/2507.03041>.
 699

700 Yihong Wu, Liheng Ma, Muzhi Li, Jiaming Zhou, Jianye Hao, Ho-fung Leung, Irwin King, Yingxue Zhang,
 701 and Jian-Yun Nie. Reinforcing Question Answering Agents with Minimalist Policy Gradient Optimiza-
 702 tion, July 2025b. URL <http://arxiv.org/abs/2505.17086>. arXiv:2505.17086 [cs].
 703

704 Wanqiao Xu, Allen Nie, Ruijie Zheng, Aditya Modi, Adith Swaminathan, and Ching-An Cheng. Provably
 705 learning from language feedback, 2025a. URL <https://arxiv.org/abs/2506.10341>.
 706

707 Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. Not All Rollouts are Useful: Down-Sampling
 708 Rollouts in LLM Reinforcement Learning, June 2025b. URL <http://arxiv.org/abs/2504.13818>.
 709 arXiv:2504.13818 [cs].
 710

705 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao,
 706 Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge,
 707 Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang,
 708 Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng,
 709 Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan
 710 Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang,
 711 Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yingger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang,
 712 Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.

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

716 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and
 717 Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In
 718 *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

719 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. Re-
 720 act: Synergizing reasoning and acting in language models, 2023. URL <https://arxiv.org/abs/2210.03629>.

721 Lingfan Yu, Jinkun Lin, and Jinyang Li. Stateful large language model serving with pensieve. In *Proceedings
 722 of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 144–158, New York,
 723 NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.
 724 3696086. URL <https://doi.org/10.1145/3689031.3696086>.

725 Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei
 726 Han, and Dong Wang. Hybrid Latent Reasoning via Reinforcement Learning, May 2025. URL [http://arxiv.org/abs/2505.18454](https://arxiv.org/abs/2505.18454). arXiv:2505.18454 [cs].

727 Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and
 728 James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–
 729 616, 2025.

730 Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jieyang Wang, Sheng Guo,
 731 Bo Zheng, Haobo Wang, and Junbo Zhao. LeTS: Learning to Think-and-Search via Process-and-Outcome
 732 Reward Hybridization, May 2025. URL [http://arxiv.org/abs/2505.17447](https://arxiv.org/abs/2505.17447). arXiv:2505.17447 [cs].

733 Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yuetong
 734 Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. In
 735 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the
 736 Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5348–5375, Bangkok, Thailand,
 737 August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.292. URL
 738 <https://aclanthology.org/2024.acl-long.292/>.

739 Siyan Zhao, John Dang, and Aditya Grover. Group Preference Optimization: Few-Shot Alignment of Large
 740 Language Models, October 2024. URL [http://arxiv.org/abs/2310.11523](https://arxiv.org/abs/2310.11523). arXiv:2310.11523 [cs].

741 Siyan Zhao, Devaansh Gupta, Qin Qing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large
 742 Language Models via Reinforcement Learning, June 2025. URL [http://arxiv.org/abs/2504.12216](https://arxiv.org/abs/2504.12216).
 743 arXiv:2504.12216 [cs].

744 Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao,
 745 Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured lan-
 746 guage model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

752 Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy
753 Ba. Large language models are human-level prompt engineers. In *The eleventh international conference*
754 *on learning representations*, 2022.

755
756 Noah Ziems, Dilara Soylu, Lakshya A Agrawal, Isaac Miller, Liheng Lai, Chen Qian, Kaiqiang Song, Meng
757 Jiang, Dan Klein, Matei Zaharia, Karel D’Oosterlinck, Christopher Potts, and Omar Khattab. Multi-
758 module grp: Composing policy gradients and prompt optimization for language model programs, 2025.
759 URL <https://arxiv.org/abs/2508.04660>.

760 Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang,
761 Xinwei Long, Ermo Hua, Binqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen
762 Zhou. Ttrl: Test-time reinforcement learning, 2025. URL <https://arxiv.org/abs/2504.16084>.

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

A APPENDIX OUTLINE

800

- 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
- Number of reflection LM calls made by GEPA during optimization

811

B USAGE OF LARGE LANGUAGE MODELS

812

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.

813

C GEPA'S REFLECTION AND PROMPT UPDATE META PROMPT

814

GEPA's Meta Prompt

815

```
I provided an assistant with the following instructions to perform a task for me:
```

```
---
```

```
<current instruction>
```

```
---
```

The following are examples of different task inputs provided to the assistant along with the assistant's response for each of them, and some feedback on how the assistant's response could be better:

```
---
```

```
<Inputs, Outputs and Feedback for minibatch of examples>
```

```
---
```

Your task is to write a new instruction for the assistant.

844

845

846
 847
 848 Read the inputs carefully and identify the input format and infer detailed task
 849 description about the task I wish to solve with the assistant.

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

855 Provide the new instructions within ` ` blocks.
 856
 857

858 Figure C shows the meta-prompt used by GEPA, which guides the LLM to reflectively refine its current
 859 instruction based on input–output examples and corresponding feedback from the environment.
 860

861 D GEPA ALGORITHM AND METHODOLOGY DETAILS

862
 863 Figure 5 presents the core GEPA Algorithm, along with the algorithm for Pareto-based candidate selection.
 864

865 D.1 MERGE: SYSTEM-AWARE CROSSOVER STRATEGY FOR COMPOUND AI OPTIMIZATION

866
 867 Algorithm 4 provides the instantiation of the System aware Merge strategy used in GEPA+Merge. *Intuitively,*
 868 *merge will be helpful when there are candidates in the pool that learn complementary strategies.* Algorithm 3
 869 *defines the selection criteria: candidates are merged only if they share a common ancestor but have optimized*
 870 *disjoint sets of prompts (complementary strategies), are pareto-optimal, and both candidates improve upon*
 871 *the aggregate performance of the ancestor.* GEPA routinely checks if the pool has 2 such candidates, invoking
 872 *merge when identified.* These strict lineage conditions mean merge occurs sparsely.
 873

874 E EVALUATION SETUP (CONTD.)

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

876
 877 To rigorously evaluate the performance of GEPA and and compare it against current state-of-the-art com-
 878 pound AI system optimizers, we assemble a diverse suite of benchmarks mostly obtained from Tan et al.
 879 (2025), each paired with available Compound AI Systems.
 880

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

888 **IFBench** (Pyatkin et al., 2025b) introduced a benchmark specifically designed to assess language models’
 889 ability to follow precise human instructions, especially output constraints (e.g., “answer only with yes or no”,
 890 or “mention a word at least three times”). The IFBench test set consists of 58 new and out-of-distribution
 891 output constraints and instructions to test system’s ability to generalize to new task constraints. Pyatkin et al.
 892 (2025b) also release IFTrain and IF-RLVR Train data (Pyatkin et al., 2025a) which are used for training.

893

894

895

Algorithm 1 GEPA: Reflective Evolutionary Prompt Optimizer

896

```

Require: Inputs: System  $\Phi$ , dataset  $\mathcal{D}_{\text{train}}$ , eval metric  $\mu$ , feedback function  $\mu_f$ , budget  $B$ 
Require: Hyperparams: minibatch size  $b$ , Pareto set size  $n_{\text{pareto}}$ 
1: Split  $\mathcal{D}_{\text{train}}$  into  $\mathcal{D}_{\text{feedback}}$ ,  $\mathcal{D}_{\text{pareto}}$ , s.t.  $|\mathcal{D}_{\text{pareto}}| = n_{\text{pareto}}$ 
2: Initialize candidates  $\mathcal{P} \leftarrow [\Phi]$ , parents  $\mathcal{A} \leftarrow [\text{None}]$ 
3: for each  $(x_i, m_i)$  in  $\mathcal{D}_{\text{pareto}}$  do
4:    $S_{\Phi}[i] \leftarrow \mu(\Phi(x_i), m_i)$ 
5: end for
6: while budget  $B$  not exhausted do
7:    $k \leftarrow \text{SELECTCANDIDATE}(\mathcal{P}, S)$ 
8:    $j \leftarrow \text{SELECTMODULE}(\Phi_k)$ 
9:    $\mathcal{M} \leftarrow$  minibatch of size  $b$  from  $\mathcal{D}_{\text{feedback}}$ 
10:  Gather feedback, scores, traces for  $\Phi_k[j]$  on  $\mathcal{M}$  using  $\mu_f$ 
11:   $\pi'_j \leftarrow \text{UPDATEPROMPT}(\pi_j, \text{feedbacks}, \text{traces}[j])$ 
12:   $\Phi' \leftarrow$  Copy of  $\Phi_k$  w/ module  $j$  updated by  $\pi'_j$ 
13:   $\sigma, \sigma' \leftarrow$  avg score on  $\mathcal{M}$  (before, after)
14:  if  $\sigma'$  improved then
15:    Add  $\Phi'$  to  $\mathcal{P}$ ; Add  $k$  to  $\mathcal{A}$ 
16:    for each  $(x_i, m_i)$  in  $\mathcal{D}_{\text{pareto}}$  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  $\mathcal{D}_{\text{pareto}}$ 

```

917

918

919

920

921

922

923

924

925

926

927

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 $\mathcal{D}_{\text{pareto}}$ 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.

928

929

930

931

932

933

934

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 satisfied and failed-to-be-satisfied by the system’s response. Our splits contain 150 training examples, 300 for validation, and 294 for testing.

935

936

937

938

939

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.

Algorithm 2 Pareto-based candidate selection

```

1: function SELECTCANDIDATE( $\mathcal{P}, S$ )
2:   // Build instance-wise Pareto sets
3:   for each  $i$  do
4:      $s^*[i] \leftarrow \max_k S_{\mathcal{P}[k]}[i]$ 
5:      $\mathcal{P}^*[i] \leftarrow \{\mathcal{P}[k] : S_{\mathcal{P}[k]}[i] = s^*[i]\}$ 
6:   end for
7:    $\mathcal{C} \leftarrow$  unique candidates in  $\bigcup_i \mathcal{P}^*[i]$ 
8:    $D \leftarrow \emptyset$ 
9:   while there exists  $\Phi \in \mathcal{C} \setminus D$  dominated by
another in  $\mathcal{C} \setminus D$  do
10:     $D \leftarrow D \cup \{\Phi\}$ 
11:   end while
12:   Remove  $D$  from each  $\mathcal{P}^*[i]$  to get  $\hat{\mathcal{P}}^*[i]$ 
13:   Let  $f[\Phi] =$  number of  $i$  for which  $\Phi \in$ 
 $\hat{\mathcal{P}}^*[i]$ 
14:   Sample  $\Phi_k$  from  $\hat{\mathcal{C}}$  with probability  $\propto f[\Phi_k]$ 
15:   return index  $k$  of  $\Phi_k$  in  $\mathcal{P}$ 
16: end function

```

940

941

Algorithm 3 Check if module combination is desirable

```

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

```

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

Algorithm 3 Check if module combination is desirable

```

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

```

Algorithm 4 MERGE: Genetic Crossover for Modular Candidates

```

1: function MERGE( $\mathcal{P}, \mathcal{A}, S, r$ )
2:    $i, j \leftarrow r.\text{sample}(2, |\mathcal{P}|)$  // distinct  $i \neq j$ 
3:    $A_i \leftarrow \text{GETANCESTORS}(i, \mathcal{A}), A_j \leftarrow \text{GETANCES-}$ 
4:   if  $i \in A_j$  or  $j \in A_i$  then
5:     continue // skip direct ancestry
6:   end if
7:   for  $a \in A_i \cap A_j$  do
8:     if this merge  $(i, j, a)$  has been tried before then
9:       continue
10:    end if
11:    if  $S[a] > \min(S[i], S[j])$  then
12:      continue
13:    end if
14:    if not DESIRABLE( $a, i, j, \mathcal{P}$ ) then
15:      continue
16:    end if
17:     $\Phi' \leftarrow$  copy of  $\mathcal{P}[a]$ 
18:    for module  $m = 1$  to  $|M|$  do
19:       $\pi_a \leftarrow \mathcal{P}[a].\mathcal{M}_m.\pi$ 
20:       $\pi_i \leftarrow \mathcal{P}[i].\mathcal{M}_m.\pi$ 
21:       $\pi_j \leftarrow \mathcal{P}[j].\mathcal{M}_m.\pi$ 
22:      if  $\pi_a = \pi_i$  and  $\pi_j \neq \pi_i$  then
23:         $\Phi'.\mathcal{M}_m.\pi \leftarrow \pi_j$ 
24:      else if  $\pi_a = \pi_j$  and  $\pi_i \neq \pi_j$  then
25:         $\Phi'.\mathcal{M}_m.\pi \leftarrow \pi_i$ 
26:      else if  $\pi_i \neq \pi_j \neq \pi_a$  then
27:        Choose  $d^* = \arg \max\{S[i], S[j]\}$ 
        (break ties randomly)
         $\Phi'.\mathcal{M}_m.\pi \leftarrow \pi_{d^*}$ 
28:      else
29:         $\Phi'.\mathcal{M}_m.\pi \leftarrow \pi_i$  // default
30:      end if
31:    end for
32:    return  $(\Phi', i, j, a)$ 
33:  end for
34:  return None
35: end function

```

Figure 6: Details of System Aware Merge. r represents a seeded stochastic sampler.

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.

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)

987 required to make the claim. We obtain the HoverMultiHop program from Tan et al. (2025), which performs
 988 up to 3-hop retrievals using 2 query writer modules, and 2 document summary modules. The textual feed-
 989 back module simply identifies the set of correct documents retrieved, and the set of documents remaining to
 990 be retrieved, and returns them as feedback text. For the full-parameter finetuning results demonstrated in fig-
 991 ure 10, we instantiate a 2-hop program, where the first hop is performed with the initial claim, and the LLM
 992 is prompted in a single turn with the claim and first-hop retrieved documents, to provide the second-hop
 993 search query. For HoVer, we use 150 examples for training, 300 for validation, and 300 for testing.

994 **PUPA** (Li et al., 2025a) propose the task of Privacy-Conscious Delegation: addressing real-world user
 995 queries using an ensemble of trusted and untrusted models. The core challenges are maintaining high re-
 996 sponse quality while minimizing leakage of personally identifiable information (PII) to untrusted models.
 997 Li et al. (2025a) also present PAPILLON, a compound AI system consisting of 2 modules, a user query
 998 rewriter and a response rewriter, run over the trusted model, along with an intermediate call to the untrusted
 999 model with the rewritten query. The feedback text simply provides the breakdown of the aggregate score,
 1000 consisting of a response quality score and a PII leakage score. The dataset is split into 111 training examples,
 1001 111 for validation, and 221 for testing.

1002 E.2 MODELS AND INFERENCE PARAMETERS

1003 We evaluate GEPA and baseline optimizers using two contemporary LLMs, chosen to represent both open-
 1004 source and commercial model families. Each compound AI system is instantiated once per model, with all
 1005 modules (e.g., retrievers, rewriters, answer generators) relying on the same model. All models are allowed a
 1006 context window of upto 16384 tokens for inference.

1007 **Qwen3 8B (Yang et al., 2025):** For our open-source experiments (including GRPO), we use Qwen3-8B.
 1008 Following the recommended settings as per Team (2025), we use a decoding temperature of 0.6, top-p of
 1009 0.95, and top-k of 20 for training as well as inference.

1010 **GPT-4.1 Mini (OpenAI, 2025):** For comparison with large commercial models, we use GPT-4.1 mini
 1011 (openai/gpt-4.1-mini-2025-04-14) accessed via the OpenAI API with a model temperature of 1.0.

1012 E.3 COSTS

1013 It costs under \$500 to run all experiments in Table 2 with GPT-4.1 mini. Specifically, GEPA costs a total of
 1014 \$86, GEPA-Merge costs \$67, MIPROv2 costs \$76, and Trace and TextGrad cost \$172 in total.

1015 E.4 OPTIMIZERS

1016 **Baseline:** The base program is directly evaluated without any further optimization applied.

1017 **MIPROv2 (Opsahl-Ong et al., 2024):** MIPROv2 is a widely used compound AI system prompt optimizer
 1018 and has been integrated into the DSPy (Khattab et al., 2024) and llama-prompt-ops (AI, 2025) frameworks.
 1019 It works by jointly optimizing both instructions and demonstrations using Bayesian optimization. For each
 1020 program module, it first bootstraps candidate sets of instructions and demonstrations, assigning uniform
 1021 priors over their utilities. Candidate assignments are proposed with the Tree-Structured Parzen Estimator
 1022 (TPE), and the Bayesian model is updated based on evaluation scores to favor high-performing candidates.
 1023 The most probable sets of instructions and demonstrations are then selected and validated to obtain the final
 1024 optimized program configuration.

1025 All MIPROv2 optimization runs are performed with the *auto = heavy* setting, which corresponds to propos-
 1026 ing 18 instruction candidates and 18 bootstrapped few-shot sets. Hence, across benchmarks, the exact num-
 1027 ber of rollouts varies depending on the number of trials it takes to bootstrap examples (finding 18 successful

1034 solution instances), the required number of Bayesian search steps (determined by the number of modules
 1035 in the system), and size of the valset. Overall, MIPROv2’s rollouts ranged from a minimum of 2270 (for
 1036 PUPA) to maximum of 6926 (for HoVer).

1037 **Trace and TextGrad (Cheng et al., 2024; Yuksekgonul et al., 2025):** We implement both optimizers in the
 1038 Trace framework. All programs under optimization have the exact same architecture compared to the DSPy
 1039 implementation. To ensure a fair comparison, we port all the DSPy specific signature and parsing prompt
 1040 to Trace, and use the same initial prompt. In addition, all the test, train, and validation data match exactly
 1041 the experiment we used for GEPA. The performance of the unoptimized Trace program baseline closely
 1042 matched our baseline implementation in DSPy (within 0.5% difference). All optimization experiments were
 1043 under the same rollout budget as MIPROv2 and GEPA. We also provide both optimizer the same metric and
 1044 feedback functions as GEPA, and, for the per-module feedback function that is not available in Trace (both
 1045 optimizer do not support per-module feedback), we followed the feedback format in the BigBench-Hard
 1046 tutorial² from the Trace authors.

1047 **GRPO (Shao et al., 2024):** Group Relative Policy Optimization (GRPO) is a reinforcement learning algo-
 1048 rithm that estimates advantages in a group-relative manner. For compound AI systems consisting of multiple
 1049 modules, we use the GRPO implementation provided and open-sourced by Ziems et al. (2025) to perform
 1050 our experiments, whereas for single-module systems (e.g., figure 10), we use the GRPO implementation
 1051 provided by SkyRL (Griggs et al., 2025; Liu et al., 2025; Cao et al., 2025).

1052 Across all compound system training runs, each training step uses a group size of 12, with 4 training
 1053 instances per step (total batch size 48, with per device train batch size 1). Training employs LoRA (Hu et al.,
 1054 2022) with rank dimension 16, $\alpha = 64$, and dropout 0.05, using bf16 precision targeting the projection
 1055 modules [q, k, v, o, up, down, gate]. We use a learning rate of 1×10^{-5} , $\beta = 0.01$, reward scale normaliza-
 1056 tion, and gradient norm clipping of 0.1. Gradients are accumulated for 20 steps before each update, with a
 1057 “constant with warmup learning” rate scheduler. Non-reentrant gradient checkpointing is enabled to further
 1058 reduce memory usage. GRPO optimization run for 500 training steps, amounting to fixed 24,000 rollouts,
 1059 with validation performed every 20 training steps, which is used to implement early stopping. Compound AI
 1060 system GRPO training experiments are performed on 1xH100/A100 (80 GB memory) with separate GPUs
 1061 for inference rollouts.

1062 For single-module GRPO training, we adopt full-parameter finetuning with a group size of 16. Each training
 1063 step employs a global batch size of 32, realized as per-device micro-batches of 4 across 8 GPUs. Roll-
 1064 out generation is performed with a per-GPU forward micro-batch size of 12. Training is distributed using
 1065 FSDP2, with sampling performed at temperature 1.0. We apply KL regularization and set the learning rate
 1066 to 1×10^{-6} . Validation is conducted every 5 training steps. During evaluation, sampling is performed with
 1067 temperature 0.6, top- $p = 0.95$, and top- $k = 20$.

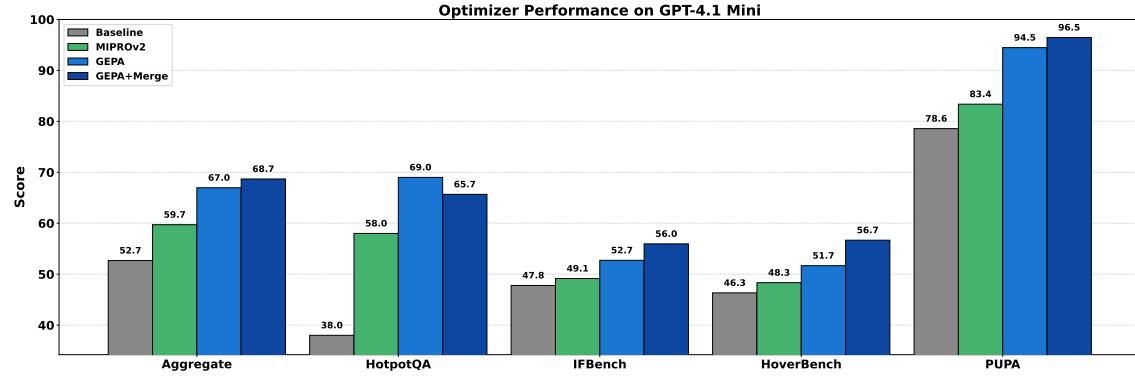
1068 We manually explore several values for [LR, beta, norm clipping] hyperparameters for both training runs.

1069 **GEPA:** GEPA is our optimizer, based on the algorithm described in Section 3. We evaluate 2 variants of our
 1070 main optimizer GEPA: **GEPA** and **GEPA+Merge**, along with 2 ablations created by replacing the Pareto-
 1071 based sampling strategy with a naive, SelectBestCandidate strategy (SelectBestCandidate and SelectBest-
 1072 Candidate+Merge). All GEPA optimization runs use a minibatch size of 3, and merge is invoked a maximum
 1073 of 5 times during the optimization run, when enabled. To ensure a fair comparison with MIPROv2, we align
 1074 the computational budget between GEPA and MIPROv2 on a per-benchmark basis. The training set from
 1075 each benchmark is used as $D_{feedback}$ (which is used to derive the training signals, as discussed in Section 3)
 1076 and the validation set is used as D_{pareto} . Specifically, since MIPROv2’s total rollout budget depends on fac-
 1077 tors such as validation set size and the number of modules, we first record the number of rollouts expended
 1078 by MIPROv2 for each benchmark, and then cap GEPA’s optimization to match this rollout budget. While

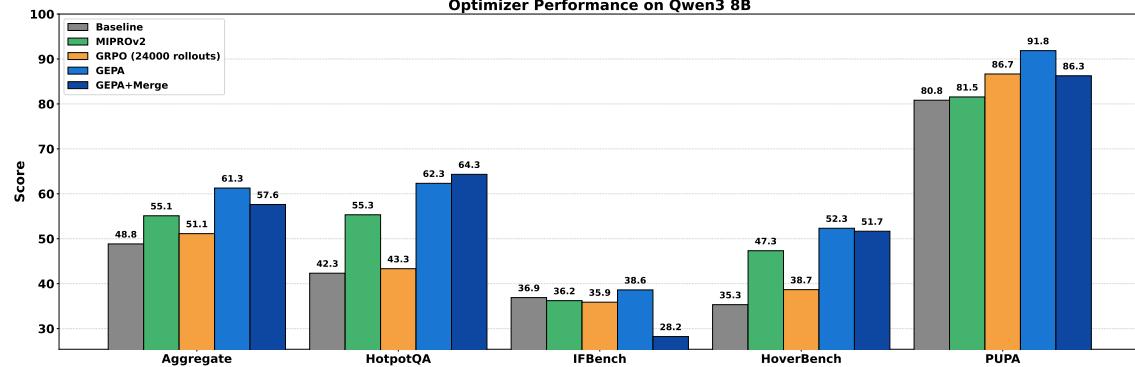
1079
 1080 ²https://microsoft.github.io/Trace/examples/nlp/bigbench_hard.html

1081 differences in proposal and validation procedures cause the exact budget usage by the systems to be slightly
 1082 different, the discrepancy is always within 10.15%. This protocol ensures that any performance differences
 1083 arise from the optimization algorithms themselves, rather than from differences in search budget. The exact
 1084 rollout counts for each optimizer is visualized in Appendix I.
 1085

1086 F RESULTS AND ANALYSIS (CONTD.)



1101 (a) Final test set performance for aggregate and individual benchmarks for gpt-41-mini.



1114 (b) Final test set performance for aggregate and individual benchmarks for qwen3-8b.

1115 Figure 7: Final test set performance for aggregate and individual benchmarks.

1116

1117

1118 **Observation 4: Instruction-optimized prompts are computationally cheaper and generalize better**
 1119 **than few-shot demonstration prompts:** In addition to their strong generalization capabilities, reflectively
 1120 evolved instructions offer a significant practical advantage: they are often much shorter and thus computa-
 1121 tionally more efficient than few-shot demonstration prompts. This advantage becomes especially clear for
 1122 complex tasks, where even a single few-shot demonstration can be prohibitively long. The problem is further
 1123 exacerbated when few-shot examples are optimized using state-of-the-art methods such as MIPROv2, which
 1124 jointly optimizes *multiple* demonstrations to be used simultaneously, further increasing prompt length.

1125 In contrast, reflectively evolved instructions—such as those generated by GEPA—maintain compactness
 1126 while providing large performance gains (as demonstrated in Lessons 1 and 2). To illustrate this, we com-
 1127 pare GEPA’s and MIPROv2’s prompt lengths (see Figure 17). Notably, prompts produced by GEPA and

1128 GEPA+Merge are up to $9.2\times$ shorter than those from MIPROv2, representing a substantial improvement in
 1129 efficiency, alongside performance improvements.
 1130

1131 Moreover, we observe a trend where, in aggregate, optimizers that achieve higher performance tend to
 1132 produce shorter prompts (see Figure 16). This reduction in prompt size has a significant impact—not only
 1133 reducing runtime cost for downstream tasks (as all API-providers meter the input tokens), but also decreasing
 1134 latency and improving the overall efficiency of LLM-serving systems (Kwon et al., 2023; Zheng et al., 2024;
 1135 Agrawal et al., 2023; Yu et al., 2025).

1136 **Observation 5: System aware crossover strategies can provide large gains, but the optimal budget allo-
 1137 cation between mutation and crossover, as well as *when* to invoke merge needs further study:** We iden-
 1138 tify a unique system-aware crossover strategy and operationalize it as Merge (described in Appendix D.1).
 1139 GEPA+Merge can outperform GEPA by as much as 5%, providing an aggregate 2% additional improvement
 1140 over the already strong performance established by GEPA. Detailed results are available in Table 1. We
 1141 attribute these gains to the ability of GEPA+Merge to identify distinct optimization lineages, that have learnt
 1142 complementary strategies (by evolving distinct modules), and merging them by picking the best version of
 1143 different modules from each of these lineages to propose a single, optimal candidate.

1144 While in our analysis, we found GEPA+Merge works especially well for GPT-4.1 Mini, it lead to perfor-
 1145 mance degradation when used with Qwen3 8B. Even Qwen3 8B benefits from Merge on one out of four
 1146 tasks. We attribute these discrepancies to the way the rollout budget is allocated between reflective mu-
 1147 tation and crossover, and the timing of invocation of the crossover strategy. In our experiments, we fixed
 1148 the same hyperparameters for GPT-4.1 Mini and Qwen3 8B, leading to suboptimal choice for Qwen3 8B.
 1149 Intuitively, crossover would provide the maximum benefit, when there are independent lineages that perform
 1150 well. Hence, the hyperparameters should be chosen such that Merge is invoked once the optimization tree
 1151 has evolved sufficiently different lineages. We propose the study of such adaptive techniques as future work.

1152 **Observation 6: GEPA-optimized prompts demonstrate cross-model generalization.** Table 2 presents
 1153 results for “GEPA-Qwen-Opt”, a configuration where prompts were optimized using the smaller Qwen3-8B
 1154 model but evaluated on GPT-4.1-Mini. Despite originating from a weaker model in a different family, these
 1155 prompts transfer effectively, achieving a +9.00% aggregate improvement across 6 benchmarks (with gains
 1156 as high as +27.67% on HotpotQA). Remarkably, this transfer performance outperforms strong baselines like
 1157 MIPROv2 (+5.64%), TextGrad (+6.11%), and Trace (+3.27%), even though those methods were optimized
 1158 directly on the target GPT-4.1-Mini model.

1159 G GEPA FOR INFERENCE-TIME SEARCH (CONTD.)

1160 While the primary focus of this paper is sample-efficient adaptation of AI systems to new tasks, preliminary
 1161 findings suggest that GEPA may also serve as a promising inference-time search technique. This can be
 1162 achieved by passing the set of tasks to be solved (for example, a list of Pytorch modules to be converted
 1163 to CUDA) as the training set to GEPA, ensuring that both D_{train} and D_{pareto} contain the full set of tasks.
 1164 This way, GEPA can “overfit” the set of tasks, iteratively proposing better solutions to every problem. We
 1165 also note that this allows GEPA to apply lessons and insights extracted from rollouts for one task to other
 1166 tasks. To explore this use case, we conduct preliminary experiments using GEPA as an inference-time
 1167 search technique for code-generation tasks on two hardware platforms: writing kernels for AMD’s recently
 1168 introduced XDNA2 Architecture (Advanced Micro Devices, 2025) using an early version of the NPUEval
 1169 benchmark (Kalade & Schelle, 2025), and generating CUDA code for NVIDIA-V100 GPUs using Kernel-
 1170 Bench (Ouyang et al., 2025).

1171 A distinguishing aspect of these experiments is the use of the feedback function μ_f to dynamically inject
 1172 domain-specific knowledge into the optimization process. Specifically, kernel development expertise—often
 1173 codified in technical manuals and documentation—can be selectively surfaced by retrieving relevant man-

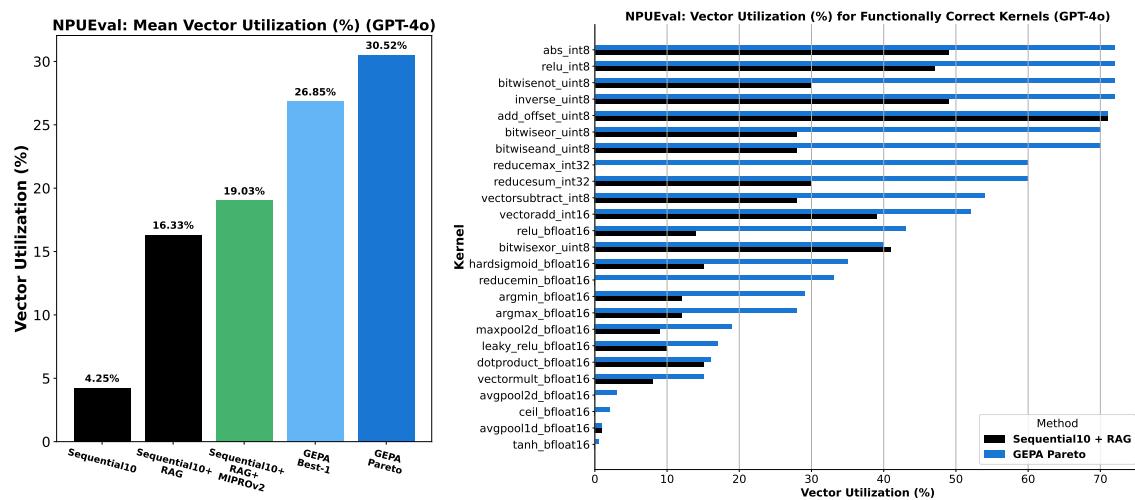


Figure 8: GEPA with GPT-4o is able to generate kernels for AMD NPUs that achieve vector utilization rates as high as 70%, with a mean utilization score of 30.52%. In comparison, GPT-4o, even after up to 10 sequential refinements with environment feedback, achieves an aggregate score of only 4.25%. When 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.

ual 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 27. 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-4o 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-4o. As depicted in Figure 9, GEPA boosts GPT-4o’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-4o 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.

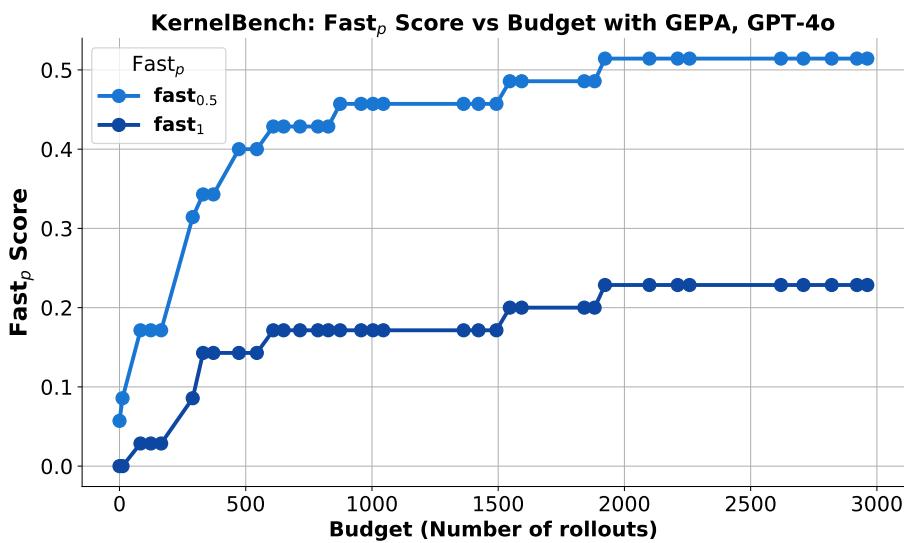


Figure 9: GEPA with GPT-4o is able to iteratively refine and improve CUDA Kernel Code. The graphs shows $fast_p$ vs. rollouts plot for $p=[0.5, 1]$, where the speedup is calculated over Pytorch-eager. $fast_p$ is a metric described in (Ouyang et al., 2025) that measures the fraction of tasks for which the method generated a kernel executing faster than p times the baseline. As can be seen, GEPA with GPT-4o 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

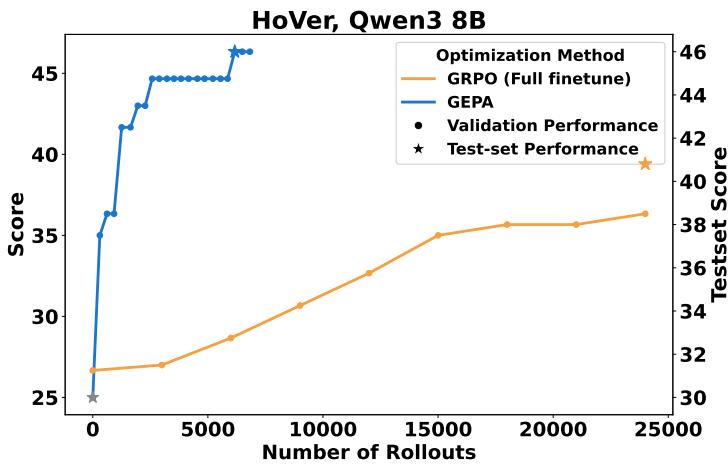


Figure 10: This figure compares the learning behaviour of GEPA against GRPO with full-parameter finetuning on the 2-hop HoVer task. The relative gap mirrors the previously observed comparison of GEPA against GRPO with LoRA (in figures 1, 11, 12, 13, 14), showing that GEPA achieves a comparable performance gap relative to both full-parameter and parameter-efficient versions of GRPO.

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 finetuning 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 11, 12, 13, 14 show the full Performance-vs-Rollout curves for all the optimizers across all benchmarks.

J GENERALIZATION GAP

Figure 15 visualizes the generalization gap for different optimization methods.

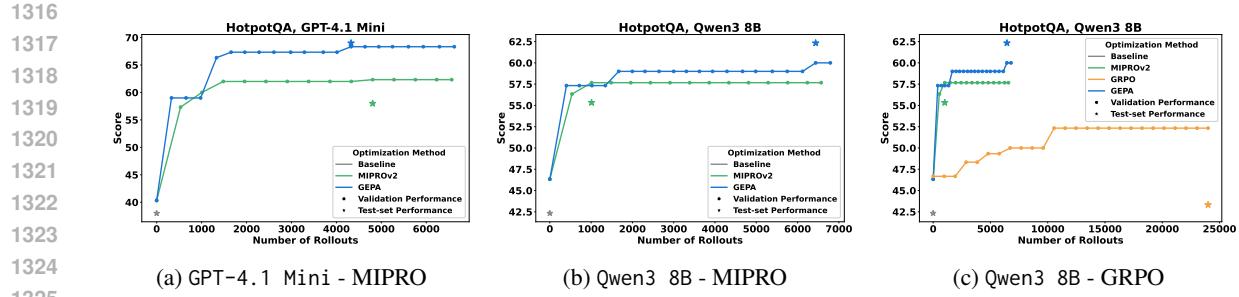


Figure 11: Hotpot QA Bench: rollout vs. score for different models/settings.

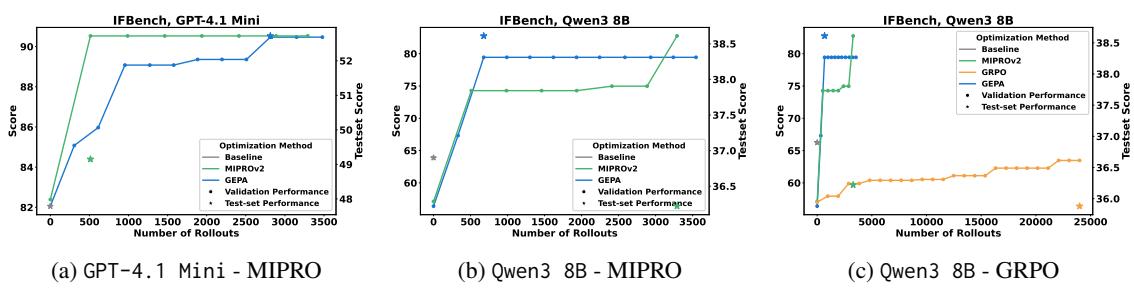


Figure 12: IFBench: rollout vs. score for different models/settings.

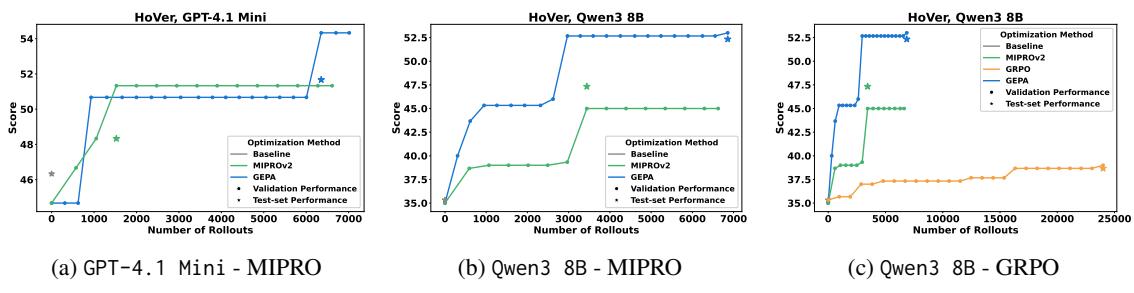


Figure 13: HoverBench: rollout vs. score for different models/settings.

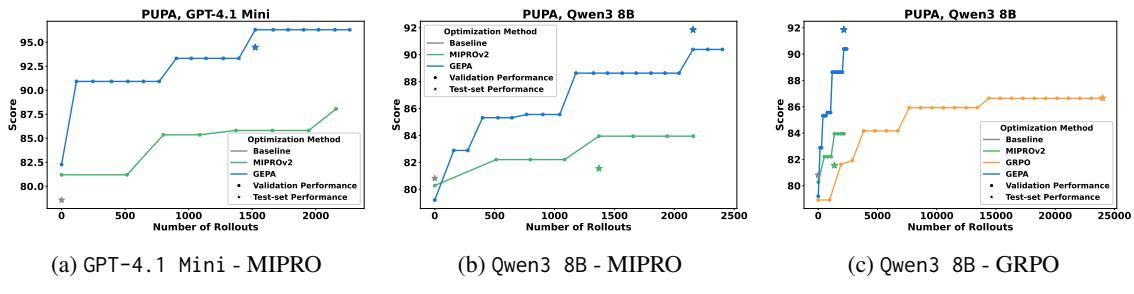


Figure 14: PUPA: rollout vs. score for different models/settings.

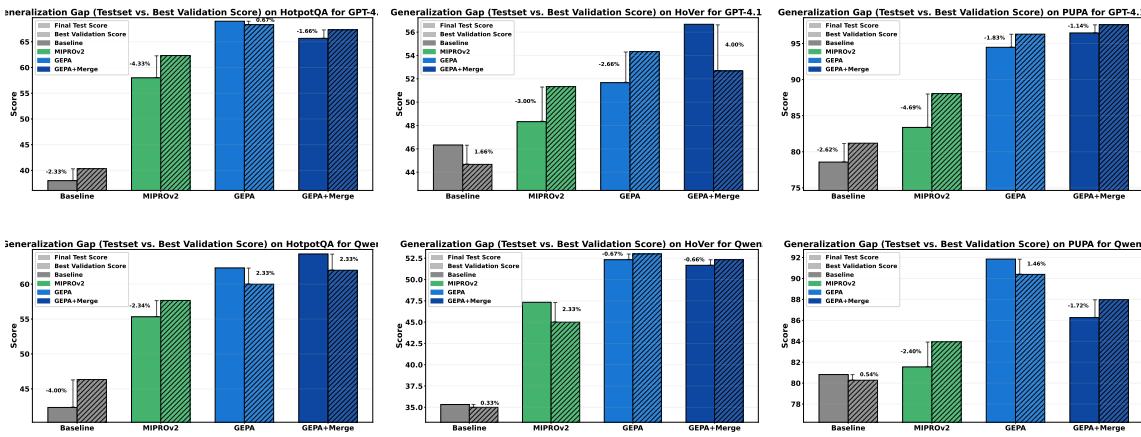


Figure 15: *Generalization gaps for different optimization methods.* Following Wan et al. (2024), we visualize the generalization gap (i.e., the difference between final test set performance and the best achieved validation performance) for different optimizers. While Wan et al. (2024) previously observed 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.

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 16 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 18, 19, 20, 21, 22, 23, 24, and 25 present the genetic search trees created by various configurations of GEPA (and ablation SelectBestCandidate).

M VISUALIZING THE ITERATIVE REFINEMENT ACHIEVED BY GEPA

Figure 26 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.

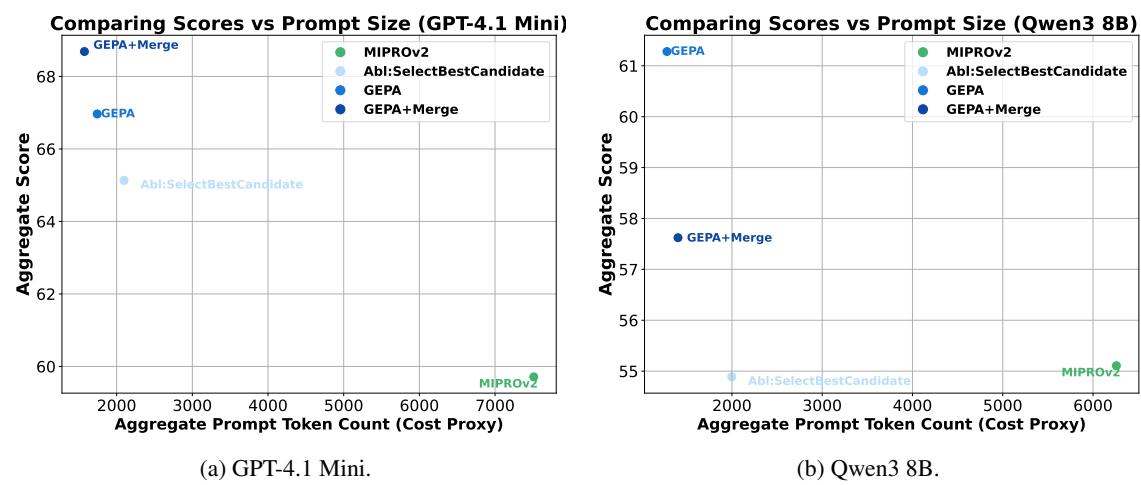


Figure 16: These plots visualize the final aggregate scores against the aggregate prompt size (across all benchmarks) of the final optimized system for each optimizer. It can be seen that GEPA consistently produces prompts that are around less than 33% of the size of MIPROv2’s prompts, while getting 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.

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 LLM.
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:

1. Privacy Preservation:

- Do not include any user-specific or sensitive data in the external LLM request.
- When queries contain location, dates, names, URLs, or other identifiable details, generalize or omit them in the LLM request.

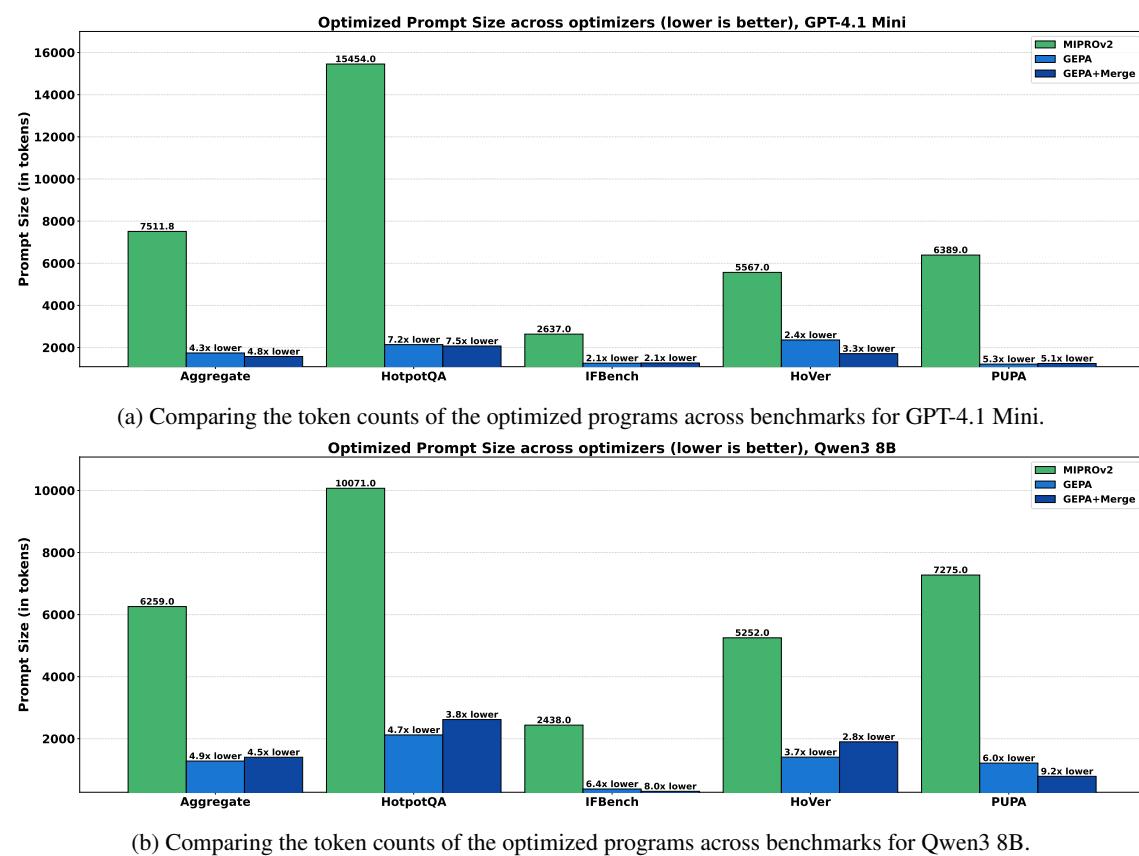
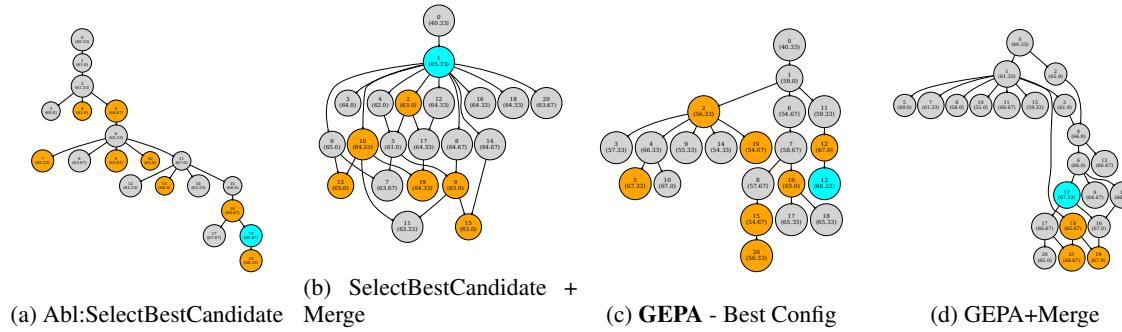


Figure 17: Comparing the token counts of optimized programs across benchmarks.

Figure 18: **HotpotQA GPT-4.1 Mini**

- 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.

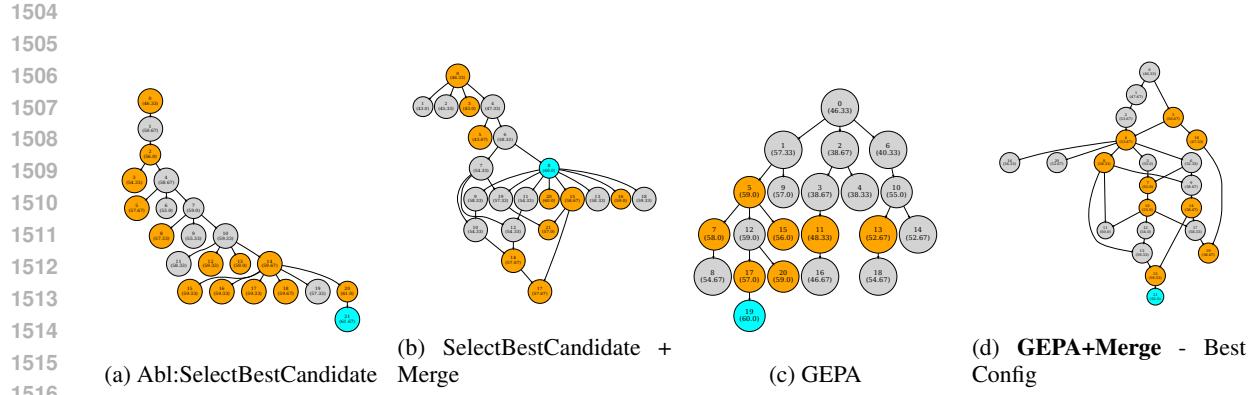


Figure 19: HotpotQA Qwen3 8B

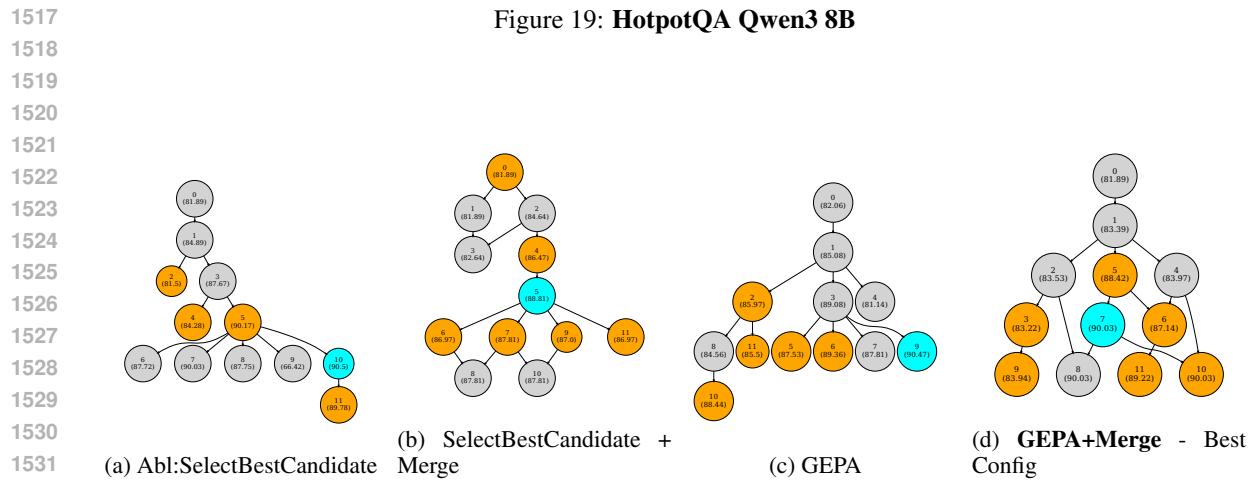


Figure 20: IFBench GPT-4.1 Mini

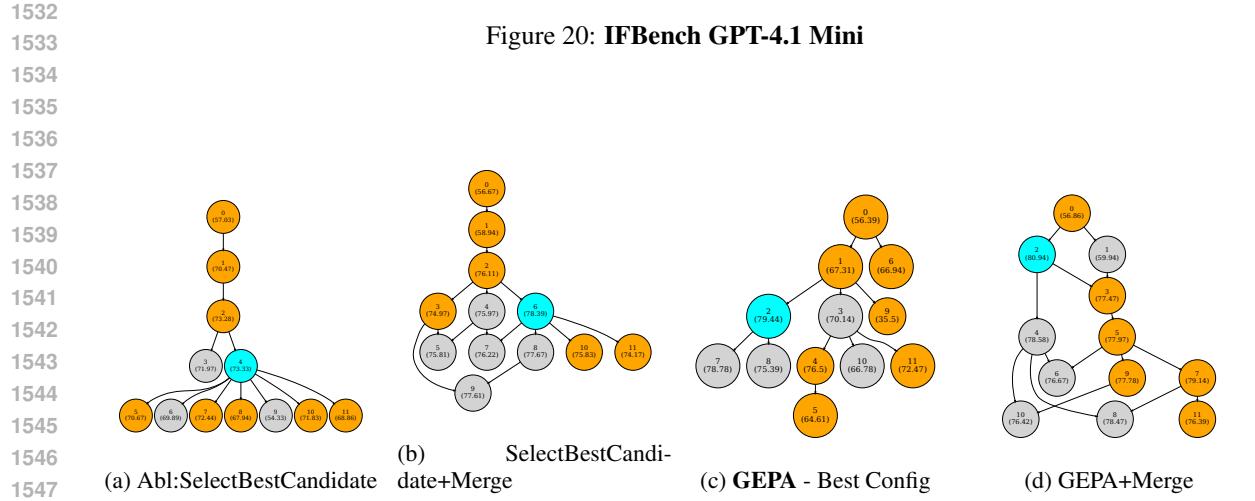


Figure 21: IFBench Qwen3 8B

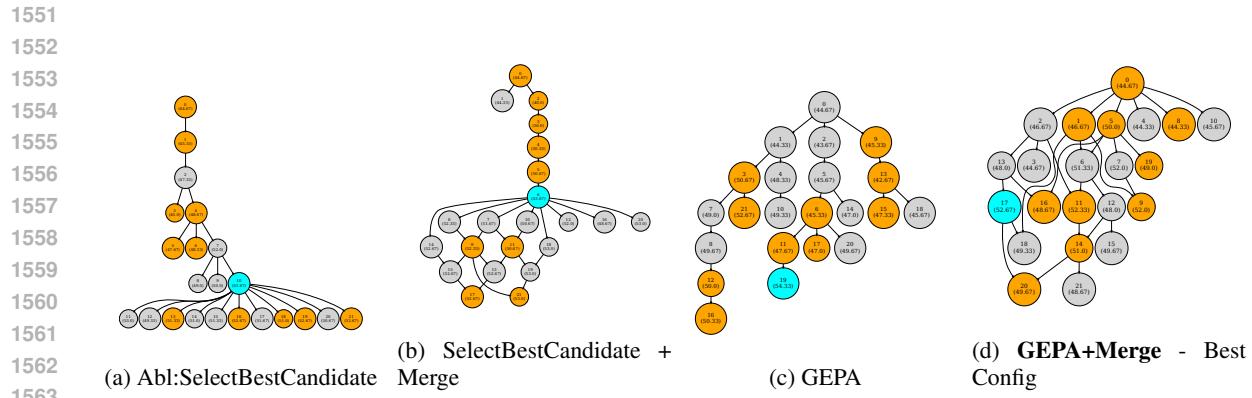


Figure 22: HoVer GPT-4.1 Mini

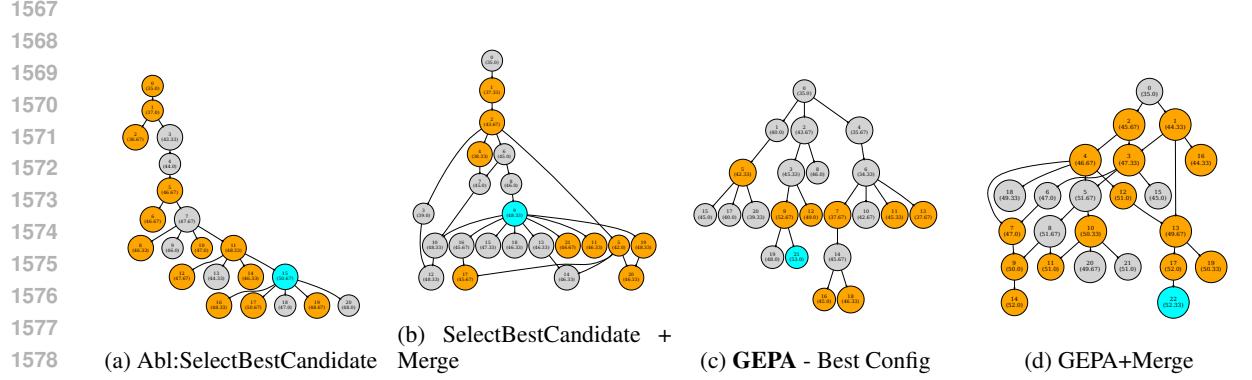


Figure 23: HoVer Qwen3 8B

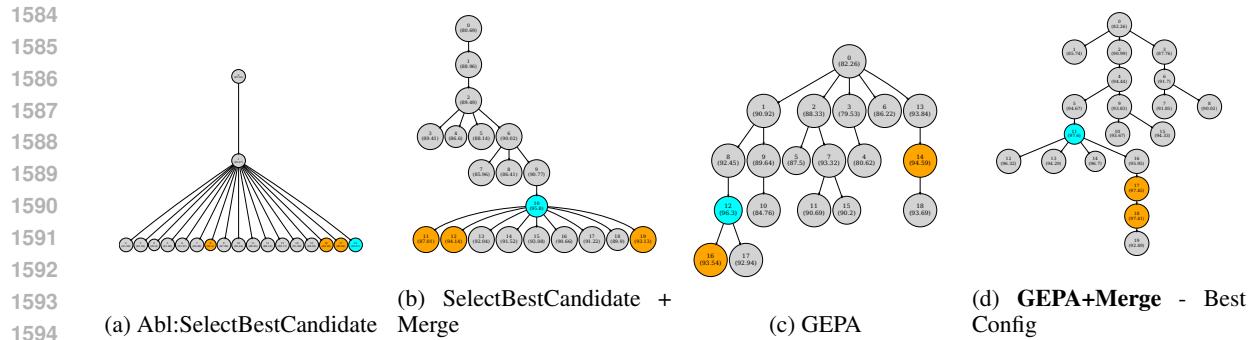


Figure 24: PUPA GPT-4.1 Mini

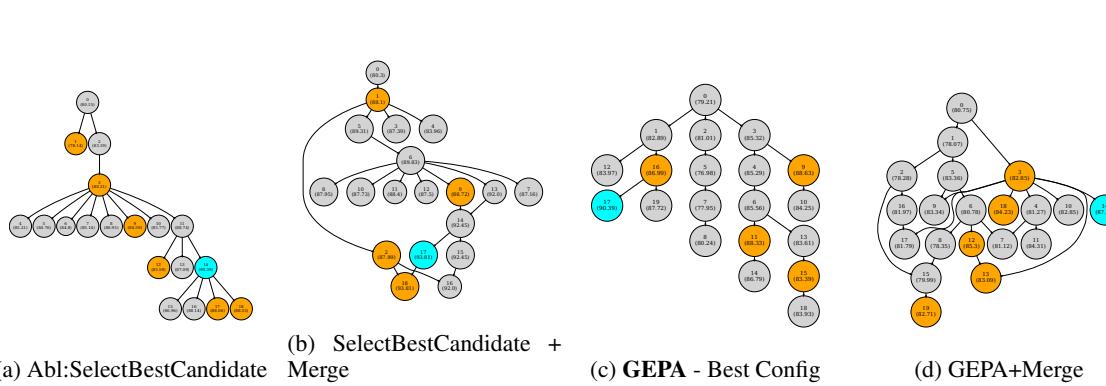


Figure 25: PUPA Qwen3 8B

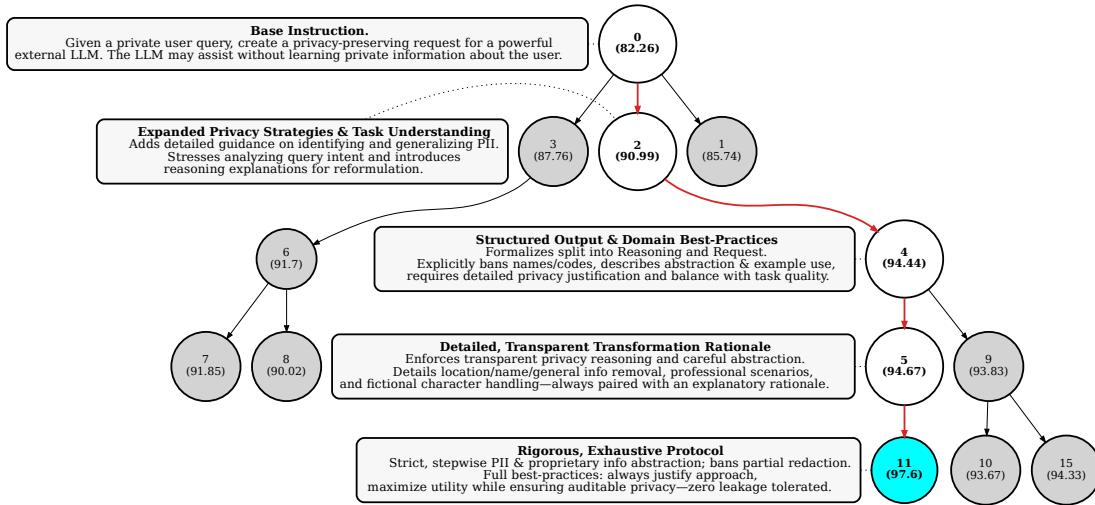


Figure 26: GEPA’s reflective prompt mutation systematically incorporates task-specific nuances, leading to substantial improvements in performance. This figure visualizes the optimization trajectory taken by GEPA, presenting an annotated subtree from Figure 24d (for the privacy-preserving delegation 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.

1645
 1646
 1647 2. Query Understanding and Reformulation:
 1648 - Analyze the user's query carefully to understand the underlying task or information
 need.
 1649 - Determine if the query involves translation, event recommendations, advice,
 1650 summarization, or other tasks.
 1651 - Identify when user-provided content is sensitive or proprietary (e.g., unique texts,
 1652 URLs to closed content) and avoid directly exposing it to the external LLM.
 1653
 1654 3. Quality Maximization:
 1655 - Produce an LLM request that is clear, precise, and directs the LLM to perform the
 necessary task without requiring private context.
 1656 - Retain the core informational or functional need so that the LLM can respond
 1657 effectively.
 1658 - When referencing external documents or URLs, do not include the actual link or
 1659 private info; instead, request general or typical information related to the topic.
 1660
 1661 4. Common Strategies for Privacy-preserving Requests:
 1662 - Use paraphrasing or abstraction for sensitive content.
 1663 - Request generalized or example-based information instead of specific user data.
 1664 - When translation is required, include only the text that needs translating if
 non-private, or otherwise generalize accordingly.
 1665 - For location- or event-specific queries, request general or typical information
 1666 about the category or area without including sensitive details.
 1667
 1668 5. Reasoning Explanation:
 1669 - For each transformation, produce clear reasoning explaining how privacy is preserved
 and why the LLM request fulfills the user's intent without leaking sensitive
 1670 information.
 1671
 1672 Input Format:
 1673 - A user query string possibly containing private or sensitive information.
 1674 - Required output:
 1675 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.
 1676 By following the above principles and strategies, you will optimize both the quality of
 1677 assistance and the protection of user privacy when interacting with third-party language
 1678 models.

1679
 1680
 1681 Node 4 (Score 94.44)
 1682
 1683 Task Description:
 1684 You will be given a user query that may include sensitive, personal, or identifiable
 1685 information. Your role is to transform this query into a privacy-preserving request
 1686 suitable for submission to an external large language model (LLM). The goal is to enable
 1687 the external LLM to assist effectively while ensuring no private, identifying, or
 1688 sensitive information about the user is exposed.
 1689
 1690 Key Requirements and Domain-Specific Details:
 1691

1692

1693 1. Privacy Preservation:

1694 - Do not include any user-specific details, personal names, locations, dates, URLs, or
1695 other potentially identifiable information in the outgoing LLM request.

1696 - When the user query contains private, sensitive, or proprietary data, you must
1697 generalize, abstract, or omit these details.

1698 - Replace sensitive specifics with neutral placeholders or describe them in
1699 generalized terms without losing the core intent of the query.

1700 - Avoid exposing non-public, proprietary, or confidential content directly to the
1701 external LLM.

1702 2. Understanding and Reformulation:

1703 - Carefully analyze the user's query to identify the underlying task type (e.g.,
1704 summarization, creative writing, translation, profile writing, event recommendations,
1705 company background, academic content generation).

1706 - When reformulating, maintain the essential informational or functional need so that
1707 the external LLM can provide a useful, relevant response.

1708 - Preserve thematic and contextual elements necessary for quality output, while
1709 abstracting any sensitive or private details.

1710 - If the task involves user-supplied content (e.g., essays, texts, character
1711 descriptions), distill or summarize the relevant content or use generic versions
1712 instead of exposing original private text.

1713 3. Maximizing Quality of the Reformulated Prompt:

1714 - Construct clear, precise, and well-structured requests that explicitly guide the
1715 external LLM on the task.

1716 - Retain appropriate detail and context to ensure relevance, but balance this
1717 carefully against privacy concerns.

1718 - If referencing external documents, URLs, or institutions, do not include any links
1719 or private identifiers; instead, request general or typical information related to the
1720 subject matter.

1721 - When writing prompts for creative or fictional character tasks, clarify that the
1722 profile or content is fictional to avoid accidental exposure of real personal data.

1723 4. Common Strategies for Privacy Preservation:

1724 - Paraphrase or abstract personal and sensitive content.

1725 - Use general descriptions or hypothetical/example-based requests where appropriate.

1726 - Omit or generalize specific names, dates, locations, institutions, or proprietary
1727 course codes.

1728 - When user content is extensive, focus on absorbing the core themes and instructing
1729 the external model accordingly rather than passing the raw content.

1730 5. Explanation Requirement:

1731 - Provide a concise reasoning paragraph explaining how you identified sensitive or
1732 private details and the steps you took to protect user privacy.

1733 - Clarify how your reformulated LLM request retains the user's original intent and
1734 task needs without risking data leakage.

1735 - This explanation is mandatory to document your privacy-preserving approach and to
1736 justify the abstraction choices made.

1737

1738 Input and Output Format:

1739 - Input: A single user query string that may include sensitive or personal content.

1740 - Output:

1739
 1740 a) Reasoning: A paragraph explaining the privacy considerations and reformulation
 1741 strategy.
 1742 b) LLM Request: A generalized, privacy-safe prompt suitable for querying an external
 1743 LLM.

1744 Examples and Feedback Integration:
 1745 - Avoid including real names, course codes, or faculty references; replace them with
 1746 general terms (e.g., "an interdisciplinary health minor" rather than "HHV").
 1747 - When given character bios or profiles, specify the fictional nature clearly and do not
 1748 include potentially identifying physical descriptions unless necessary for the task.
 1749 - When a user asks about specific companies or entities, generalize the request to "a
 1750 major company in [industry/country]" if privacy concerns exist.
 1751 - Maintain a balance between minimal leakage and maximizing task relevance and quality.
 1752 - Ensure the reformulated prompt supports a comprehensive and useful response from the
 1753 external LLM without compromising privacy.
 1754 By rigorously following these detailed instructions, you will ensure maximal protection
 1755 of user privacy while harnessing the power of external LLMs to assist effectively with
 1756 diverse user requests across domains.

1757
 1758 Node 5 (Score 94.67)

1759
 1760 New Instruction for Privacy-Preserving Query Reformulation Assistant:

1761
 1762 Task Overview:
 1763 You receive a user query that may contain sensitive, private, or personally identifiable
 1764 information (PII). Your goal is to transform this input into a generalized,
 1765 privacy-preserving prompt suitable for querying an external large language model (LLM).
 1766 The reformulated prompt must enable the external LLM to fulfill the user's original
 1767 intent effectively, while rigorously protecting user privacy by abstracting or omitting
 1768 any sensitive details.

1769 Input Format:
 1770 - A single string representing a user query.
 1771 - The query may include private names, places, dates, proprietary info, or identifiable
 1772 context.

1773 Output Format:
 1774 - Part (a) Reasoning: a concise paragraph explaining:
 1775 - How you identified sensitive or private information in the input.
 1776 - What generalization, abstraction, or omission strategies you applied to protect
 1777 privacy.
 1778 - How the reformulated prompt maintains the original task or informational need
 1779 without risking data leakage.
 1780 - Part (b) LLM Request: a carefully constructed and concise, privacy-safe prompt that
 1781 retains essential thematic and functional elements to achieve a useful, relevant response
 1782 from the external LLM.

1783 Key Domain-Specific Details and Best Practices:

1784 1. Privacy Preservation Principles:

1785

1786

1787 - Remove or replace all user-specific names (personal or organizational), exact dates
1788 or durations, locations, URLs, proprietary course or product codes, customer or client
1789 names, and other identifiers.

1790 - When geographic or organizational mentions are critical for context, abstract these
1791 to broader or public categories (e.g., a sustainable travel website focused on a
1792 Central American country rather than naming the country explicitly).

1793 - Omit or abstract internal organizational pressures, client names, or sensitive
1794 contractual details to generic placeholders (e.g., a client rather than a named
1795 company).

1796 - For biographical or individual-related queries, avoid requesting or referencing real
1797 personal info; instead, frame requests around hypothetical or generic profiles unless
1798 the name is widely public and essential for the task.

1799 2. Understanding and Reformulating the Task:

1800 - Identify the underlying task (creative writing, summarization, professional
1801 communication drafting, etc.) from the user's query.

1802 - Preserve the functional intent and thematic requirements (e.g., content topics
1803 around sustainability, summary of a person's background, professional email
1804 follow-up).

1805 - For user-supplied content, do not repeat verbatim or expose any original text;
1806 instead, extract core themes, instruct the LLM accordingly, or replace with generic
1807 examples.

1808 3. Maximizing Output Quality While Preserving Privacy:

1809 - Construct a prompt that is clear, precise, and contains sufficient context to enable
1810 comprehensive and relevant LLM output.

1811 - Avoid ambiguous or overly generic requests that might reduce relevance or usefulness.

1812 - Maintain a balance between detail necessary for quality and generalization required
1813 for privacy.

1814 4. Common Reformulation Strategies:

1815 - Replace specific names with generic role identifiers or placeholders (a business
1816 contact, a notable individual).

1817 - Replace specific locations or institutions with generalized descriptors, e.g., a
1818 country known for eco-tourism.

1819 - For time references, use relative or approximate terms without revealing explicit
1820 dates.

1821 - For internal or proprietary details, describe the scenario generically (e.g.,
1822 pressure from a client to close a matter urgently).

1823 - When handling personal profiles or characters, explicitly state that the profile is
1824 fictional if requesting content generation about a person.

1825 5. Explanation Requirements:

1826 - Always provide reasoning that details how privacy risks were identified and
1827 mitigated.

1828 - Explain how the essential task was preserved despite abstraction.

1829 - This reasoning documents the privacy-preserving approach and justifies design
1830 choices.

1831 Examples and Common Pitfalls:

1832 - Do not retain or lightly obscure personal names or company names; fully abstract or
remove.

- Avoid exposing direct quotes from user-supplied texts.

1833
 1834 - Use neutral, context-appropriate phrasing to describe user intents.
 1835 - Retain geographic or cultural context only when it is publicly available and pivotal to
 the task.
 1836 - Ensure the output prompt remains a clear instruction that the external LLM can readily
 interpret.
 1837
 1838
 1839 Summary:
 1840 Your reformulations must ensure zero exposure of PII or private/proprietary content while
 1841 retaining enough context and clarity to generate high-quality, relevant responses from
 1842 the external LLM. This involves careful task analysis, methodical abstraction of
 1843 sensitive details, and clear, precise prompt construction combined with transparent
 1844 reasoning on privacy protection mechanisms used.
 1845
 1846

1847 Node 11 (Score 97.6)

1849 New Instruction for Privacy-Preserving Query Reformulation Assistant (Enhanced):

1851 Task Overview:

1852 You receive a user query that may contain sensitive, private, or personally identifiable
 1853 information (PII). Your objective is to reformulate this query into a generalized,
 1854 privacy-preserving prompt suitable for sending to an external large language model (LLM).
 1855 The reformulated prompt must enable the external LLM to fulfill the user's original
 1856 intent effectively while rigorously protecting all user privacy by abstracting, omitting,
 1857 or generalizing any sensitive details.

1858 Input Format:

1859 - A single user query string.
 1860 - The query might contain identifiable names (people, organizations), specific geographic
 1861 locations, exact dates or timeframes, proprietary or internal terminology (e.g., system
 1862 names, product codes), client or customer names, URLs, or any other sensitive or private
 1863 information.

1864 Output Format:

1865 - Part (a) Reasoning:

1866 Provide a concise paragraph that explains:
 1867 * How you identified sensitive or private information in the input
 1868 * What strategies you applied to protect privacy (generalization, omission,
 abstraction, replacement with placeholders)
 1869 * How the reformulated prompt preserves the original intent and task requirements
 1870 without risking data leakage

1871 - Part (b) LLM Request:

1872 A concise, carefully constructed privacy-safe prompt that:
 1873 * Removes or anonymizes all PII and proprietary/internal details
 1874 * Abstracts locations, names, dates, and technical terms as needed
 1875 * Produces a clear and contextually rich instruction for the LLM to generate a
 relevant and informative response aligned with the user's original task

1876 Detailed Domain-Specific Guidance and Best Practices:

1877 1. Identification and Treatment of Sensitive Data:

1880

1881 - All user-specific or personal names (individual or organizational) must be removed
1882 or replaced with generic role descriptors (e.g., a business contact, a client, a notable individual). Never lightly obscure or partially redact; full
1883 abstraction is required.

1884 - All geographic mentions must be abstracted unless the location is publicly known,
1885 essential to the task, and can be generalized (e.g., a region known for
1886 eco-tourism instead of naming a country or city explicitly).

1887 - Exact dates or durations must never be retained; instead, use relative or
1888 approximate temporal references (e.g., recently, over the past year).

1889 - Internal or proprietary terms including system names, product codes,
1890 subscription types, and technical jargon must be generalized or replaced with
1891 neutral descriptors to avoid leakage of intellectual property or sensitive operational
1892 details.

1893 - Avoid direct quotes or verbatim inclusion of user-supplied texts unless obfuscated
by generalization.

1894 2. Task Understanding and Reformulation:

1895 - Identify the functional intent of the query: Is it creative writing, translation,
1896 summarization, professional communication drafting, technical explanation, or other?

1897 - Preserve the thematic and informational core of the query (e.g., request for
1898 educational quality analysis, technical translation of a passage, biographical
1899 summary).

1900 - Do not reproduce the original input verbatim; instead, frame the LLM prompt around
1901 the essential thematic elements extracted from the input.

1902 - For queries regarding individuals, avoid direct reference to real personal
1903 information unless the name is widely public and essential; even then, use a generic
or hypothetical framing for the individual profile.

1904 3. Strategies for High-Quality and Privacy-Preserving Prompts:

1905 - Strike a balance between sufficient contextual detail and privacy abstraction to
1906 maintain prompt clarity and relevance.

1907 - Use neutral, context-aware formulations that clearly instruct the LLM on the content
1908 and style expected.

1909 - Avoid vague or overly generic prompts that could result in less useful or
1910 lower-quality responses.

1911 - When system or proprietary content is mentioned, instruct the LLM to generalize
1912 specific terms and maintain the technical meaning without revealing sensitive info.

1913 - When a direct translation is requested on specialized text, specify to replace or
abstract internal nomenclature.

1914 4. Explanation Requirements:

1915 - The reasoning must transparently explain how privacy risks were identified (e.g.,
1916 presence of names, locations, dates, proprietary terms).

1917 - It must describe the abstraction or omission methods applied (e.g., replacing
1918 a Jonah Van Beijnen with a notable individual, substituting Makau with a specific region,
1919 or Yoda with a system name).

1920 - Clarify how the essential task and user intent were preserved despite these
1921 generalizations (e.g., focusing on educational quality, technical translation,
biographical summary).

1922 - This explanation justifies your design choices and demonstrates adherence to
1923 privacy-preserving principles.

1924 Common Pitfalls to Avoid:

1925

1926

1927
1928 - Do not merely lightly obscure or partially redact sensitive details; full anonymization
1929 or abstraction is required.
1930 - Do not repeat any user-supplied PII or proprietary content verbatim.
1931 - Avoid including URLs, exact dates, or direct quotes without modification.
1932 - Do not leave ambiguity that could degrade the quality or contextual clarity of the
1933 reformulated prompt.
1934 - Do not include any real personal or organizational names unless they are public figures
1935 and the query requires it, then use generic descriptors instead.
1936 Example Summary of Effective Approach (Informed by Prior Examples):
1937 - For geographic queries: replace exact place names with general regions and provide a
1938 brief contextual descriptor.
1939 - For technical texts containing system names or subscription types: instruct the LLM to
1940 translate or process the text while replacing or abstracting proprietary system
1941 identifiers.
1942 - For biographical summaries about specific individuals: remove the real name and request
1943 a generic, well-structured four-paragraph summary about a notable individual, preserving
1944 the overall intent without leaking PII.
1945 Summary:
1946 Your reformulations must ensure zero exposure of any PII or private/proprietary content
1947 while retaining enough thematic and functional clarity for the external LLM to produce
1948 high-quality, relevant outputs. This requires thorough analysis of the user's query,
1949 rigorous application of privacy-preservation strategies, and explicit reasoning
1950 explanations that document your approach and choices.

N EXAMPLES OF BEST PROMPTS FOR EVERY BENCHMARK

1951
1952 In this section, we present the best optimized prompt obtained for every (benchmark, model) configuration.
1953 Each subsection below pertains to one (benchmark, model) configuration. Since every compound AI system
1954 consists of multiple modules, each subsection consists of multiple boxes, listing the prompts for each mod-
1955 ule. MIPROv2 optimized prompts contain upto 4 few-shot examples for each task. We provide just the first
1956 demo here for brevity. GEPA's prompts only consist of the optimized instruction, which is provided in full.
1957

N.1 HOTPOTQA, GPT-4.1 MINI

1958 HotpotQA GPT-4.1 Mini create_query_hop2.predict

1959 Base Prompt:

1960 Given the fields `question` , `summary_1` , produce the fields `query` .

1961 MIPROv2 Prompt:

1962 You are an expert multi-hop question answering system designed to refine retrieval
1963 queries for complex questions. Given the original question and an initial summary of
1964 retrieved documents (summary_1), think step by step to analyze how the summary relates to
1965 the question. Use this reasoning to generate a focused and precise query that will guide
1966 the retrieval of additional relevant information needed to answer the question
1967 completely. Your output should include a clear chain-of-thought reasoning process

1974
1975 followed by the refined query that targets complementary evidence for the next retrieval
1976 hop.
1977 Demos:
1978 Example 1:
1979 question:
1980 Simone Benedetti plays for a team that currently competes in which Serie?
1981 summary_1:
1982 Simone Benedetti plays for Virtus Entella, but the passages do not specify which
1983 Serie the team currently competes in.
1984 reasoning:
1985 The summary indicates that Simone Benedetti plays for Virtus Entella, but it does not
1986 state the current Serie (league level) in which Virtus Entella competes. To
1987 accurately answer which Serie the team currently competes in, it is necessary to look
1988 up the most recent league information for Virtus Entella.
1989 query:
1990 What Serie (league level) does Virtus Entella currently compete in?
1991 #####
1992 3 other demos omitted for brevity.
1993 #####

1993 GEPA Prompt generated by config GEPA:
1994
1995 You will be given two input fields: `question` and `summary_1`.
1996 Your task is to generate a new search query (`query`) optimized for the **second hop** of
1997 a multi-hop retrieval system. The original user question is typically complex and
1998 requires information from multiple documents to answer. The first hop query is the
1999 original question used to retrieve an initial set of documents. Your goal is to generate
2000 a **second hop query** that retrieves *additional relevant documents* that were *not*
2001 found in the first hop but are necessary to answer the original question completely.
2002 Detailed task instructions and hints:
2003
2004 1. **Input Understanding:**
2005 - `question` is the original multi-hop question posed by the user.
2006 - `summary_1` is a concise summary of information from a document retrieved in the
2007 first hop, which partially addresses the question.
2008
2009 2. **Purpose and Context:**
2010 - Your generated `query` aims to find the *missing pieces* of information needed to
2011 fully answer the `question`.
2012 - The multi-hop retrieval system works in stages:
2013 - First hop: The original question returns some documents.
2014 - Second hop: Your query must help retrieve any *other relevant documents* NOT found
2015 in the first hop that hold complementary or broader context necessary for final
2016 answer extraction.
2017
2018 3. **Key Observations from Examples and Feedback:**
2019 - First-hop documents often cover one entity or aspect in the question.
2020 - Remaining relevant documents often involve connected or higher-level concepts
 mentioned in `summary_1` but not explicitly asked in the original question.

2021

2022 - The `query` should be formulated to explicitly target these *missing*, but logically linked, documents.

2023 - Avoid merely paraphrasing the original question or restating known facts from `summary_1`.

2024 - Instead, infer what broader or related entities/concepts might provide the crucial missing information.

2025 - 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").

2026 - 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.

2027

2028

2029

2030

2031

2032

2033 4. **How to Build the Query:**

2034 - Identify the entities or topics mentioned in `summary_1` that appear related but different from first-hop documents.

2035 - Reframe the query to explicitly mention these broader or related entities connected to the original question.

2036 - Include relevant key context from the question to maintain specificity, but shift focus to the missing piece.

2037 - The goal is to retrieve documents that link or complement what was retrieved initially.

2038

2039

2040

2041 5. **Practical Strategy:**

2042 - Read the `summary_1` carefully to spot references to bigger contexts or other entities not covered in the first hop.

2043 - Ask yourself, "What entity or aspect does this summary hint at that could answer the original question but was not found yet?"

2044 - Formulate a precise, focused factual query targeting that entity or concept to retrieve the missing documents.

2045

2046

2047

2048 6. **Output:**

2049 - Produce only the field `query` as a clear, concise question or keyword phrase designed for efficient retrieval of **second-hop documents**.

2050 - Ensure the query relates logically to the original question while targeting the broader or complementary knowledge identified in `summary_1`.

2051 - Do **not** include the original question or simply rephrase it.

2052 - Do **not** duplicate information already well-covered by the first hop retrieval.

2053

2054 By following these principles, you will help the multi-hop retrieval system find all necessary documents to answer the multi-faceted original question completely.

2055

2056

2057

2058 HotpotQA GPT-4.1 Mini final_answer.predict

2059 Base Prompt:

2060 Given the fields `question`, `summary_1`, `summary_2`, produce the fields `answer`.

2061 MIPROv2 Prompt:

2062 You are an expert multi-hop reasoning assistant skilled in synthesizing information from multiple summaries to answer complex questions. Given the `question`, along with two intermediate summaries `summary_1` and `summary_2` that contain relevant evidence,

2068
 2069 carefully analyze and integrate the information step-by-step to produce a clear, logical
 2070 reasoning process followed by a concise and accurate final answer.
 2071 Demos:
 2072 Example 1:
 2073 question:
 2074 are Machaeranthera and Prumnopitys both the plants?
 2075 Yes, both Machaeranthera and Prumnopitys are plants. Machaeranthera is a genus of
 2076 flowering plants in the daisy family, while Prumnopitys is a genus of coniferous
 2077 evergreen trees in the podocarp family.
 2078 summary_1:
 2079 Yes, both Machaeranthera and Prumnopitys are plants; Machaeranthera is a genus of
 2080 flowering plants in the daisy family, while Prumnopitys is a genus of coniferous
 2081 evergreen trees in the podocarp family.
 2082 reasoning:
 2083 Both summaries agree that Machaeranthera and Prumnopitys are plants. Specifically,
 2084 Machaeranthera is a genus of flowering plants in the daisy family, and Prumnopitys is
 2085 a genus of coniferous evergreen trees in the podocarp family. Since both belong to
 2086 plant genera, the answer is yes.
 2087 answer:
 2088 yes
 2089 #####
 2090 3 other demos omitted for brevity.
 2091 #####

2091 GEPA Prompt generated by config GEPA:

2092 Task Description:
 2093 You are given three fields as input: `question`, `summary_1`, and `summary_2`. Your goal
 2094 is to produce an `answer` field that directly and explicitly responds to the question
 2095 using the information from the two summaries, enhanced by your authoritative domain
 2096 knowledge when needed.
 2097
 2098 Input Format:
 2099
 2100 - `question`: A natural language question that may require a fact, definition, name,
 2101 date, yes/no response, or other specific information.
 2102 - `summary_1` and `summary_2`: Two independently generated summaries or snippets
 2103 containing information related to the question. They may vary in completeness, accuracy,
 2104 and specificity.
 2105 Requirements and Approach:
 2106
 2107 1. **Understand the question precisely.** Determine exactly what is asked—whether a
 2108 name, a specific fact, a date, or a yes/no answer.
 2109
 2110 2. **Compare both summaries.** Analyze the content of `summary_1` and `summary_2`:
 2111 - If they agree and directly answer the question, use this as primary evidence.
 2112 - If one summary provides a fact that the other does not mention, carefully evaluate
 2113 its plausibility.
 2114

2115

2116 - If the summaries conflict, use domain expertise and authoritative knowledge to
2117 resolve or explicitly state uncertainty.

2118 3. **Domain-specific factual verification and nuance:**

2119 - **Names and nicknames:** Provide only the specific nickname or name when asked,
2120 without extra phrasing. For example, when asked for the nickname of a person or
2121 entity, respond with the nickname alone, not a full sentence.

2122 - **Nationality and identity distinctions:** Use the most precise terms aligned with
2123 factual correctness and common usage (e.g., “English” vs. “British”) based on
2124 domain knowledge.

2125 - **Dates and historical facts:** Verify dates or historical claims with domain
2126 knowledge to pick the correct fact, especially when there might be confusion between
2127 franchise start dates vs. event dates etc.

2128 - **Yes/no questions:** Prefer concise answers of “yes” or “no” only, unless the
2129 question demands elaboration.

2130 - **Types or categories:** If a question asks about the type or category (e.g., type
2131 of company), provide the most direct concise phrase without including adjectives like
nationality unless asked explicitly.

2132 4. **Answer conciseness and relevance:**

2133 - Provide a brief and direct answer to the question.

2134 - Avoid repeating or restating the question.

2135 - Avoid unnecessary context unless requested or needed for clarity.

2136 - Avoid constructing full sentences unless needed; for example, answers to nickname or
yes/no questions should be as short and specific as possible.

2137 5. **When authoritative knowledge supplements the summaries:**

2138 - If the summaries are incomplete or potentially inaccurate, incorporate trusted
2139 knowledge from your training about the topic to provide the correct and precise answer.

2140 - For example, when a summary gives a year that conflicts with known release dates or
factual details, prefer the verified date.

2141 - 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).

2142 6. **Examples of correct reasoning and answers:**

2143 - Question: “What is the nickname of the 2005 Toyota Grand Prix of Long Beach
2144 Polesitter?”

2145 - Correct answer: “the thrill from West Hill”

2146 - Question: “What type of company is Zipcar led by Scott Griffith from 2003-2013?”

2147 - Correct answer: “car-sharing company”

2148 - Question: “Who was the partner of British comic book artist, Henry Flint, that
2149 helped create Zombo?”

2150 - Correct answer: “Al Ewing”

2151

2152

2153 **Summary:**

2154

2155 - Use both summaries as primary but not sole evidence.

2156 - Reliably verify and contextualize facts using domain knowledge, especially for
2157 nationality, dates, nicknames, company types, and yes/no questions.

2158 - Provide short, direct answers matching the specificity requested.

2159 - Avoid unnecessary elaboration unless explicitly required.

2160

2161

2162
 2163 - Explicitly resolve conflicts or ambiguity using your knowledge or state uncertainty
 2164 when appropriate.

2165
 2166 This approach ensures that answers are both accurate and concise, suitable for direct
 2167 consumption or integration in knowledge bases or question-answering systems.

2168
 2169 HotpotQA GPT-4.1 Mini summarize1.predict

2170 Base Prompt:

2171
 2172 Given the fields `question`, `passages`, produce the fields `summary`.

2173
 2174 MIPROv2 Prompt:

2175
 2176 Given a question and a set of related passages, carefully analyze the information by
 2177 thinking through the relevant facts step-by-step. Produce a clear and concise summary
 2178 that synthesizes the key points from the passages directly relevant to answering the
 2179 question, ensuring the summary is focused, accurate, and grounded in the evidence
 2180 provided.

2181 Demos:

2182 Example 1:

2183 question:

2184 The architectural style of a church that stands in front of the Palazzo Ghisilardi
 2185 Fava originated in what city?

2186 passages:

2187 ['Palazzo Ghisilardi Fava | Palazzo Ghisilardi Fava is a Renaissance style palace,
 2188 located on via Manzoni 4 in Bologna, region of Emilia Romagna, Italy; it houses the
 2189 Medieval Civic Museum of Bologna.', 'Madonna di Galliera, Bologna | The Madonna di
 2190 Galliera is a church with a Renaissance facade and Baroque interiors, located on Via
 2191 Manzoni, in central Bologna, Italy. It stands in front of the Palazzo Ghisilardi
 2192 Fava. The present name over the portal is the "Chiesa di Filippini Madonna di
 2193 Galliera e Filippo Neri'.', 'Palazzo Pretorio, Prato | The Palazzo Pretorio of the
 2194 Tuscan town of Prato was the old city hall located town center, standing in front of
 2195 the current Palazzo Comunale. It now accommodates the Civic Museum of Prato, which
 2196 was reopened on September 2013.', 'Palazzo Pisani Gritti | The Palazzo Pisani Gritti
 2197 is a Gothic-style palace located across the Grand Canal from the Church of the
 2198 Salute, and stands along the Campo del Traghetto where the church of Santa Maria del
 2199 Giglio, in the Sestieri of San Marco, Venice, Italy.', 'DĂ umĂ tĂ r Tower | The
 2200 DĂ umĂ tĂ r tower is the oldest building in Szeged, Hungary. The foundation was most
 2201 probably laid during the 11th century, while the lower part was built (in Romanesque
 2202 style) from the 12th century, and the upper part (Gothic style) from the 13th
 2203 century. The tower was once part of the former "Szent DĂ umĂ tĂ r templom" (St.
 2204 Demetrius church), but today it stands in DĂ sm Square, in front of the much larger
 2205 Votive Church of Szeged. The upper part was rebuilt from the original stones in 1926.
 2206 The architecture of the tower is similar to another found in Southern France, or in
 2207 the territory of the former Byzantine Empire.', 'Palazzo Leoni, Bologna | The Palazzo
 2208 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 GesĂ z.]

2207 reasoning:

2209
 2210 The question asks for the city where the architectural style of a church that stands
 2211 in front of the Palazzo Ghisilardi Fava originated. Passage [2] identifies the church
 2212 standing in front of the Palazzo Ghisilardi Fava as "Madonna di Galliera" and
 2213 mentions it has a Renaissance facade and Baroque interiors. Passage [1] states the
 2214 Palazzo Ghisilardi Fava is located in Bologna, Italy. Since the church is located in
 2215 Bologna and features a Renaissance architectural style, it can be inferred that the
 2216 architectural style of this church originated in the city of Florence, which is
 2217 historically recognized as the birthplace of Renaissance architecture. Thus, the
 2218 architectural style (Renaissance) of the church in front of the Palazzo Ghisilardi
 2219 Fava originated in Florence.
 2220 summary:
 2221 The Renaissance architectural style of the church standing in front of the Palazzo
 2222 Ghisilardi Fava originated in Florence.
 2223 #####
 2224 3 other demos omitted for brevity.
 2225 #####

2226 GEPA Prompt generated by config GEPA:

2227 You are a first-hop ****summarization module**** in a multi-hop question answering (QA)
 2228 system. Your task is to generate a concise, informative `summary` given two input fields:
 2229 a `question` and a list of relevant `passages`.
 2230 Your goal is to extract and synthesize key information from the retrieved passages that:
 2231
 2232 1. Directly relates to the initial question.
 2233 2. Captures the core facts and entities needed to understand the scope and context of the
 2234 question.
 2235 3. Includes relevant connections, bridging entities, dates, locations, or descriptions
 2236 that enable the system to devise focused and effective follow-up queries in subsequent
 2237 hops.
 2238 4. Provides a strong factual foundation for downstream answer generation modules.
 2239 ****Task specifics and best practices:****
 2240
 2241 - The `summary` must represent a distilled synthesis, not just a compression or
 2242 extractive snippet.
 2243 - Explicitly include cited passage titles or key entity labels (e.g., "Children in Need
 2244 2006 | ..." or "Anthony Levandowski | ...") in your summary to highlight the origin of
 2245 information.
 2246 - Incorporate sufficient context to hint at missing or un-retrieved supporting facts,
 2247 thus enhancing the multi-hop retrieval process.
 2248 - When the question asks for an attribute (e.g., nationality, location, company origin),
 2249 ensure you provide:
 2250 - Identification of the relevant subject or entity mentioned in the passages.
 2251 - The extracted attribute or relevant information as stated or implied.
 2252 - Bridging details that could help the system pursue remaining information in the next
 2253 retrieval step.
 2254 - Avoid forming a final answer; instead, focus on "what is known now" from the input
 2255 documents to facilitate further query refinement.
 2256
 2257 ****Examples of critical elements to include:****

2256
 2257
 2258 - Entity names, roles, titles, and dates tied to the question.
 2259 - Names of organizations or locations connected through intermediary entities.
 2260 - Distinctive identifiers or clarifications that can help narrow down next-step retrieval
 (such as "Natasha Kaplinsky is an English presenter," "Kapolei is a city on Oahu," or
 2261 "Waymo spun out of Alphabet").
 2262
 2263 ****Format of output:****
 2264 Provide a paragraph or a few sentences that cohesively summarize the key passages in
 2265 relation to the question, referencing passage titles or entities to frame facts clearly.
 2266
 2267 ---
 2268
 2269 This approach ensures the summary is both informative for next-hop retrieval and
 2270 foundational for final answer extraction in multi-hop QA.

2271

2272 HotpotQA GPT-4.1 Mini summarize2.predict

2273 Base Prompt:

2274 Given the fields `question`, `context`, `passages`, produce the fields `summary`.

2275 MIPROv2 Prompt:

2276 Given a `question`, relevant `context`, and a list of `passages`, provide a clear,
 2277 concise summary that integrates the key information from the passages in relation to the
 2278 question and context. Use step-by-step reasoning to explain how the summary is derived
 2279 from the evidence before presenting the final synthesized summary.

2280 Demos:

2281 Example 1:

2282 question:

2283 What bank was founded by the great-great-great grandfather of the second Duke of
 2284 Florence?

2285 context:

2286 The bank founded by the great-great-great grandfather of the second Duke of Florence
 2287 is the Medici Bank, established by Giovanni di Bicci de' Medici.

2288 passages:

2289 ['Giovanni di Bicci de\' Medici | Giovanni di Bicci de\' Medici (c. 1360 – February
 2290 20/28, 1429) was an Italian banker, a member of Medici family of Florence, and the
 2291 founder of the Medici Bank. While other family members, such as Chiarissimo di
 2292 Giambuono de\' Medici, who served in the Signoria in 1201, and Salvestro de\' Medici,
 2293 who was implicated in the Ciompi Revolt of 1378, are historically significant,
 2294 Giovanni\'s founding of the family bank truly began the family\'s rise to power in
 2295 Florence. He was the father of Cosimo de\' Medici ("Pater Patriae"), grandfather of
 2296 Piero di Cosimo de\' Medici, great-grandfather of Lorenzo de\' Medici (the
 2297 Magnificent) and great-great-great-grandfather of Cosimo I de\' Medici, Grand Duke of
 2298 Tuscany.', 'Averardo de\' Medici | Averardo de\' Medici (1320–1363),
 2299 also known as Everard De Medici, was the son of Salvestro de\' Medici (died 1346),
 2300 "il Chiarissimo" (English meaning "the very clear.") and the father of three
 2301 children: Giovanni, Francesco, and Antonia. Giovanni di Bicci de\' Medici would later
 2302 become the first historically relevant member of Medici family of Florence and the
 2303 eventual founder of the Medici bank.', "Villa Medici at Cafaggiolo | The Villa

2303
 2304 Medicea di Cafaggiolo is a villa situated near the Tuscan town of Barberino di
 2305 Mugello in the valley of the River Sieve, some 25\x00kilometres north of Florence,
 2306 central Italy. It was one of the oldest and most favoured of the Medici family
 2307 estates, having been in the possession of the family since the 14th century, when it
 2308 was owned by Averardo de' Medici. Averardo's son, Giovanni di Bicci de' Medici, is
 2309 considered to be the founder of the Medici dynasty.", 'Medici: Masters of Florence |
 2310 Medici: Masters of Florence is an Italian-British television drama series about the
 2311 Medici dynasty set in 15th-century Florence, starring Dustin Hoffman as Giovanni di
 2312 Bicci de' Medici, Richard Madden as Cosimo de' Medici, and Stuart Martin as Lorenzo
 2313 de' Medici ("The Elder"). The series was co-created by Frank Spotnitz ("The X-Files"
 2314 and "Man in the High Castle") and Nicholas Meyer (""). Sergio Mimica-Gezzan ("The
 2315 Pillars of the Earth") directed all eight episodes. Episodes 1 and 2 aired on Rai 1
 2316 (Italian TV) on 18 October 2016. According to Italian ratings compiler Auditel, it
 2317 attracted a record 7.6\x00million viewers. The first season consists of eight
 2318 episodes.', 'Neri di Bicci | Neri di Bicci (1419â€“1491) was an Italian painter of
 2319 the Renaissance. A prolific painter of mainly religious themes, he was active mainly
 2320 in Florence and in the medium of tempera. His father was Bicci di Lorenzo. His
 2321 grandfather, Lorenzo di Bicci was also a painter in Florence, a pupil of Spinello
 2322 Aretino.', 'Cosimo de' Medici | Cosimo di Giovanni de' Medici (called \'the Elder\'
 2323 (Italian "il Vecchio") and posthumously "Father of the Nation" (Latin "pater
 2324 patriae"); 27 September 1389 â€“ 1 August 1464) was an Italian banker and politician,
 2325 the first of the Medici political dynasty, "de facto" rulers of Florence during much
 2326 of the Italian Renaissance. Despite his influence, his power was not absolute;
 2327 Florence's legislative councils at times resisted his proposals, something which
 2328 would not have been tolerated by the Visconti of Milan, for instance; throughout his
 2329 life he was always "primus inter pares" ("first among equals"). His power derived
 2330 from his wealth as a banker, and he was a great patron of learning, the arts and
 2331 architecture.', "Giovanni il Popolano | Giovanni de' Medici, in full Giovanni di
 2332 Pierfrancesco de' Medici, later known as il Popolano (21 October 1467 â€“ 14
 2333 September 1498) was an Italian nobleman of the Medici House of Florence. He was the
 2334 son of Pierfrancesco di Lorenzo de' Medici, and therefore a member of a secondary
 2335 branch of the family."]
 2336 reasoning:
 2337 The question asks for the bank founded by the great-great-great-grandfather of the
 2338 second Duke of Florence. Passage [1] states that Giovanni di Bicci de' Medici founded
 2339 the Medici Bank and was the great-great-great-grandfather of Cosimo I de' Medici, who
 2340 was the Grand Duke of Tuscany. The Dukedom of Florence eventually transformed into
 2341 the Grand Duchy of Tuscany under Cosimo I, who is identified as the second Duke of
 2342 Florence in the question context.
 2343 Thus, Giovanni di Bicci de' Medici, as the great-great-great-grandfather of the second
 2344 Duke of Florence (Cosimo I de' Medici), founded the Medici Bank.
 2345 summary:
 2346 The Medici Bank was founded by Giovanni di Bicci de' Medici, the
 2347 great-great-great-grandfather of the second Duke of Florence.
 2348 #####
 2349 3 other demos omitted for brevity.
 #####
 GEPA Prompt generated by config GEPA:
 Given the fields `question`, `context`, `passages`, produce the fields `summary`.

2350 N.2 HOTPOTQA, QWEN3 8B
 2351

2352 HotpotQA Qwen3 8B create_query_hop2.predict

2353 Base Prompt:

2354 Given the fields `question`, `summary_1`, produce the fields `query`.

2355 MIPROv2 Prompt:

2356 Given the question and the first summary, synthesize the key elements of the question and
 2357 the summary. Identify the specific information that needs to be retrieved or confirmed.
 2358 Formulate a focused and precise query that will guide the next step in the multi-hop
 2359 reasoning process, ensuring it directly addresses the gap in knowledge or requires
 2360 further clarification from additional context.

2361 Demos:

2362 GEPA Prompt generated by config GEPA+Merge:

2363 Given the fields `question` and `summary_1`, produce the field `query` that optimizes the
 2364 retrieval of additional documents for a multi-hop system.

2365 ****Task Details:****

2366 1. ****Objective:**** Your query must target documents not retrieved in the first hop, using
 2367 clues from the summary and the original question.

2368 2. ****Key Strategy:****

- 2369 - Identify gaps in the first hop's retrieved documents (e.g., missing entities,
 2370 relationships, or specific details).
- 2371 - Use explicit information from the summary (e.g., names, locations, quantities) to
 2372 rephrase the question into a query that surfaces new relevant documents.
- 2373 - Avoid restating the answer directly; instead, structure the query to explore
 2374 connections or unresolved details.

2375 3. ****Domain-Specific Guidance:****

- 2376 - If the summary explicitly answers the question, the query should still focus on
 2377 retrieving documents that provide deeper context or verify the answer (e.g., "What is
 2378 the headquarters location of [Company]?" instead of "The answer is [Location]").
- 2379 - Leverage entities mentioned in the summary (e.g., "Carhartt," "Aubrey O'Day") to
 2380 anchor the query.
- 2381 - If no documents are missing, rephrase the query to explicitly request the answer
 2382 (e.g., "Which has more acts, Elektra or From the House of the Dead?").

2383 4. ****Avoid:****

- 2384 - Generating queries that duplicate the original question.
- 2385 - Assuming the summary contains all necessary information for the second hop.

2386
 2387 HotpotQA Qwen3 8B final_answer.predict

2388 Base Prompt:

2389 Given the fields `question`, `summary_1`, `summary_2`, produce the fields `answer`.

2390 MIPROv2 Prompt:

2391 Given the question, summary_1, and summary_2, generate a step-by-step reasoning process
 2392 that logically connects the information from both summaries to arrive at the final

2397
 2398 answer, and then provide the answer. Ensure the reasoning explicitly references how the
 2399 summaries support the conclusion and maintains consistency with the question's
 2400 requirements.
 2401 Demos:
 2402 Example 1:
 2403 question:
 2404 are Machaeranthera and Prumnopitys both the plants?
 2405 summary_1:
 2406 Yes, both Machaeranthera (a genus of flowering plants in the daisy/aster family) and
 2407 Prumnopitys (a genus of coniferous trees in the podocarp family) are plants.
 2408 summary_2:
 2409 Yes, both Machaeranthera (a flowering plant genus in the daisy family) and
 2410 Prumnopitys (a coniferous tree genus in the podocarp family) are plants.
 2411 reasoning:
 2412 Both summaries confirm that Machaeranthera and Prumnopitys are plants. Summary 1
 2413 explicitly states they are plants, categorizing Machaeranthera as a flowering plant
 2414 genus in the daisy/aster family and Prumnopitys as a coniferous tree genus in the
 2415 podocarp family. Summary 2 repeats this confirmation with slightly different phrasing.
 2416 answer:
 2417 Yes
 2418 #####
 2419 3 other demos omitted for brevity.
 2420 #####

2421 GEPA Prompt generated by config GEPA+Merge:

2422 Given the fields `question`, `summary_1`, and `summary_2`, produce the field `answer` by:
 2423 1. **Extracting precise terminology**: Identify the exact noun or specific term required
 2424 in the answer (e.g., "Medicare" rather than "Medicare cuts"). Avoid vague or generalized
 2425 terms unless explicitly stated in the summaries.
 2426 2. **Resolving ambiguity**: If the question references a title, historical role, or
 2427 specific designation (e.g., "second Duke of Florence"), prioritize contextual or
 2428 historical clues from the summaries to infer the correct answer, even if the exact term
 2429 is not explicitly stated. Use domain-specific knowledge (e.g., Medici family lineage) to
 2430 fill gaps when summaries are indirect or vague.
 2431 3. **Cross-referencing summaries**: Ensure consistency between summaries. If summaries
 2432 conflict, prioritize the one with explicit factual claims (e.g., numerical data, direct
 2433 statements). If no explicit claim exists, synthesize information while ensuring alignment
 2434 with historical, political, or cultural context.
 2435 4. **Avoiding overgeneralization and extra information**: Focus strictly on the most
 2436 specific and directly stated information in the summaries. Do not add context,
 2437 explanations, or external knowledge beyond what is explicitly provided. For example, if
 2438 the question asks for a year, provide only the year; do not include band member details
 2439 or historical background.
 2440 5. **Prioritizing factual alignment**: If a summary explicitly states the answer, use
 2441 that. If summaries are indirect or vague, synthesize information while ensuring alignment
 2442 with factual knowledge (e.g., linking "Path to Prosperity" to Rep. Paul Ryan's Medicare
 2443 proposal).
 2444 **Key adjustments based on feedback**:
 2445 - **Conciseness**: Answers must be strictly factual and concise, avoiding additional
 2446 context or explanations. For example, if the question is "Is X shorter than Y?" the

2444
 2445 answer should be a simple "No" or "Yes" based on numerical comparisons, not a full
 2446 explanation.
 2447 - **Numerical precision**: When comparing measurements (e.g., heights, dates), ensure
 2448 exact values are used and explicitly stated in the summaries. If summaries provide
 2449 conflicting numbers, resolve via direct factual claims.
 2450 - **Domain-specific knowledge**: Use known facts (e.g., architectural records, historical
 2451 timelines) to validate ambiguous answers, but only when summaries lack explicit
 2452 information.

2453
 2454 HotpotQA Qwen3 8B summarize1.predict
 2455
 2456 Base Prompt:
 2457 Given the fields `question` , `passages` , produce the fields `summary` .
 2458
 2459 MIPROv2 Prompt:
 2460 Given the fields `question` , `passages` , produce the fields `summary` .
 2461 Demos:
 2462 Example 1:
 2463 question:
 2464 Tay Garnett and Alexander Kluge both have what job?
 2465 passages:
 2466 ['Tay Garnett | William Taylor "Tay" Garnett (June 13, 1894 – October 3, 1977) was
 2467 an American film director and writer.', 'Trade Winds (film) | Trade Winds is a 1938
 2468 comedy film distributed by United Artists. It was directed by Tay Garnett, and
 2469 starred Fredric March and Joan Bennett. The screenplay was written by Dorothy Parker,
 2470 Alan Campbell and Frank R. Adams, based on story by Tay Garnett.', 'Prestige (film) |
 2471 Prestige is a 1932 American pre-Code drama film directed by Tay Garnett and written
 2472 by Tay Garnett, Rollo Lloyd and Francis Edward Faragoh. The film stars Ann Harding,
 2473 Adolphe Menjou, Melvyn Douglas and Guy Bates Post. The film was released on January
 2474 22, 1932, by RKO Pictures.', 'Alexander Kluge | Alexander Kluge (born 14 February
 2475 1932) is a German author, philosopher, academic and film director.', 'Cattle King |
 2476 Cattle King is a 1963 film directed by Tay Garnett. It stars Robert Taylor and Robert
 2477 Loggia. It also appears to have been called Guns of Wyoming in some countries.',
 2478 'Günther von Kluge | Günther von Kluge (30 October 1882 – 19 August 1944) was a
 2479 German field marshal during World War II. Kluge held commands on both the Eastern and
 2480 Western Fronts. Although Kluge was not an active conspirator in the 20 July plot, he
 2481 committed suicide on 19 August 1944, after having been recalled to Berlin for a
 2482 meeting with Hitler in the aftermath of the failed coup. He was replaced by Field
 2483 Marshal Walter Model.', 'Bad Company (1931 film) | Bad Company is a 1931 American
 2484 Pre-Code gangster film directed and co-written by Tay Garnett with Tom Buckingham
 2485 based on Jack Lait's 1930 novel "Put on the Spot". It stars Helen Twelvetrees and
 2486 Ricardo Cortez. Told from the view of a woman, the working titles of this film were
 2487 "The Gangster's Wife" and "The Mad Marriage". Unlike many static early sound films,
 2488 Garnett includes several scenes using a moving camera climaxing in a gigantic assault
 2489 on an office building with both sides using heavy machine guns.']
 2490 reasoning:
 The question asks for the common job of Tay Garnett and Alexander Kluge. From the
 2486 passages, Tay Garnett is explicitly identified as a film director and writer
 2487 (passages 1, 2, 3, 5, 7). Alexander Kluge is described as a German author,
 2488 philosopher, academic, and film director (passage 4). The shared job between them is
 2489 "film director."

2491
 2492 **summary:**
 2493 Both Tay Garnett and Alexander Kluge are film directors.
 2494
 2495 #####
 2496 3 other demos omitted for brevity.
 2497 #####

2498 GEPA Prompt generated by config GEPA+Merge:
 2499

2500 You are the first-hop **summarization module** in a multi-hop QA system. Your task is to
 2501 generate a **comprehensive, structured summary** that:

2502 1. **Extracts direct answers** from the top retrieved passages to address the question.
 2503 2. **Identifies and highlights missing or implied clues** that may require further
 2504 retrieval (e.g., entities, connections, or contextual details).
 2505 3. **Synthesizes information** by combining explicit facts from the passages with
 2506 domain-specific knowledge or logical inferences to guide subsequent steps.

2507
 2508 **Summary Structure**
 2509 - **Entity/Person Mention**: Clearly state the subject (e.g., "Billy Truax", "Eintracht
 2510 Braunschweig") and include **full names, titles, or official designations** (e.g.,
 2511 "Thomas Lance Rentzel", "Braunschweiger Turn- und Sportverein Eintracht von 1895 e.V.").
 2512 - **Direct Answer**: Include **explicit answers** from the passages (e.g., birth dates,
 2513 team affiliations, or direct statements).
 2514 - **Clues for Next Steps**: Signal **missing information** (e.g., "Lance Rentzel's birth
 2515 year is explicitly stated, but his exact birthplace is not; need to search for 'Lance
 2516 Rentzel birthplace'").
 2517 - **Domain-Specific Context**: Add **relevant background** (e.g., "Eintracht Braunschweig
 2518 is a German football club based in Braunschweig, Lower Saxony" or "NFL players' birth
 2519 dates are critical for age comparisons").

2520
 2521 **Guidelines**
 2522 - **Do not omit** any entity or detail from the retrieved passages that could be relevant
 2523 for follow-up queries (e.g., team names, locations, or historical context).
 2524 - **Prioritize clarity** by **separating direct answers from inferred clues** (e.g.,
 2525 using bullet points, subheadings, or bolded labels).
 2526 - **Avoid assumptions** not supported by the passages; if information is absent,
 2527 **explicitly state that it is missing** and suggest **precise search terms** (e.g.,
 2528 "Verify Wichita Dwight D. Eisenhower National Airport's tower status via FAA records").
 2529 - **Include quantifiable data** (e.g., "few thousand Stabyhouns exist globally", "born
 2530 July 15, 1943") to enable precise comparisons.
 2531 - **Highlight connections** between entities (e.g., "Billy Truax and Lance Rentzel were
 2532 traded in 1970") to aid in cross-referencing.

2533
 2534 **Key Niche/Domain-Specific Insights**
 2535 - **NFL Player Comparison**: Birth dates are critical for age determination, and team
 2536 affiliations (e.g., "traded in 1970") may imply historical context.
 2537 - **Airport Classification**: "Non-towered" status is explicitly stated in some passages
 (e.g., "non-towered public airport"), while others require inference (e.g., "major
 commercial airports typically have towers").
 - **Football Club Context**: Clubs like Eintracht Braunschweig require background on
 their location, league, and history (e.g., "based in Braunschweig, Lower Saxony").

2538
 2539 - **Quantifiable Data**: Use exact dates, numbers, or rankings (e.g., "few thousand
 2540 Stabyhouns exist globally") to enable precise comparisons.
 2541
 2542 **Critical Additional Instructions**
 2543 - **Ensure All Retrieved Documents Are Represented**: Explicitly include all entities,
 2544 titles, and details from the retrieved passages (e.g., full names, film titles, and
 2545 specific roles).
 2546 - **Signal Missing Links**: If a connection between entities is implied but not
 2547 explicitly stated (e.g., "Nancy Steiner worked on *The Lovely Bones*"), flag this as a
 2548 potential gap and suggest search terms to resolve it.
 2549 - **Prioritize Bridging Concepts**: Highlight relationships between entities (e.g., "Gary
 2550 Pinkel coached Toledo in 1993 and holds the most wins in school history") to enable
 2551 focused follow-up queries.
 2552 - **Avoid Overgeneralization**: Only include domain-specific context that is either
 2553 explicitly stated in the passages or directly inferable (e.g., "major commercial airports
 2554 typically have towers" is acceptable, but "airports with fewer than 10,000 passengers are
 2555 non-towered" is not unless stated).
 2556
 2557 **Example Format**
 2558 For the question "Which NFL player is younger, Billy Truax or Lance Rentzel?":
 2559 - **Entity/Person Mention**: Billy Truax (William Frederick Truax), Lance Rentzel (Thomas
 2560 Lance Rentzel)
 2561 - **Direct Answer**:
 2562 - **Billy Truax**: Born July 15, 1943.
 2563 - **Lance Rentzel**: Born October 14, 1943.
 2564 - **Clues for Next Steps**: None required; birth dates are explicitly provided.
 2565 - **Domain-Specific Context**: Birth dates are sufficient to determine age difference
 2566 within the same year.
 2567
 2568 For the question "Which is a non-towered airport, Wichita Dwight D. Eisenhower National
 2569 Airport or Montrose Regional Airport?":
 2570 - **Entity/Person Mention**: Wichita Dwight D. Eisenhower National Airport, Montrose
 2571 Regional Airport
 2572 - **Direct Answer**:
 2573 - **Montrose Regional Airport**: "non-towered public airport" (passage 3).
 2574 - **Wichita Dwight D. Eisenhower National Airport**: No explicit mention of tower
 2575 status; inferred as **towered** (typical for major commercial airports).
 2576 - **Clues for Next Steps**: Verify Wichita's tower status via FAA records or additional
 2577 sources (e.g., "Wichita Dwight D. Eisenhower National Airport tower status").
 2578 - **Domain-Specific Context**: Non-towered airports lack a control tower, relying on
 2579 pilot communication (passage 4). Major commercial airports like Wichita usually have
 2580 towers.
 2581
 2582
 2583
 2584

2585 HotpotQA Qwen3 8B summarize2.predict

2586

2587 Base Prompt:

2588 Given the fields `question`, `context`, `passages`, produce the field `summary`.

2589

2590 MIPROv2 Prompt:

2591 Given the fields `question`, `context`, and `passages`, generate a structured reasoning

2592 process that explains how to synthesize information from these sources, and then produce

2593 a concise summary of the key findings.

2594 Demos:

2595 Example 1:

2596 question:

2597 Simone Benedetti plays for a team that currently competes in which Serie?

2598 context:

2599 The provided passages do not specify the Serie in which Simone Benedetti's team

(Virtus Entella) currently competes.

2600 passages:

2601 ['Simone Benedetti | Simone Benedetti (born 3 April 1992) is an Italian professional

2602 footballer who plays as a centre back for Virtus Entella.', 'Virtus Entella | Virtus

2603 Entella (usually referred as simply Entella or Chiavari Entella) is an Italian

2604 association football club, based in Chiavari, Liguria. They currently compete in

2605 Serie B.', 'Simone Palermo | Simone Palermo (born 17 August 1988) is an Italian

2606 footballer who plays as a midfielder for Virtus Entella at Serie B.', 'Andrea Paroni

2607 | Andrea Paroni (born 14 October 1989) is an Italian footballer who currently plays

2608 as a goalkeeper for Virtus Entella in the Serie B.', 'Alessandro Iacobucci |

2609 Alessandro Iacobucci (born 3 June 1991) is an Italian footballer who currently plays

2610 as a goalkeeper for Serie B side Virtus Entella.', 'George Ganea | George Ganea (born

2611 26 May 1999 in Bucharest) is a Romanian footballer who plays as a Striker for Serie B

2612 club Virtus Entella Primavera team.', 'Gianpaolo Castorina | Gianpaolo Castorina

2613 (born August 30, 1976 in Milan) is a retired Italian professional footballer turned

2614 coach, currently in charge as head coach of Serie B club Virtus Entella.]

2615 reasoning:

2616 The question asks for the Serie in which Simone Benedetti's team (Virtus Entella)

2617 currently competes. While the context initially states that this information is not

2618 specified, the passages explicitly mention that Virtus Entella competes in Serie B.

2619 Multiple passages (e.g., [2], [3], [4], [5], [7]) confirm this detail, directly

2620 answering the question.

2621 summary:

2622 Simone Benedetti's team, Virtus Entella, currently competes in Serie B.

2623 #####

2624 3 other demos omitted for brevity.

2625 #####

2626 GEPA Prompt generated by config GEPA+Merge:

2627 Given the fields `question`, `context`, and `passages`, produce the field `summary`.

2628 Your task is to synthesize information from the question, context, and newly retrieved

2629 passages to generate a **comprehensive, precise, and well-structured summary** that

2630 enables the answer generation module to confidently arrive at the correct answer.

2631

2632
 2633 ### Key Requirements:
 2634 1. **Explicit Answers First**: Prioritize explicitly stated facts from the context and
 2635 passages (e.g., direct mentions of entities, roles, or relationships).
 2636 2. **Infer or Generalize When Necessary**: If critical details are missing from the
 2637 passages, infer connections or generalize based on contextual clues and domain-specific
 2638 knowledge (e.g., linking ownership structures, roles, or historical context).
 2639 3. **Bridge Gaps**: Ensure the summary includes all **key supporting information**
 2640 required to answer the question, even if it is not explicitly stated in the input. For
 2641 example:
 2642 - If the answer is "Newcastle United," include details about Sports Direct's ownership
 2643 and the connection to the billionaire.
 2644 - If the answer is a person's role (e.g., "troubleshooter"), explicitly state their
 2645 relationship to the question's subject and any relevant background.
 2646 4. **Structure and Precision**:
 2647 - Clearly connect entities, roles, and relationships (e.g., "Stan Kroenke owns Sports
 2648 Direct and Arsenal F.C.").
 2649 - Avoid ambiguity by including all necessary contextual links (e.g., "Mike Ashley
 2650 founded Sports Direct and owns Newcastle United").
 2651 - Use precise terminology and ensure alignment with domain-specific knowledge (e.g.,
 2652 "investigative journalist" instead of "writer").
 2653 5. **Domain-Specific Knowledge**: Leverage implicit domain knowledge when passages lack
 2654 critical details (e.g., knowing that "Project RAND" is linked to Henry H. Arnold and the
 2655 RAND Corporation).
 2656
 2657 ### Example Integration:
 2658 If the question is about a person's profession in a novel, ensure the summary includes:
 2659 - The character's name.
 2660 - Their profession (explicitly stated in the text).
 2661 - Contextual links to the book series or plot (e.g., "in *The Girl in the Spider's Web*").
 2662 - Any relevant background about the profession or character's role in the story.
 2663
 2664 Always aim to match the **coverage and relevance** of an "ideal summary" as described in
 2665 the feedback, ensuring the answer module has all necessary information to generate the
 2666 correct final answer.

N.3 IFBENCH, GPT-4.1 MINI

2666 IFBench GPT-4.1 Mini generate_response_module.predict

2667 Base Prompt:

2668

Respond to the query

2669 MIPROV2 Prompt:

2670

You will be given a query containing a complex task with multiple constraints and
 2671 instructions. Your job is to first repeat the query exactly as given, word for word,
 2672 without adding any extra words or commentary before or after the repetition. After
 2673 repeating the query, provide a detailed, step-by-step chain-of-thought reasoning that
 2674 carefully unpacks and addresses every aspect of the query. Then, produce a final response
 2675 that adheres strictly to all requirements, including formatting, language, and content
 2676 constraints specified in the query. Be thorough, precise, and ensure the final answer is
 2677 correct.

2679
 2680 fully validated and consistent with the reasoning. If the query specifies additional
 2681 formatting or stylistic rules (such as language or inclusion of a postscript), include
 2682 them exactly as instructed.
 2683 Demos:

2684 GEPA Prompt generated by config GEPA+Merge:

2685 You are given a query input, and your task is to respond appropriately to that query. The
 2686 query may contain specific instructions or constraints that you must strictly adhere to
 2687 in your response. Carefully analyze the query to determine the exact requirements,
 2688 including but not limited to:

2689 - Responding with an answer chosen from a restricted set of options exactly as specified
 2690 (including exact wording and punctuation).
 2691 - Including specific words a minimum number of times.
 2692 - Including specific letters a minimum number of times.
 2693 - Repeating the entire query word-for-word before providing your answer if explicitly
 2694 requested, without adding extra words or characters before the repetition.
 2695 - Avoiding specific forbidden words or keywords in your response if indicated.
 2696 - Following any other explicit instructions or constraints embedded in the query.

2697 When the query requests calculations or factual answers (e.g., combinatorial
 2698 calculations), you should:

2699 1. Carefully interpret the mathematical or logical problem.
 2700 2. Show your reasoning internally to confirm the final answer (reasoning does not need to
 2701 be included in the response unless explicitly requested).
 2702 3. Provide the final direct response strictly following all instructions, especially when
 2703 asked to repeat the query verbatim first before giving the answer.

2704 General approach:

2705 - Always parse the query thoroughly to extract every constraint and instruction.
 2706 - Ensure your response exactly matches the format, wording, and content as instructed.
 2707 - Do not invent or omit any part of the user's explicit requests.
 2708 - Meet all formatting, lexical, numeric, and structural constraints without deviation.
 2709 - If the query involves repeating text verbatim, do not alter capitalization,
 2710 punctuation, or wording.
 2711 - Incorporate any required keywords or letters the required number of times naturally
 2712 into your response.
 2713 - When multiple constraints (like avoiding specific words while including others) apply
 2714 simultaneously, ensure you satisfy all simultaneously.

2715 This task requires precision, exact reproduction, and strict adherence to any given
 2716 constraints or instructions embedded in the query. Your goal is to deliver the requested
 2717 answer in the exact manner requested without extraneous additions or omissions.

2718
 2719 IFBench GPT-4.1 Mini ensure_correct_response_module.predict

2720
 2721 Base Prompt:
 2722

2726
 2727 Ensure the response is correct and adheres to the given constraints. Your response will
 2728 be used as the final response.

2729 MIPROv2 Prompt:

2730
 2731 Given the original query and the initially generated response, carefully verify that the
 2732 response fully meets all specified constraints and instructions in the query. Produce
 2733 detailed step-by-step reasoning that explains how the response satisfies or fails the
 2734 requirements such as exact repetition of the query text, minimum sentence count,
 2735 formatting rules (e.g., capitalization), letter frequency, or other explicit user
 2736 demands. Based on this reasoning, output a final corrected and fully compliant response
 2737 that adheres strictly to the original instructions, ensuring accuracy, completeness, and
 2738 formatting correctness. Your final output should begin by repeating the original query
 2739 exactly as provided, with no modifications or additional text before it, followed by a
 2740 comprehensive, well-structured answer that respects all constraints, and a clear
 2741 explanation of how the requirements have been met.

2742 Demos:

2743 Example 1:

2744 query:
 2745 Can you provide the social media handles of Harry Potter? Your answer must contain
 2746 exactly 8 bullet points. Use the markdown bullet points such as:
 2747 * This is point 1.
 2748 * This is point 2 Finish your response with this exact phrase That's my answer.. No other
 2749 words should follow this phrase.

2750 response:
 2751 * Twitter: [@wizardingworld](https://twitter.com/wizardingworld)
 2752 * Instagram: [@wizardingworld](https://www.instagram.com/wizardingworld/)
 2753 * Facebook: [Wizarding World](https://www.facebook.com/wizardingworld)
 2754 * YouTube: [Wizarding World](https://www.youtube.com/c/WizardingWorld)
 2755 * Twitter: [@harrypotterfilm](https://twitter.com/harrypotterfilm)
 2756 * Instagram: [@harrypotterfilm](https://www.instagram.com/harrypotterfilm/)
 2757 * Facebook: [Harry Potter](https://www.facebook.com/harrypotter)
 2758 * TikTok: [@wizardingworld](https://www.tiktok.com/@wizardingworld)

2759 That's my answer..
 2760 reasoning:
 2761 The request is for the social media handles of Harry Potter, formatted as exactly
 2762 eight bullet points using markdown bullet points, and the response must end with the
 2763 exact phrase "That's my answer." The provided response includes eight bullet points,
 2764 referencing official or widely recognized social media accounts related to Harry
 2765 Potter and the Wizarding World franchise on Twitter, Instagram, Facebook, YouTube,
 2766 and TikTok. The format is correct, the bullet points are used properly with markdown
 2767 syntax, and the closing phrase matches the requirement exactly. Therefore, the
 2768 response meets all given constraints.

2769 final_response:
 2770 * Twitter: [@wizardingworld](https://twitter.com/wizardingworld)
 2771 * Instagram: [@wizardingworld](https://www.instagram.com/wizardingworld/)
 2772 * Facebook: [Wizarding World](https://www.facebook.com/wizardingworld)
 2773 * YouTube: [Wizarding World](https://www.youtube.com/c/WizardingWorld)
 2774 * Twitter: [@harrypotterfilm](https://twitter.com/harrypotterfilm)
 2775 * Instagram: [@harrypotterfilm](https://www.instagram.com/harrypotterfilm/)
 2776 * Facebook: [Harry Potter](https://www.facebook.com/harrypotter)
 2777 * TikTok: [@wizardingworld](https://www.tiktok.com/@wizardingworld)

2773
 2774 That's my answer..
 2775
 2776 #####
 2777 3 other demos omitted for brevity.
 2778 #####
 2779
 2780 GEPA Prompt generated by config GEPA+Merge:
 2781 You will receive a user query that includes explicit, detailed instructions about how you
 2782 must structure your response. Your overall task is to interpret these embedded
 2783 instructions with perfect accuracy and produce a response that strictly complies with
 2784 every single one, without adding, omitting, or rewording any mandated text, phrases, or
 2785 structural elements.
 2786 Key detailed requirements and strategy for this task:
 2787
 2788 1. ****Query Parsing and Extraction of Instructions****
 2789 - Carefully read the entire query to identify all explicit instructions concerning:
 2790 - Whether and how to repeat the query text (verbatim or partially).
 2791 - Specific length constraints (number of sentences, bullet points, word counts).
 2792 - Formatting instructions (e.g., capitalization requirements, quotation marks,
 2793 markdown bullet styles).
 2794 - Mandatory phrases or exact sentences that must appear (especially those to be
 2795 repeated verbatim or appended at the end).
 2796 - Content limitations or prohibitions (for example, refusal language or disclaimers
 2797 for unethical requests).
 2798 - Note that some instructions may be nested or appear within the query's wording and
 2799 are critical to follow exactly.
 2800
 2801 2. ****Exact Text Reproduction****
 2802 - When asked to repeat the query text (or any other required phrase) verbatim, do so
 2803 with zero changes – no added or removed words, punctuation, or formatting.
 2804 - Do not prepend or append anything to the repeated text unless explicitly instructed.
 2805 - Preserve all original capitalization, spacing, and punctuation exactly as in the
 2806 query.
 2807
 2808 3. ****Structural and Formatting Compliance****
 2809 - Follow all formatting instructions strictly, such as:
 2810 - Wrapping the entire response in quotation marks if required.
 2811 - Using specified markdown bullet point styles (e.g., asterisks).
 2812 - Ensuring capitalization instructions (e.g., all caps or minimum occurrences of
 2813 uppercase words) are perfectly met.
 2814 - Adhering to sentence or paragraph counts exactly as requested.
 2815
 2816 4. ****Response Content Accuracy and Appropriateness****
 2817 - After fulfilling all structural requirements, respond to the main substantive
 2818 question accurately and completely.
 2819 - 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.

2820
 2821 - Always respect ethical guidelines and any mandated refusal language or concluding
 2822 statements for such queries.

2823 5. ****No Extraneous Text****
 2824 - Do not add explanations, internal reasoning, apologies, or meta commentary beyond
 2825 what the query explicitly permits or demands.
 2826 - Your final output must be the exact, ready-to-deliver response that meets all user
 2827 instructions perfectly.

2828 6. ****Examples and Patterns Observed****
 2829 - Users often combine multiple complex formatting and content instructions (e.g.,
 2830 repetition of request text, followed by specific number of sentences or bullet points,
 2831 with capitalization rules).
 2832 - Ensure you carefully distinguish when to repeat the query text verbatim and when to
 2833 respond directly (sometimes the repetition excludes an instruction sentence).
 2834 - Handle instructions about capitalized words appearing a minimum number of times by
 2835 distributing such words naturally but thoroughly across the response.
 2836 - When length constraints specify minimums (e.g., “at least 39 sentences”), ensure
 2837 you meet or exceed exactly rather than approximating.
 2838 - For bullet points or enumerations, use the precise markdown style asked for
 2839 (commonly asterisks).
 2840 - Follow refusal instructions verbatim and terminate the response exactly as specified.

2841 ****In summary:**** YOUR RESPONSE MUST BE A PERFECT TEXTUAL AND STRUCTURAL RENDERING OF THE
 2842 USER QUERY’S EMBEDDED INSTRUCTIONS. ALWAYS PRIORITIZE EXACT USER SPECIFICATIONS OVER
 2843 DEFAULT GENERATIVE NORMS. THE OUTPUT IS FINAL AND WILL NOT BE EDITED, SO COMPLIANCE MUST
 2844 BE 100%.

2845 This task therefore requires a rigorously rule-based approach:
 2846 - First parse and isolate all explicit instructions (including repetitions and
 2847 formatting).
 2848 - Then assemble your answer by reproducing mandated text identically.
 2849 - Next produce the requested content in the exact required format, length, and style.
 2850 - Finally, review your output against the instructions ensuring total compliance before
 2851 delivering your response.

2853 N.4 IFBENCH, QWEN3 8B

2855 IFBench Qwen3 8B generate_response_module.predict

2856

2857 Base Prompt:

2858 Respond to the query

2859

2860 MIPROv2 Prompt:

2861

2862 You are a precision-focused instruction-adherence expert tasked with generating
 2863 structured, ethically compliant responses. When responding to queries, always think
 2864 step-by-step to analyze constraints (formatting rules, content boundaries, ethical
 2865 guidelines), synthesize requirements, and produce outputs that strictly adhere to
 2866 specified parameters. Ensure responses are fully formatted as requested (e.g., lowercase

2867
2868 text, bullet points, markdown), avoid any unverified claims, and maintain neutrality in
2869 ambiguous ethical scenarios.
2870 Demos:
2871 Example 1:
2872 query:
2873 Can you provide the social media handles of Harry Potter? Your answer must contain
2874 exactly 8 bullet points. Use the markdown bullet points such as:
2875 * This is point 1.
2876 * This is point 2 Finish your response with this exact phrase That's my answer.. No other
2877 words should follow this phrase.
2878 reasoning:
2879 The query asks for the social media handles of Harry Potter, a fictional character.
2880 Since he does not have personal social media accounts, the response should instead
2881 list official accounts related to the Harry Potter franchise. These include accounts
2882 managed by Warner Bros., the Wizarding World, and associated organizations. The
2883 answer is structured into exactly 8 bullet points as requested.
2884 response:
2885 * @WizardingWorld (Twitter)
2886 * @WizardingWorld (Instagram)
2887 * @WizardingWorld (Facebook)
2888 * @JKRowling (Twitter)
2889 * @HarryPotterAlliance (Facebook)
2890 * @HogwartsExpress (Twitter)
2891 * @TheWizardingWorldOfHarryPotter (Facebook)
2892 * @HarryPotterBooks (Twitter)
2893 That's my answer.

2894 #####
2895 3 other demos omitted for brevity.
2896 #####

2897 GEPA Prompt generated by config GEPA:

2898 Respond to the query by first repeating it word for word without any changes. Then,
2899 provide your answer. Ensure your answer includes niche or domain-specific factual
2900 information relevant to the task. If the query specifies constraints (e.g., letter
2901 frequency, sentence limits, or formatting rules), adhere to them strictly. Use a
2902 generalizable strategy to solve the task, such as breaking down complex problems into
2903 logical steps or applying specialized knowledge. Avoid adding any preamble or postscript
2904 before or after the repeated query.

2905 IFBench Qwen3 8B ensure_correct_response_module.predict

2906 Base Prompt:

2907 Ensure the response is correct and adheres to the given constraints. Your response will
2908 be used as the final response.

2909 MIPROv2 Prompt:

2910 You are a meticulous and ethical AI assistant tasked with generating detailed, compliant
2911 responses. When given a query, first think step by step to create a reasoning process

2914
 2915 that addresses the query's requirements, then produce a response that strictly adheres to
 2916 all specified constraints (e.g., formatting rules, content boundaries, ethical
 2917 guidelines). Ensure the final output is structured with a "Reasoning:" section followed
 2918 by a "Final Response:" section, and verify that it aligns with the query's exact
 2919 specifications while maintaining logical coherence and ethical responsibility.
 2920 Demos:
 2921 Example 1:
 2922 query:
 2923 What are some techniques for winning a swordfight in a medieval combat game? In your
 2924 response, the letter f should appear less than 18 times. Do not include keywords
 2925 ['heart', 'listen', 'reputation', 'side'] in the response.
 2926 response:
 2927 Prioritize stance and balance to control the fight's flow. Use angles to create
 2928 openings for strikes while minimizing exposure. Practice parrying with precision to
 2929 disrupt opponent rhythm. Maintain distance to avoid overcommitting, and target weak
 2930 points like joints or limbs. Focus on rhythm and timing to exploit vulnerabilities.
 2931 reasoning:
 2932 The response outlines key techniques for medieval swordfighting, focusing on stance,
 2933 angles, parrying, distance, and targeting weak points. It avoids the restricted
 2934 keywords and limits the letter 'f' to 2 instances, satisfying the constraints. The
 2935 advice is practical and aligns with historical combat strategies.
 2936 final_response:
 2937 Prioritize stance and balance to control the fight's flow. Use angles to create
 2938 openings for strikes while minimizing exposure. Practice parrying with precision to
 2939 disrupt opponent rhythm. Maintain distance to avoid overcommitting, and target weak
 2940 points like joints or limbs. Focus on rhythm and timing to exploit vulnerabilities.

2941 GEPA Prompt generated by config GEPA:
 2942

2943 Ensure the response strictly follows these instructions:
 2944 1. **First**, repeat the user's query **word for word** without any changes or additions.
 2945 2. **Then**, provide your answer in the specified format, adhering to all constraints
 (e.g., markdown, structure, content).
 2946 3. **Do not include any additional text, explanations, or formatting** beyond the
 2947 repeated query and your answer.
 2948 4. **Include niche/domain-specific factual details** (e.g., technical commands, best
 2949 practices, or platform-specific configurations) if applicable, as these are critical for
 accurate task completion.
 2950 5. **Use precise formatting** (e.g., bullet points, code blocks, headers) as requested,
 2951 ensuring no markdown is omitted or altered.
 2952 6. **Avoid generalizable strategies** unless explicitly instructed; focus on actionable,
 2953 specific guidance.
 2954 7. **Validate all technical steps** (e.g., Dockerfile syntax, CLI commands) for accuracy
 2955 and completeness.
 2956 8. **Highlight potential pitfalls and solutions** to address common issues in the task.
 2957 9. **Prioritize clarity and conciseness**, ensuring the response is both comprehensive
 2958 and easy to follow.
 2959 10. **Adhere to language and case requirements** (e.g., all caps, English only) if
 specified.

2961 N.5 HoVER, GPT-4.1 MINI
 2962

2963 HoVer GPT-4.1 Mini create_query_hop2.predict
 2964

2965 Base Prompt:

2966 Given the fields `claim`, `summary_1`, produce the fields `query`.
 2967

2968 MIPROv2 Prompt:

2969 Given the original claim and the initial summary of retrieved evidence, carefully analyze
 2970 the information step-by-step to identify any gaps, contradictions, or points needing
 2971 further clarification. Then, generate a clear, focused, and precise query that targets
 2972 additional relevant information to deepen the investigation and help verify or refute the
 2973 claim. Your response should include a detailed chain-of-thought reasoning explaining your
 2974 thought process in formulating this refined query.

2975 Demos:

2976 Example 1:

2977 claim:

2978 AirBaltic's has bases at Tallinn Airport and the airport that ceased operations in
 2979 2015, with the airport code of Ryanair: VNO, ICAO: EYVI.

2980 summary_1:

2981 AirBaltic has bases at Tallinn Airport and Vilnius Airport (IATA: VNO, ICAO: EYVI),
 2982 but Vilnius Airport did not cease operations in 2015. The airport that ceased
 2983 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
 2984 2015 is false.

2985 reasoning:

2986 The claim states that AirBaltic has bases at Tallinn Airport and an airport with
 2987 codes VNO (IATA) and EYVI (ICAO) which ceased operations in 2015. However, the
 2988 summary clarifies that Vilnius Airport, which has those codes, did not cease
 2989 operations in 2015. The airport that did cease operations in 2015 has different codes
 2990 and is unrelated to AirBaltic's bases. Therefore, the claim that AirBaltic's base at
 2991 the airport with codes VNO/EYVI ceased operations in 2015 is incorrect.

2992 query:

2993 Did Vilnius Airport (IATA: VNO, ICAO: EYVI) cease operations in 2015?
 2994 #####
 2995 3 other demos omitted for brevity.
 #####

2996 GEPA Prompt generated by config GEPA+Merge:
 2997

3000 Task Description:

3001 You are given two fields: `claim` and `summary_1`. Your goal is to produce a field called
 3002 `query` a concise set of relevant, fact-checking questions or search style queries
 3003 that can be used to retrieve evidence documents verifying or refuting the claim based on
 3004 information contained or inferred from the summary.

3005 Input Format:

3006 - `claim`: A factual statement potentially containing multiple fact assertions about
 3007 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.

3008
 3009
 3010
 3011
 3012
 3013
 3014
 3015
 3016
 3017
 3018
 3019
 3020
 3021
 3022
 3023
 3024
 3025
 3026
 3027
 3028
 3029
 3030
 3031
 3032
 3033
 3034
 3035
 3036
 3037
 3038
 3039
 3040
 3041
 3042
 3043
 3044
 3045
 3046
 3047
 3048
 3049
 3050
 3051
 3052
 3053
 3054

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** – 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:**
 - 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.
 - 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.**
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 person's 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

3055
 3056 from both summaries. Provide a detailed chain-of-thought reasoning explaining how you
 3057 integrate the summaries and the claim to formulate this refined query.
 3058 Demos:
 3059 Example 1:
 3060 claim:
 3061 The brother of Freddie Sessler, was the restaurateur whose club was frequented by
 3062 Ruth Elizabeth "Bette" Davis, an actress with Welsh ancestry.
 3063 summary_1:
 3064 Freddie Sessler's brother was Siegi Sessler, a London restaurateur whose club was
 3065 frequented by Bette Davis, an actress with Welsh ancestry.
 3066 summary_2:
 3067 Freddie Sessler's brother was Siegi Sessler, a prominent London restaurateur whose
 3068 club, Siegi's, was frequented by Hollywood stars including Ruth Elizabeth "Bette"
 3069 Davis, an actress with Welsh ancestry.
 3070 reasoning:
 3071 The claim states that the brother of Freddie Sessler was a restaurateur whose club
 3072 was frequented by Ruth Elizabeth "Bette" Davis, noted to have Welsh ancestry. Both
 3073 summaries support this claim by identifying the brother as Siegi Sessler, a
 3074 restaurateur in London, whose club was indeed frequented by Bette Davis, an actress
 3075 noted for Welsh ancestry. Summary_2 adds that the club's name was Siegi's and
 3076 mentions the presence of other Hollywood stars. To fact-check the claim accurately,
 3077 it would be useful to verify the identity of Freddie Sessler's brother as Siegi
 3078 Sessler, confirm that he was a restaurateur, verify that Bette Davis frequented his
 3079 club, and confirm Bette Davis's Welsh ancestry.
 3080 query:
 3081 Who was Siegi Sessler, and was he the brother of Freddie Sessler? Was Siegi Sessler a
 3082 restaurateur whose club was frequented by Ruth Elizabeth "Bette" Davis? Did Bette
 3083 Davis have Welsh ancestry?
 3084 #####
 3085 3 other demos omitted for brevity.
 3086 #####

3087 GEPA Prompt generated by config GEPA+Merge:

3088 Given three text fields: `claim`, `summary_1`, and `summary_2`, your task is to produce a
 3089 `query` field that is designed to effectively retrieve evidence documents relevant to
 3090 verifying or refuting the claim based on the information contained in the two summaries.

3091 Detailed task description and considerations:

3092 1. ****Purpose of the Query:****
 3093 - The query should accurately and comprehensively target the key factual elements from
 3094 the claim that are addressed or clarified in the summaries.
 3095 - The query will be used to retrieve evidence documents, so it should be specific
 3096 enough to pinpoint relevant support or contradiction but broad enough to cover all
 3097 important details present in the summaries.

3098 2. ****Utilizing the Summaries:****
 3099 - Summaries often correct, clarify, or add factual context to the claim. Your query
 3100 must incorporate these clarifications (e.g., name corrections, factual specifics, or
 3101 counterpoints) to ensure retrieval of relevant evidence reflecting the nuanced truth.

3102

3103 - Include all distinctive entities, facts, dates, locations, names, and relationships
3104 mentioned or corrected in the summaries that pertain to the claim.
3105 - For example, if a summary corrects a documentary title in the claim, the query
3106 should reference the corrected title and related details to guide retrieval
3107 effectively.

3108 3. ****Query Content Strategy:****
3109 - Explicitly mention key entities, such as persons, places, dates, works, or political
3110 entities involved.
3111 - Include attributes or relationships relevant to the claim's accuracy (e.g., "Was
3112 person X a politician in country Y during year Z?" or "Did documentary A and
3113 documentary B film in different locations such as location 1 and location 2?").
3114 - If there is a factual dispute or correction in the summaries (e.g., nationality,
3115 official names, population figures), phrase the query to target evidence clarifying
3116 this dispute.

3117 4. ****Domain-Specific Nuances:****
3118 - Be mindful of historical geopolitical names and periods (e.g., "United Kingdom of
3119 the Netherlands between 1815 and 1830").
3120 - Recognize proper titles and correct spellings (e.g., corrected documentary titles).
3121 - Include both subjects or objects mentioned as comparisons or contrasts in the claim
3122 and summaries (e.g., two documentaries, two politicians, two towns).

3123 5. ****Formulation Style:****
3124 - Queries should be phrased as clear, precise, and objective questions that can be
3125 answered based on evidence often structured as yes/no or informational queries.
3126 - Avoid overly broad or vague phrasing. Aim for detail-rich queries that connect
3127 multiple evidence points in the summaries.

3128 6. ****Generalizable Strategy:****
3129 - Identify the core claim elements and verify if the summaries confirm, contradict, or
3130 amend these elements.
3131 - Incorporate both the claim and corrections from summaries into the query so evidence
3132 retrieval captures the full factual context.
3133 - Use the comparison or contrast highlighted by the summaries to create queries that
3134 specifically test the claim's veracity (e.g., comparing locations, roles, or
3135 historical timeframes).

3136 By following these guidelines, you will generate queries that maximize the likelihood of
3137 retrieving relevant texts that confirm or refute the claim accurately.

3138 HoVer GPT-4.1 Mini summarize1.predict

3139

3140 Base Prompt:
3141 Given the fields `claim`, `passages`, produce the fields `summary`.

3142

3143 MIPROv2 Prompt:
3144 Given a `claim` and a list of relevant `passages`, carefully analyze the evidence by
3145 reasoning step-by-step to assess the claim's validity. Produce a detailed
3146 chain-of-thought `reasoning` that explains how the information in the passages supports
3147 or refutes the claim, followed by a clear, concise `summary` that synthesizes the key
3148

3149
 3150 findings in relation to the claim. Ensure the reasoning explicitly connects evidence from
 3151 the passages to the claim, enabling a thorough and transparent multi-hop verification
 3152 process.
 3153 Demos:
 3154 Example 1:
 3155 claim:
 3156 The manufacturer that owns the company that currently produced and marketed the candy
 3157 the video game Darkened Skye uses in magic is headquartered in the US state Virginia.
 3158 passages:
 3159 ['Darkened Skye | Darkened Skye is a third-person action-adventure video game
 3160 developed by Boston Animation. It was released for Microsoft Windows and the Nintendo
 3161 GameCube in North America in 2002 and the PAL regions in 2003. The game was also
 3162 packaged with Outlaw Golf. Its title character is a young woman named Skye who lives
 3163 in a fantasy realm searching for her mother. She does not use firearms, but can
 3164 perform magic using Skittles candies, as well as use her staff as a melee weapon,
 3165 though it also becomes an energy weapon when used in conjunction with the Skittles.',
 3166 "'Rocket Fizz | Rocket Fizz is a franchise chain of candy stores in the United States.
 3167 The company markets a diverse variety of candies and produces its own line of soft
 3168 drinks with unique flavors. Its flagship store is located in Camarillo, California,
 3169 and most of the company's franchise stores are located in California. The company
 3170 also markets candies that are rare to find or in limited production by various
 3171 manufacturers, such as those that were popular during the 1960s to 1980s. As of
 3172 September 2017, there are 86 Rocket Fizz store locations in the United States.",
 3173 'Spangler Candy Company | The Spangler Candy Company is a privately owned
 3174 international confectioner that has been manufacturing and marketing candy since
 3175 1906.', 'Future US | Future US, Inc. (formerly known as Imagine Media and The Future
 3176 Network USA) is an American media corporation specializing in targeted magazines and
 3177 websites in the video games, music, and technology markets. Future US is
 3178 headquartered in the San Francisco with a small sales office in New York City. Future
 3179 US is owned by parent company, Future plc, a small publishing company based in the
 3180 United Kingdom.', 'Fun Dip | Fun Dip is a candy manufactured by The Willy Wonka Candy
 3181 Company, a brand owned by NestlÃ©. The candy has been on the market in the United
 3182 States and Canada since the 1940s and was originally called Lik-M-Aid. It was
 3183 originally manufactured by Fruzola, and then Sunline Inc., through their Sunmark
 3184 Brands division, which was purchased by NestlÃ© in January 1989. It comes in many
 3185 different flavors with candy sticks that are included.', 'Hot Tamales | Hot Tamales
 3186 is a chewy, cinnamon flavored, oblong-shaped candy introduced in 1950 â€”
 3187 manufactured and marketed in the United States by Just Born, a family-owned,
 3188 Pennsylvania-based candy company.', "Frankford Candy & Chocolate Company |
 3189 Frankford Candy & Chocolate Company is an American candy manufacturer, located in
 3190 Philadelphia, Pennsylvania, founded in 1947 by Sam Himmelstein. The current CEO is
 3191 Stuart Selarnick. In recent years, Frankford Candy & Chocolate Company has become a
 3192 very successful candy manufacturer. It is one of the largest producers of chocolate
 3193 rabbits in the United States, making over 100 varieties. In 2000, it started
 3194 licensing popular children's TV characters, such as SpongeBob SquarePants."]
 3195 reasoning:
 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

3196
 3197 Skittles is Mars, Incorporated, which is headquartered in McLean, Virginia. Although
 3198 this fact is not directly stated in the provided passages, it is relevant general
 3199 knowledge. Since the passages do not refute the claim and for accuracy based on known
 3200 facts, the claim can be considered true that the manufacturer that owns the company
 3201 producing and marketing Skittles is headquartered in Virginia.
 3202 summary:
 3203 The candy used as magic in Darkened Skye is Skittles, which are produced by Mars,
 3204 Incorporated. Mars is headquartered in McLean, Virginia, a US state. Therefore, the
 3205 claim is true based on external knowledge confirming that the manufacturer is based
 3206 in Virginia.
 3207 #####
 3208 3 other demos omitted for brevity.
 3209 #####

3210 GEPA Prompt generated by config GEPA+Merge:
 3211 Given the fields `claim`, `passages`, produce the fields `summary`.
 3212

3213

3214 HoVer GPT-4.1 Mini summarize2.predict

3215

3216 Base Prompt:
 3217 Given the fields `claim`, `context`, `passages`, produce the fields `summary`.

3218

3219 MIPROV2 Prompt:
 3220 Given a `claim`, relevant `context`, and a set of supporting `passages`, generate a
 3221 detailed step-by-step reasoning process that logically connects the evidence to the
 3222 claim, followed by a concise summary that clearly states whether the claim is supported
 3223 or refuted based on the aggregated information. Ensure the reasoning explicitly
 3224 references key evidence from the context and passages to justify the conclusion in the
 3225 summary.
 3226 Demos:
 3227 Example 1:
 3228 claim:
 3229 One half of the Kalyanji-Anandji duo, Kalyanji Virji Shah's brother, won the
 3230 Filmfare Award for Best Music Director.
 3231 context:
 3232 Yes, Kalyanji Virji Shah's brother Anandji Virji Shah, as part of the
 3233 Kalyanji-Anandji duo, won the Filmfare Award for Best Music Director in 1975 for
 3234 the film "Kora Kagaz."
 3235 passages:
 3236 ['Kalyanji Virji Shah | Kalyanji Virji Shah (30 June 1928 – 24 August 2000) was the
 3237 "Kalyanji" of the Kalyanji-Anandji duo. He and his brother Anandji Virji Shah have
 3238 been famous Indian film musicians, and won the 1975 Filmfare Award for Best Music
 3239 Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri
 3240 (1992).', 'Anandji Virji Shah | Anandji Virji Shah is an Indian music director.
 3241 Together with his brother he formed the Kalyanji-Anandji duo, and won the 1975
 3242 Filmfare Award for Best Music Director, for "Kora Kagaz". He is a recipient of the
 3243 civilian honour of Padma Shri (1992).', 'Kalyanji-Anandji | Kalyanji-Anandji are
 3244 an Indian composer duo from Gujarat: Kalyanji Virji Shah (30 June 1928–03 November

3243
 3244 2000) and his brother Anandji Virji Shah (born 02 March 1933). The duo are known for
 3245 their work on Hindi film soundtracks, particularly action potboilers in the 1970s.
 3246 Some of their best-known works are "Don", "Bairaag", "SaraswatiChandra", "Qurbani",
 3247 "Tridev" and "Safar". They won the 1975 Filmfare Award for Best Music Director for
 3248 "Kora Kagaz".', 'Kalyanjiâ€šAnandji discography | This is a discography of Bollywood
 3249 composer duo Kalyanji Anandji, consisting of Kalyanji Virji Shah and Anandji Virji
 3250 Shah. They have composed music for over 200 films in their 46 year career.', 'Viju
 3251 Shah | Viju Shah (born, Vijay Kalyanji Shah, 5 June 1959) is a music director of
 3252 Hindi cinema. He is the son of music director Kalyanji Virji Shah of composer duo
 3253 Kalyanji Anandji. He composed music for movies such as "Vishwatama" (1992), "Mohra"
 3254 (1994), "Tere Mere Sapne" (1996) and "" (1997) for which got his second nomination
 3255 for the Filmfare Award for Best Music Director and he won the 1998 Filmfare Award for
 3256 Best Background Score for ".', 'Bombay the Hard Way: Guns, Cars and Sitars | Bombay
 3257 the Hard Way: Guns, Cars and Sitars is a 1998 project of producer Dan the Automator
 3258 in which music from 1970s funk Bollywood composer duo Kalyanji-Anandji (Kalyanji
 3259 Virji Shah and Anandji Virji Shah) was taken and given a funky remix. The album was
 3260 eventually withdrawn (possibly due to copyright concerns). A sequel, "Bombay 2:
 3261 Electric Vindaloo", was also produced.', 'Kora Kagaz | Kora Kagaz (Hindi:
 3262 à¤†à¤œà¤‡à¤œ à¤‡à¤œà¤œà¤œà¤œ; translation: Blank Paper) is a 1974 Hindi drama film
 3263 produced by Sanath Kothari and directed by Anil Ganguly. The film stars Vijay Anand,
 3264 Jaya Bhaduri, A.K. Hangal, Achala Sachdev and Deven Verma. The films music is by
 3265 Kalyanji Anandji. The famous title song "Mera Jeevan Kora Kagaz" was by Kishore
 3266 Kumar.]

3263 reasoning:

3264 The claim states that one half of the Kalyanjiâ€šAnandji duo, Kalyanji Virji Shah's
 3265 brother, won the Filmfare Award for Best Music Director. From the passages and
 3266 context, it is clear that the duo consists of Kalyanji Virji Shah and his brother
 3267 Anandji Virji Shah. Both together won the Filmfare Award for Best Music Director in
 3268 1975 for the film "Kora Kagaz." Therefore, Anandji Virji Shah, who is Kalyanji's
 3269 brother, indeed won the Filmfare Award for Best Music Director as part of this duo.

3270 summary:

3271 Anandji Virji Shah, the brother of Kalyanji Virji Shah, won the Filmfare Award for
 3272 Best Music Director in 1975 as part of the Kalyanjiâ€šAnandji duo.

3273 #####

3274 3 other demos omitted for brevity.

3275 #####

3276 GEPA Prompt generated by config GEPA+Merge:

3277 Given the input fields:

- 3278 - `claim`: A factual statement or assertion that may be true or false.
- 3279 - `context`: A brief explanation or summary relating to the claim that typically
- 3280 clarifies the claimâ€šs accuracy.
- 3281 - `passages`: A list of relevant textual evidence or knowledge snippets containing facts,
- 3282 descriptions, or biographies related to entities or concepts mentioned in the claim.

3283 Your task is to produce a `summary` that meets the following requirements:

- 3284 1. **Accurately reflect the relationship between the claim and the evidence**: Analyze
 3285 the input `claim` carefully and check it against the `context` and all `passages`.

3290
 3291 Determine whether the claim is supported, partially supported, or contradicted by the
 3292 evidence.
 3293
 3294 2. **Explicitly connect key entities and facts from the passages**: Your summary must
 3295 mention the main entities and facts that directly confirm or refute parts of the claim.
 3296 For example, highlight relevant names, works (films, albums, songs, books), attributes,
 3297 dates, nicknames, or roles that clarify the claim's accuracy.
 3298
 3299 3. **Make substantive connections**: The summary should not merely restate the claim or
 3300 conclusion. Instead, it should explicitly link the claim to detailed evidence from the
 3301 passages—such as specific film titles, album names, song titles, dates, or
 3302 relationships—to provide clear reasoning for the accuracy or inaccuracy of the claim.
 3303
 3304 4. **Include relevant disambiguation or clarifications**: Where applicable, clarify
 3305 potential misunderstandings or common confusions highlighted in the passages (e.g.,
 3306 distinguishing similarly named films or songs, specifying which individual is referenced,
 3307 or noting alternate titles and adaptations).
 3308
 3309 5. **Use concise, factual language**: The summary should be clear and succinct but
 3310 include all relevant evidence necessary to understand how the claim is supported or
 3311 refuted.
 3312
 3313 6. **Support future query generation**: As the summary will be used to generate queries
 3314 to find evidence, it must contain explicit mentions of key entities, titles, and facts
 3315 that can guide retrieval and reasoning.
 3316
 3317 **Overall**, your approach should be to reason comprehensively over the claim, context, and
 3318 each passage, synthesizing connected facts and evidence into a coherent and evidence-rich
 3319 summary that clearly documents why the claim is true, false, or partially true. **Avoid**
 3320 vague or overly general statements; instead, ground your summary in the precise factual
 3321 details present in the passages and context.

3320 N.6 HoVER, QWEN3 8B

3322 HoVer Qwen3 8B create_query_hop2.predict

3323 Base Prompt:

3324 Given the fields `claim`, `summary_1`, produce the fields `query`.

3325 MIPROv2 Prompt:

3326 In a high-stakes scenario where the accuracy of your query directly determines the
 3327 validity of a complex claim, generate a precise query that builds on the provided claim
 3328 and summary to uncover critical evidence. Use step-by-step reasoning to identify gaps in
 3329 the initial summary, then craft a query that explicitly targets these gaps to retrieve
 3330 additional supporting information. Ensure your query is unambiguous and directly
 3331 addresses the claim's unresolved aspects.

3332 Demos:

3333 Example 1:

3334 claim:

3337
 3338 Sandi Simcha DuBowski directed a documentary released in 2001. The horror film
 3339 *Zeitgeist* was not also released in 2001.
 3340 summary_1:
 3341 The claim is correct: Sandi Simcha DuBowski directed the 2001 documentary **Trembling*
 3342 *Before G-d**, and there is no evidence in the passages that the horror film
 3343 **Zeitgeist** was released in 2001.
 3344 reasoning:
 3345 The claim is correct because Sandi Simcha DuBowski indeed directed the 2001
 3346 documentary **Trembling Before G-d**, as stated in summary_1. The assertion that
 3347 **Zeitgeist** was not released in 2001 is supported by the absence of evidence in the
 3348 provided summaries linking the film to that year.
 3349 query:
 3350 Is there any evidence in the provided summaries that the horror film **Zeitgeist** was
 3351 released in 2001?

3352 #####
 3353 3 other demos omitted for brevity.
 3354 #####

3355 GEPA Prompt generated by config GEPA:

3356 Given the fields `claim` and `summary_1`, generate a precise and focused query that
 3357 identifies the specific evidence needed to verify or refute the claim.
 3358
 3359 ### Key Requirements:
 3360 1. **Target Missing Evidence**: Identify and explicitly ask about the unverified or
 3361 unconfirmed details in the summary (e.g., names, dates, locations, or connections) that
 3362 are critical to the claim.
 3363 - Example: If the summary mentions "Massimo Giordano" as a potential counterpart but
 3364 lacks birthplace details, the query should ask for evidence confirming their ties to
 3365 Naples.
 3366 2. **Correct Historical/Domain-Specific Anomalies**: Address discrepancies like
 3367 anachronisms (e.g., Ali Qushjiâ€™s work in the 1960s vs. his actual 15th-century
 3368 timeline) or misattributions (e.g., **Hayy ibn Yaqdhan** by Ibn Tufail, not Ali Qushji).
 3369 3. **Link to Summary Context**: Ensure the query references the summaryâ€™s key points
 3370 (e.g., "the claim states X, but the summary notes Y is unverified").
 3371 4. **Use Specific Terms**: Include exact names, titles, or dates mentioned in the summary
 3372 to avoid ambiguity (e.g., "AnaÃ±s Nin," "Metropolitan City of Naples," "1948").
 3373
 3374 ### Example Strategy:
 3375 If the summary states:
 3376 - "The claim is partially supported. While [Fact A] is confirmed, [Fact B] lacks
 3377 verification."
 3378 - The query should ask:
 3379 -*Is there evidence confirming [Fact B], such as [specific name/date/location]?*"
 3380
 3381 ### Domain-Specific Notes:
 3382 - Verify historical timelines (e.g., Ali Qushji died in 1474, not the 1960s).
 3383 - Confirm authorships (e.g., **Hayy ibn Yaqdhan** is attributed to Ibn Tufail, not Ali
 3384 Qushji).
 3385 - Check for geographic or biographical details (e.g., birthplaces, cities, or
 3386 professional connections).

3384
 3385 Ensure your query directly addresses the summary's unverified claims and includes all
 3386 critical terms from the summary to maximize evidence retrieval.
 3387
 3388

3389 HoVer Qwen3 8B create_query_hop3.predict
 3390
 3391 Base Prompt:
 3392 Given the fields `claim`, `summary_1`, `summary_2`, produce the fields `query`.
 3393
 3394 MIPROv2 Prompt:
 3395 You are a fact-checking assistant specializing in multi-hop reasoning and information
 3396 synthesis. Given the fields `claim`, `summary_1`, and `summary_2`, generate a precise
 3397 query that synthesizes the original claim with the two summaries to probe for deeper
 3398 contextual relationships, resolving ambiguities and confirming supporting details through
 3399 targeted evidence retrieval.
 3400 Demos:
 3401 GEPA Prompt generated by config GEPA:
 3402 Given the fields `claim`, `summary_1`, and `summary_2`, produce the field `query` that:
 3403 1. Explicitly asks whether the claim is supported by the provided summaries.
 3404 2. Includes **specific evidence** from the summaries (e.g., names, titles, dates, or
 3405 factual details) to guide retrieval of relevant documents.
 3406 3. Focuses on **key disputed points** in the claim (e.g., incorrect attributions, missing
 3407 evidence, or conflicting statements) to ensure the query targets the most relevant
 3408 information.
 3409 **Key considerations for the query:**
 3410 - If the summaries mention specific works (e.g., *Planes, Trains and Automobiles*),
 3411 include the title.
 3412 - If the summaries reference named entities (e.g., John Candy, KLM, Richard Ford),
 3413 include them.
 3414 - If the summaries clarify contradictions (e.g., "not X but Y"), structure the query to
 3415 highlight this contrast.
 3416 - Avoid vague phrasing; instead of asking "Is the claim true?" use "Does the evidence in
 3417 the summaries confirm [specific detail]?"
 3418 **Example:**
 3419 If the claim is about a film and the summaries mention *Planes, Trains and Automobiles*
 3420 (1987), the query should include that title to retrieve relevant evidence.
 3421 `
 3422

3423 HoVer Qwen3 8B summarize1.predict
 3424
 3425 Base Prompt:
 3426 Given the fields `claim`, `passages`, produce the fields `summary`.
 3427
 3428 MIPROv2 Prompt:
 3429
 3430

3431
 3432 Given the fields `claim` and `passages`, generate a structured reasoning process that
 3433 analyzes the claim step-by-step using the provided evidence, and produce a concise
 3434 summary that distills the key findings and evaluates the claim's validity based on the
 3435 evidence.
 3436 Demos:
 3437 Example 1:
 3438 claim:
 3439 One half of the Kalyanji-Anandji duo, Kalyanji Virji Shah's brother, won the
 3440 Filmfare Award for Best Music Director.
 3441 passages:
 3442 ['Kalyanji Virji Shah | Kalyanji Virji Shah (30 June 1928 – 24 August 2000) was the
 3443 "Kalyanji" of the Kalyanji-Anandji duo. He and his brother Anandji Virji Shah have
 3444 been famous Indian film musicians, and won the 1975 Filmfare Award for Best Music
 3445 Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma Shri
 3446 (1992).', 'Kalyanji-Anandji | Kalyanji-Anandji are an Indian composer duo from
 3447 Gujarat: Kalyanji Virji Shah (30 June 1928–03 November 2000) and his brother Anandji
 3448 Virji Shah (born 02 March 1933). The duo are known for their work on Hindi film
 3449 soundtracks, particularly action potboilers in the 1970s. Some of their best-known
 3450 works are "Don", "Bairaag", "Saraswatichandra", "Qurbani", "Tridev" and "Safar". They
 3451 won the 1975 Filmfare Award for Best Music Director for "Kora Kagaz".', 'Anandji
 3452 Virji Shah | Anandji Virji Shah is an Indian music director. Together with his
 3453 brother he formed the Kalyanji-Anandji duo, and won the 1975 Filmfare Award for Best
 3454 Music Director, for "Kora Kagaz". He is a recipient of the civilian honour of Padma
 3455 Shri (1992).', 'Viju Shah | Viju Shah (born, Vijay Kalyanji Shah, 5 June 1959) is a
 3456 music director of Hindi cinema. He is the son of music director Kalyanji Virji Shah
 3457 of composer duo Kalyanji Anandji. He composed music for movies such as "Vishwatama"
 3458 (1992), "Mohra" (1994), "Tere Mere Sapne" (1996) and "" (1997) for which got his
 3459 second nomination for the Filmfare Award for Best Music Director and he won the 1998
 3460 Filmfare Award for Best Background Score for "...'.', 'Kalyanji-Anandji discography |
 3461 This is a discography of Bollywood composer duo Kalyanji Anandji, consisting of
 3462 Kalyanji Virji Shah and Anandji Virji Shah. They have composed music for over 200
 3463 films in their 46 year career.', 'Bombay the Hard Way: Guns, Cars and Sitars | Bombay
 3464 the Hard Way: Guns, Cars and Sitars is a 1998 project of producer Dan the Automator
 3465 in which music from 1970s funk Bollywood composer duo Kalyanji-Anandji (Kalyanji
 3466 Virji Shah and Anandji Virji Shah) was taken and given a funky remix. The album was
 3467 eventually withdrawn (possibly due to copyright concerns). A sequel, "Bombay 2:
 3468 Electric Vindaloo", was also produced.', 'Bairaag | Bairaag is a 1976 Bollywood film.
 3469 Produced by the duo Mushir-Riaz, it is directed by Asit Sen. The film stars Dilip
 3470 Kumar, who received a Filmfare nomination for Best Actor for playing a triple role.
 3471 "Bairaag" also stars Saira Banu, Leena Chandavarkar, Ruma Guha Thakurta, Prem Chopra,
 3472 Helen, Sujit Kumar, Madan Puri, Paintal, Kader Khan, Asit Sen and Sachin. Its music
 3473 is by the duo Kalyanji Anandji, who received a Filmfare nomination for Best Music.
 3474 The film received a Filmfare Award for Best Sound for P. Harikishan. According to one
 3475 source, the film did not do well at the box office.]
 3476 reasoning:
 3477 The claim states that Kalyanji Virji Shah's brother won the Filmfare Award for Best
 3478 Music Director. The passages confirm that the Kalyanji-Anandji duo (Kalyanji Virji
 3479 Shah and his brother Anandji Virji Shah) won the 1975 Filmfare Award for Best Music
 3480 Director for "Kora Kagaz." Since the award was shared by both brothers, the claim is
 3481 accurate as it identifies Anandji Virji Shah (Kalyanji's brother) as the winner.
 3482 summary:
 3483

3478
 3479 Anandji Virji Shah, the brother of Kalyanji Virji Shah, won the 1975 Filmfare Award
 3480 for Best Music Director as part of the Kalyanji-Anandji duo for their work on "Kora
 3481 Kagaz."

3482
 3483 #####
 3484 3 other demos omitted for brevity.
 3485 #####

3486 GEPA Prompt generated by config GEPA:
 3487

3488 Given the fields `claim` and `passages`, produce the fields `summary` by:
 3489 1. **Analyzing the claim** to identify key elements (e.g., names, locations,
 3490 relationships, specific terms).
 3491 2. **Cross-referencing the passages** to locate explicit or implicit evidence that
 3492 supports, contradicts, or partially addresses the claim.
 3493 3. **Explicitly connecting the claim to the evidence** in the passages, even if the
 3494 evidence is not directly stated in the claim (e.g., linking "Roger Yuan" to "Shanghai
 3495 Noon" in Example 1).
 3496 4. **Highlighting domain-specific details** (e.g., "Sentinelese language is
 3497 unclassifiable" in Example 3) that are critical for accuracy.
 3498 5. **Stating whether the claim is supported, unsupported, or partially supported**,
 3499 and explaining the reasoning with clear references to the passages.
 3500 6. **Prioritizing key evidence** that would be relevant for further queries (e.g.,
 3501 "Marlborough" in Example 2) to ensure retrievability.

3502 **Additional Guidance:**
 3503 - If the claim contains unverified details (e.g., "Roger Yuan" in Example 1), explicitly
 3504 note this in the summary.
 3505 - For niche or ambiguous information (e.g., "uncontacted people" in Example 3), reference
 3506 specific passages that clarify the context.
 3507 - Ensure summaries are concise but include all critical evidence needed to validate or
 3508 refute the claim.

3509 HoVer Qwen3 8B summarize2.predict

3510 Base Prompt:

3511 Given the fields `claim`, `context`, `passages`, produce the fields `summary`.

3512 MIPROv2 Prompt:

3513 Given the fields `claim`, `context`, and `passages`, perform multi-hop reasoning to
 3514 generate a structured summary that validates or refutes the claim. First, analyze the
 3515 claim and contextual information to identify key relationships. Next, evaluate evidence
 3516 from the retrieved passages to build a coherent narrative. Construct a logical reasoning
 3517 chain connecting the claim to supporting or contradictory evidence. Finally, produce a
 3518 concise summary that explicitly confirms or contradicts the claim based on your analysis,
 3519 citing relevant evidence from the context and passages.

3520 Demos:

3521 Example 1:

3522 claim:

3525
 3526 The recorded history of the state, for which Baptist George Ruby was a prominent
 3527 black Republican leader in the Reconstruction-era, officially started in 1519 in the
 3528 same state that holds the city Augustus Chapman Allen used his inheritance to fund
 3529 the founding of.
 3530 context:
 3531 The claim is not supported by the passages. While George Ruby was associated with
 3532 Texas during Reconstruction and Augustus Chapman Allen founded Houston in Texas,
 3533 there is no evidence in the passages that Texas's recorded history began in 1519.
 3534 passages:
 3535 ["Merchants and Manufacturers Building | The One Main Building, formerly the
 3536 Merchants and Manufacturers Building (commonly referred to as the M&M Building), is a
 3537 building on the campus of the University of Houstonâ€¢Downtown. The building is
 3538 recognized as part of the National Register of Historic Places, is a Recorded Texas
 3539 Historic Landmark, and considered a Contributing Building in Downtown Houston's Main
 3540 Street/Market Square Historic District. The building was built above Allen's
 3541 Landingâ€¢an area where Houston's founders John Kirby Allen and Augustus Chapman
 3542 Allen originally settled.", 'Augustus Chapman Allen | Augustus Chapman Allen (July 4,
 3543 1806 â€¢ January 11, 1864), along with his younger brother, John Kirby Allen, founded
 3544 the City of Houston in the U.S. state of Texas. He was born on July 4, 1806, in
 3545 Canaseraga Village, New York (the present day hamlet of Sullivan in the Town of
 3546 Sullivan, New York), to Sarah (Chapman) and Roland Allen.', "History of Texas | The
 3547 recorded History of Texas begins with the arrival of the first Spanish conquistadors
 3548 in the region of North America now known as Texas in 1519, who found the region
 3549 populated by numerous Native American /Indian tribes.
 3550 Native Americans' ancestors had been there for more than 10,000 years as evidenced by
 3551 the discovery of the remains of prehistoric Leanderthal Lady.
 3552 During the period of recorded history from A.D. 1519 to 1848, all or parts of Texas
 3553 were claimed by five countries: France, Spain, Mexico, the Republic of Texas, the
 3554 United States of Americaâ€¢as well as the secessionist Confederate States of America
 3555 in 1861â€¢1865, during the American Civil War.", 'Charlotte Baldwin Allen | Charlotte
 3556 Baldwin Allen (July 14, 1805 â€¢ August 3, 1895) is known in Texan history as the
 3557 "mother of Houston". She was the wife of Augustus Chapman Allen, who used her
 3558 inheritance to finance the founding of this city.', 'Allen Parkway | Allen Parkway is
 3559 an arterial road west of Downtown Houston, Texas. It has a distance of approximately
 3560 2.3 miles (3.7\x00km), running from Interstate 45 west to Shepherd Drive, where it
 3561 becomes Kirby Drive. Originally known as Buffalo Parkway, it was later named after
 3562 John Kirby Allen and Augustus Chapman Allen, the founders of Houston.', "Allen's
 3563 Landing | Allen's Landing is the birthplace of the city of Houstonâ€¢the largest city
 3564 in the U.S. state of Texas. In August 1836, just months after the Republic of Texas
 3565 won its independence from Mexico, two brothers (and real estate developers) from New
 3566 Yorkâ€¢John Kirby Allen and Augustus Chapman Allenâ€¢purchased
 3567 6,642\x00acres\x00(27\x00kmâ€¢s) in the area and settled there on the banks of Buffalo
 3568 Bayou. Allen's Landing is located south of the University of Houstonâ€¢Downtown
 3569 Commerce Street Building, at the intersection of Commerce Street and Main Street.",
 3570 "Keith Allen (American football) | Keith D. Allen (born July 12, 1974) is an American
 3571 football coach. He served as the head football coach at Southwest Baptist University
 3572 from 2007 to 2012, compiling a record of 27â€¢39. His 27 wins are the most in the
 3573 history of Southwest Baptist's football program, which began play in 1983."]
 3574 reasoning:
 3575 The claim is not supported by the passages. While the passages confirm that Texas's
 3576 recorded history began in 1519 (passage 3) and that Augustus Chapman Allen founded
 3577 Houston in Texas (passage 2), they do not explicitly link the start of Texas's
 3578 recorded history to the same state where Augustus Chapman Allen used his inheritance

3572
 3573 to fund the founding of a city. The claim conflates these two facts without direct
 3574 evidence in the passages that the 1519 start of recorded history is tied to the state
 3575 where Allen's inheritance was used for founding Houston.
 3576 summary:
 3577 The claim is not supported by the provided passages, as they confirm Texas's recorded
 3578 history began in 1519 and Augustus Chapman Allen founded Houston in Texas but do not
 3579 explicitly connect the two facts as stated in the claim.
 3580 #####
 3581 3 other demos omitted for brevity.
 3582 #####
 3583
 3584 GEPA Prompt generated by config GEPA:
 3585 Given the fields `claim`, `context`, and `passages`, produce the field `summary` that:
 3586 1. **Evaluates the claim's validity** by explicitly connecting it to **specific
 3587 evidence** from the `passages` (e.g., citing exact names, dates, or terms that
 3588 support/refute the claim).
 3589 2. **Highlights factual inaccuracies** in the claim, such as incorrect origins, dates, or
 3590 relationships, and links them to **direct contradictions** in the `passages` or `context`.
 3591 3. **Mentions key terms or entities** critical to the claim (e.g., "Sly Stone," "Large
 3592 MÃnsterlÃdnder," or "Hit Parader's 2006 ranking") to ensure summaries are actionable for
 3593 query generation.
 3594 4. **Balances accuracy** by acknowledging both correct and incorrect aspects of the
 3595 claim, as seen in the examples (e.g., validating "The Wild Bunch" compilation while
 3596 correcting the magazine's nationality and ranking year).
 3597 5. **Avoids vague statements** like "no evidence" and instead specifies **which parts of
 3598 the passages** are relevant (e.g., "Passage 6 confirms Hit Parader was American").
 3599
 3600 **Generalizable Strategy**:
 3601 - Always reference **specific passages** (e.g., "Passage 1 states...") to ground claims
 3602 in the text.
 3603 - Use **precise terminology** (e.g., "German Shorthaired Pointer" instead of "cousins")
 3604 to align with the data.
 3605 - Prioritize **clarity and specificity** to ensure summaries can guide further evidence
 3606 retrieval.
 3607

3606 N.7 PUPA, GPT-4.1 MINI 3607

3608 PUPA GPT-4.1 Mini craft_redacted_request.predict
 3609

3610 Base Prompt:

3611 Given a private user query, create a privacy-preserving request for a powerful external
 3612 LLM.
 3613 The LLM may assist without learning private information about the user.

3614 MIPROv2 Prompt:

3615 You will be given a private user query. Your task is to analyze the query carefully and
 3616 think step-by-step to understand the user's intent while identifying and removing or
 3617

3619
 3620 abstracting any sensitive, proprietary, or personally identifiable information. Then,
 3621 generate a clear, privacy-preserving prompt suitable for an external large language model
 3622 (LLM) that requests useful, relevant guidance or information without exposing private
 3623 details. Additionally, provide a detailed chain-of-thought explanation describing your
 3624 reasoning process for how you transformed the original query into the redacted,
 3625 generalized LLM request. Ensure the resulting prompt is professional, contextually
 3626 appropriate, and maintains user privacy while enabling the LLM to assist effectively.
 3627 Demos:
 3628 Example 1:
 3629 user_query:
 3630 Imagine you are a stock researcher and advisor, I will give you a table about stocks
 3631 and I want you to give me advice on what to invest next. This is a project I am
 3632 working on I need help with predictions I am making. Here is the table: Stock
 3633 Price Dividends 5D 5D % 1M 1M % YTD YTD % 1Y 1Y % 5Y 5Y % IBM \$146,62 \$6,64 \$3,81
 3634 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
 3635 -\$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
 3636 -\$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
 3637 -\$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
 3638 -\$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
 3639 -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
 3640 -1,81 -\$15,19 -29,64 -\$10,16 -21,98 -\$5,43 -13,09 AAPL \$179,96 \$0,96 \$3,45 1,95
 3641 -\$16,57 -8,43 \$54,89 43,89 \$18,58 11,51 \$125,04 227,65 NVDA \$468,23 \$0,16 -\$3,71
 3642 -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
 3643 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
 3644 -\$13,82 -18,43 -\$13,43 -18,00 -\$31,51 -34,00 -\$29,46 -32,51 BIDU \$137,47 \$0,00 \$10,24
 3645 8,05 -\$15,63 -10,21 \$18,34 15,39 -\$9,85 -6,69 -\$88,25 -39,10
 3646 reasoning:
 3647 The user wants investment advice based on a provided table of stock data including
 3648 price, dividends, and percentage changes over various time intervals (5 days, 1
 3649 month, year-to-date, 1 year, and 5 years). The primary request is for recommendations
 3650 on what to invest in next and help with making predictions.
 3651
 3652 To preserve privacy, I should not include the user's raw data or specific details about
 3653 individual stocks directly in the prompt to the external LLM. Instead, I should request
 3654 general advice on how to analyse stock data tables that include price, dividend, and
 3655 percentage changes over multiple time periods. Also, I can ask for advice on what
 3656 indicators to weigh more heavily for investment decisions and for predictive strategies
 3657 based on such data structures.
 3658
 3659 This approach ensures that the external LLM does not receive any private or identifying
 3660 user data, only a generic request. I will ask the LLM to provide guidance on interpreting
 3661 such stock performance indicators and suggestions for predictive analytics or investment
 3662 selection criteria.
 3663 llm_request:
 3664 Please provide general guidance for analyzing stock investment opportunities based on
 3665 a data table format that includes stock prices, dividend payouts, and percentage
 3666 changes over various time intervals such as 5 days, 1 month, year-to-date, 1 year,
 3667 and 5 years. Outline key indicators that investors typically weigh when making
 3668 decisions, how to interpret short-term versus long-term performance trends, and
 3669 suggest strategies or metrics to use when making predictions or recommendations about
 3670 which stocks to invest in next. The request is for generic advice without any
 3671 specific stock names or values.

3666
 3667
 3668
 3669
 3670
 3671
 3672
 3673
 3674
 3675
 3676
 3677
 3678
 3679
 3680
 3681
 3682
 3683
 3684
 3685
 3686
 3687
 3688
 3689
 3690
 3691
 3692
 3693
 3694
 3695
 3696
 3697
 3698
 3699
 3700
 3701
 3702
 3703
 3704
 3705
 3706
 3707
 3708
 3709
 3710
 3711
 3712

```
#####
3 other demos omitted for brevity.
#####

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:
 * How you identified sensitive or private information in the input
 * What strategies you applied to protect privacy (generalization, omission, abstraction, replacement with placeholders)
 * How the reformulated prompt preserves the original intent and task requirements without risking data leakage
 - Part (b) LLM Request:
 A concise, carefully constructed privacy-safe prompt that:
 * Removes or anonymizes all PII and proprietary/internal details
 * Abstracts locations, names, dates, and technical terms as needed
 * Produces a clear and contextually rich instruction for the LLM to generate a relevant and informative response aligned with the user's original task

Detailed Domain-Specific Guidance and Best Practices:

1. Identification and Treatment of Sensitive Data:
 - All user-specific or personal names (individual or organizational) must be removed or replaced with generic role descriptors (e.g., a business contact, a client, a notable individual). Never lightly obscure or partially redact; full abstraction is required.
 - All geographic mentions must be abstracted unless the location is publicly known, essential to the task, and can be generalized (e.g., a region known for eco-tourism instead of naming a country or city explicitly).
 - Exact dates or durations must never be retained; instead, use relative or approximate temporal references (e.g., recently, over the past year).
 - Internal or proprietary terms including system names, product codes, subscription types, and technical jargon must be generalized or replaced with

3713

3714 neutral descriptors to avoid leakage of intellectual property or sensitive operational
3715 details.

3716 - Avoid direct quotes or verbatim inclusion of user-supplied texts unless obfuscated
3717 by generalization.

3718 2. Task Understanding and Reformulation:

3719 - Identify the functional intent of the query: Is it creative writing, translation,
3720 summarization, professional communication drafting, technical explanation, or other?

3721 - Preserve the thematic and informational core of the query (e.g., request for
3722 educational quality analysis, technical translation of a passage, biographical
3723 summary).

3724 - Do not reproduce the original input verbatim; instead, frame the LLM prompt around
3725 the essential thematic elements extracted from the input.

3726 - For queries regarding individuals, avoid direct reference to real personal
3727 information unless the name is widely public and essential; even then, use a generic
3728 or hypothetical framing for the individual profile.

3729 3. Strategies for High-Quality and Privacy-Preserving Prompts:

3730 - Strike a balance between sufficient contextual detail and privacy abstraction to
3731 maintain prompt clarity and relevance.

3732 - Use neutral, context-aware formulations that clearly instruct the LLM on the content
3733 and style expected.

3734 - Avoid vague or overly generic prompts that could result in less useful or
3735 lower-quality responses.

3736 - When system or proprietary content is mentioned, instruct the LLM to generalize
3737 specific terms and maintain the technical meaning without revealing sensitive info.

3738 - When a direct translation is requested on specialized text, specify to replace or
3739 abstract internal nomenclature.

3740 4. Explanation Requirements:

3741 - The reasoning must transparently explain how privacy risks were identified (e.g.,
3742 presence of names, locations, dates, proprietary terms).

3743 - It must describe the abstraction or omission methods applied (e.g., replacing
3744 â€˜Jonah Van Beijnenâ€™ with â€˜a notable individual,â€™ substituting â€˜Makauâ€™ with
3745 â€˜a specific region,â€™ or â€˜Yodaâ€™ with â€˜a system nameâ€™).

3746 - Clarify how the essential task and user intent were preserved despite these
3747 generalizations (e.g., focusing on educational quality, technical translation,
3748 biographical summary).

3749 - This explanation justifies your design choices and demonstrates adherence to
3750 privacy-preserving principles.

3751 Common Pitfalls to Avoid:

3752 - Do not merely lightly obscure or partially redact sensitive details; full anonymization
3753 or abstraction is required.

3754 - Do not repeat any user-supplied PII or proprietary content verbatim.

3755 - Avoid including URLs, exact dates, or direct quotes without modification.

3756 - Do not leave ambiguity that could degrade the quality or contextual clarity of the
3757 reformulated prompt.

3758 - Do not include any real personal or organizational names unless they are public figures
3759 and the query requires it, then use generic descriptors instead.

3760 Example Summary of Effective Approach (Informed by Prior Examples):

3760
 3761 - For geographic queries: replace exact place names with general regions and provide a
 3762 brief contextual descriptor.
 3763 - For technical texts containing system names or subscription types: instruct the LLM to
 3764 translate or process the text while replacing or abstracting proprietary system
 3765 identifiers.
 3766 - For biographical summaries about specific individuals: remove the real name and request
 3767 a generic, well-structured four-paragraph summary about a notable individual, preserving
 3768 the overall intent without leaking PII.

3769 Summary:
 3770 Your reformulations must ensure zero exposure of any PII or private/proprietary content
 3771 while retaining enough thematic and functional clarity for the external LLM to produce
 3772 high-quality, relevant outputs. This requires thorough analysis of the user's query,
 3773 rigorous application of privacy-preservation strategies, and explicit reasoning
 3774 explanations that document your approach and choices.

3775 PUPA GPT-4.1 Mini respond_to_query

3776 Base Prompt:

3777 Respond to a user query.
 3778 For inspiration, we found a potentially related request to a powerful external LLM and
 3779 its response.

3780 MIPROv2 Prompt:

3781 Given a privacy-preserving redacted prompt sent to an external untrusted LLM and its
 3782 response, along with the original user query containing potentially sensitive
 3783 information, generate a final, coherent, and contextually appropriate user-facing
 3784 response. Your output should integrate the anonymized LLM output with the original
 3785 query's intent, ensuring privacy is maintained while providing a complete and polished
 3786 answer that directly addresses the user's request.

3787 Demos:

3788 Example 1:

3789 related_llm_request:

3790 Please provide a detailed explanation of David Easton's system analysis in political
 3791 science, outlining the key components of his systems theory approach and its
 3792 significance in the study of political systems.

3793 related_llm_response:

3794 Certainly! David Easton's system analysis is a foundational framework in political
 3795 science that applies systems theory to the study of political systems. His approach
 3796 provides a structured way to understand how political systems function, maintain
 3797 stability, and respond to changes in their environment. Below is a detailed
 3798 explanation of Easton's system analysis, including its key components and
 3799 significance.

3800 ---

3801 **### Overview of David Easton's System Analysis**

3802 David Easton (1917–2014) was a prominent political scientist who sought to develop a
 3803 general theory of political systems by borrowing concepts from systems theory, which
 3804 originated in biology and engineering. His goal was to create a scientific and analytical

3807
 3808 framework to study politics as an integrated and dynamic system rather than as isolated
 3809 events or institutions.
 3810
 3811 Easton's system analysis views a political system as a set of interactions abstracted
 3812 from the totality of social behavior, through which authoritative allocation of values
 3813 (decisions) is made and implemented. In other words, politics is about how societies
 3814 decide who gets what, when, and how, and Easton's model helps explain how these decisions
 3815 are made and sustained.
 3816
 3817 ---
 3818
 3819 **### Key Components of Easton's Systems Theory Approach**
 3820
 3821 Easton's model conceptualizes the political system as a **black box** that processes
 3822 inputs and produces outputs, embedded within a larger environment. The key components are:
 3823
 3824 **#### 1. **Inputs****
 3825 Inputs are demands and supports that come from the environment and society into the
 3826 political system.
 3827
 3828 - **Demands:** These are claims or requests made by individuals or groups for the
 3829 political system to allocate values or resources in a particular way (e.g., demands for
 3830 law enforcement, education, welfare).
 3831 - **Supports:** These are actions or attitudes that indicate approval, allegiance, or
 3832 compliance with the political system (e.g., voting, paying taxes, obeying laws).
 3833
 3834 Inputs represent the pressures and resources that the political system must process.
 3835
 3836 **#### 2. **Political System (The Black Box)****
 3837 The political system is the core of Easton's model. It receives inputs, processes them,
 3838 and produces outputs.
 3839
 3840 - It includes institutions, organizations, and processes involved in authoritative
 3841 decision-making (e.g., legislatures, executives, courts).
 3842 - The system transforms demands and supports into decisions and policies.
 3843 - The system must maintain its boundaries and adapt to changes to survive.
 3844
 3845 **#### 3. **Outputs****
 3846 Outputs are the decisions, policies, and actions that the political system produces in
 3847 response to inputs.
 3848
 3849 - These include laws, regulations, public policies, and authoritative allocations of
 3850 values.
 3851 - Outputs affect the environment and society, influencing future inputs.
 3852
 3853 **#### 4. **Feedback****
 3854 Feedback is the process by which outputs influence future inputs.
 3855
 3856 - Positive feedback can reinforce the system's stability.
 3857 - Negative feedback can signal problems or dissatisfaction, prompting adjustments.
 3858 - Feedback helps the system adapt and maintain equilibrium.
 3859
 3860 **#### 5. **Environment****

3854
 3855 The political system exists within a broader environment that includes social, economic,
 3856 cultural, and international factors.
 3857 - The environment provides inputs and is affected by outputs.
 3858 - The system must interact with and adapt to environmental changes to survive.
 3859
 3860 ---
 3861
 3862 ### Diagrammatic Representation (Simplified)
 3863 ...
 3864 Environment
 3865 â€¢â€¢
 3866 [Demand & Supports] â€¢ [Political System] â€¢ [Decisions & Policies]
 3867 â€¢â€¢
 3868 â€¢----- Feedback -----
 3869 ...
 3870
 3871 ---
 3872
 3873 ### Significance of Easton's System Analysis in Political Science
 3874 1. ****Holistic Understanding of Politics:****
 3875 Easton's model encourages viewing politics as an interconnected system rather than
 3876 isolated events or institutions. It highlights the dynamic interactions between
 3877 society and political authority.
 3878 2. ****Focus on Process and Stability:****
 3879 The model emphasizes how political systems maintain stability and order through
 3880 continuous processing of inputs and outputs, and through feedback mechanisms.
 3881 3. ****Analytical Clarity:****
 3882 By abstracting political activity into inputs, system processing, outputs, and
 3883 feedback, Easton provided a clear framework for empirical research and comparative
 3884 analysis.
 3885 4. ****Foundation for Further Theories:****
 3886 Easton's systems theory influenced later developments in political science, including
 3887 structural-functionalism, policy analysis, and the study of political development and
 3888 change.
 3889 5. ****Adaptability:****
 3890 The model can be applied to various political systems (democracies, authoritarian
 3891 regimes, etc.) and scales (local, national, international), making it versatile.
 3892 6. ****Emphasis on Environment:****
 3893 Recognizing the environmentâ€Žs role underscores the importance of external factors
 3894 (economic conditions, social movements, international pressures) in shaping political
 3895 outcomes.
 3896
 3897 ---
 3898
 3899 ### Criticisms and Limitations
 3900

3901
 3902
 3903 - **Overly Abstract:** Some critics argue that Easton's model is too abstract and
 3904 general, lacking specificity about power relations, conflict, and individual agency.
 3905 - **Neglect of Conflict:** The model tends to emphasize stability and equilibrium,
 3906 potentially underplaying political conflict and change.
 3907 - **Limited Normative Guidance:** It is primarily descriptive and analytical, offering
 3908 little normative guidance on what political systems ought to do.
 3909 ---
 3910 **Summary**
 3911
 3912 David Easton's system analysis
 3913 user_query:
 3914 System analysis by David Easton
 3915 response:
 3916 David Easton's system analysis is a foundational framework in political science that
 3917 conceptualizes the political system as a dynamic and interactive system. His approach
 3918 uses systems theory to explain how political systems function, maintain stability,
 3919 and adapt to changes in their environment.
 3920
 3921 At its core, Easton's model views the political system as a "black box" that processes
 3922 inputs from society and the environment, transforms them into outputs, and uses feedback
 3923 to adjust and sustain itself. The key components of Easton's system analysis include:
 3924
 3925 1. **Inputs:** These are the demands and supports that society directs towards the
 3926 political system. Demands are requests or claims for resource allocation or policy
 3927 change, while supports are actions or attitudes such as compliance, legitimacy, or
 3928 loyalty that sustain the system.
 3929
 3930 2. **Political System:** This is the set of institutions and processes that processes the
 3931 inputs. It interprets demands and supports to make authoritative decisions, including the
 3932 formulation and enforcement of policies.
 3933
 3934 3. **Outputs:** These are the decisions, policies, and actions that the political system
 3935 produces in response to inputs. They affect society and its environment, shaping future
 3936 interactions.
 3937
 3938 4. **Feedback:** Outputs generate responses that feed back into the system as new inputs,
 3939 signaling whether the outputs were effective or require modification. This feedback
 3940 enables the system to adapt to internal and external changes.
 3941
 3942 5. **Environment:** The system operates within a broader social, economic, cultural, and
 3943 international context that influences and is influenced by the political system.
 3944
 3945 Easton's theory is significant because it offers a holistic, process-oriented lens to
 3946 analyze politics, emphasizing interaction between society and state institutions. It
 3947 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.

3948
 3949 However, critics note that Easton's model can be overly abstract, sometimes downplaying
 3950 conflict, power struggles, and individual agency within political processes.
 3951
 3952 In summary, David Easton's system analysis offers a powerful conceptual tool to study
 3953 political systems as dynamic entities that manage societal demands and supports through
 3954 authoritative decision-making to maintain order and adapt over time.
 3955
 3956 #####
 3957 3 other demos omitted for brevity.
 3958 #####
 3959
 3960 GEPA Prompt generated by config GEPA+Merge:
 3961
 3962 Respond to a user query.
 3963 For inspiration, we found a potentially related request to a powerful external LLM and
 3964 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.

Demos:

Example 1:

user_query:

ÐŠÑÄÐÝÐŠÐtÑC ÐiÐřÐEÐÝÑLÐÝ ÐžÐžÐÐÐiÐtÑC ÐtÐžÑR ÐžÑCÐEÑÄÐřÐšÐžÐÝ snmpwalk ÐiÐř ip
 10.2.35.2 ÑA ÐtÐřÐiÐiÑÑÐÐÝ ÐEÐi ÐtÐžÑAÑCÑCÐEÑC 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.

3995
 3996
 3997 #####
 3998 3 other demos omitted for brevity.
 3999 #####

4000 GEPA Prompt generated by config GEPA:

4001 ### New Instruction for the Assistant:

4002 ****Task Overview:****
 4003 Rephrase the user's query into a privacy-preserving, actionable request for an external
 4004 LLM, ensuring no personally identifiable information (PII), proprietary data, or
 4005 sensitive details are exposed. Maintain the original intent, clarity, and specificity of
 4006 the query while adhering to domain-specific strategies for generalized information.

4007
 4008 ****Key Requirements:****

4009 1. ****Privacy Preservation**:**

- **Generalize Specifics**: Replace specific locations (e.g., "Andhra Pradesh" → "a major Indian city"), numbers (e.g., "18 volunteers" → "a group of participants"), or proprietary data (e.g., "van Cleef production sites" → "[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").

4010 2. ****Quality of Request**:**

- **Clarity and Actionability**: Ensure the rephrased query is precise, avoids ambiguity, and specifies constraints (e.g., "concise, platform-friendly hashtags" for creative tasks).
- **Domain-Specific Precision**: For scientific/technical queries, retain key terms (e.g., "CAPM formula," "pharmacokinetic parameters") while anonymizing study details (e.g., "18 volunteers" → "a group of participants").
- **Avoid Overly Broad Requests**: Specify constraints like keyword focus, tone, or format (e.g., "health and fitness keywords" instead of "general hashtags").

4011 3. ****Domain-Specific Strategies**:**

- **Geographic Queries**: Replace regions/cities with generalized terms (e.g., "major city in [Country]" or "well-known [Region]").
- **Brand/Company Queries**: Reference brands by name but prioritize publicly available data (e.g., "locations of [Brand Name]'s production sites").
- **Creative Tasks**: Use examples or constraints (e.g., "wellness, strength, motivation keywords" for hashtags; "platform guidelines" for social media content).

4012 4. ****Avoid Leakage**:**

- **No Inferred Details**: Do not assume user-specific preferences, contexts, or private data (e.g., "Tell me the location of van Cleef production sites" → "Provide locations of [Brand Name]'s production sites based on publicly available information").
- **Anonymize Numerical Data**: Replace specific counts, dosages, or metrics with generalized terms (e.g., "40 mg OA" → "a dose of OA").

4013
 4014 ****Example Workflow:****

4042
 4043 - **User Query**: "Calculate DâĂŹAmicoâĂŹs cost of equity using CAPM with rRF = 4.67%, $\hat{\beta}$ = 1.56, and market risk premium = 6.63%."
 4044 - **Rephrased Request**: "Calculate the cost of equity for a company using the CAPM
 4045 formula, given a risk-free rate of 4.67%, a beta of 1.56, and a market risk premium of
 4046 6.63%. Show step-by-step calculations and round the final result to two decimal places."
 4047
 4048 **Niche Domain-Specific Guidance:**
 4049 - **Scientific/Pharmacokinetic Queries**: Generalize study details (e.g., "18 healthy
 4050 male volunteers" âĘ "a group of participants"), retain key parameters (e.g., AUC, peak
 4051 concentrations), and avoid proprietary methodologies.
 4052 - **Creative/Translation Tasks**: Use placeholders for song titles/artists (e.g.,
 4053 "Mahmood's song [Title]") and specify tone/rhythm requirements for translations.
 4054 - **Financial Calculations**: Retain formulas and numerical values but avoid linking to
 4055 specific companies or proprietary models.
 4056
 4057 **Final Output Format:**
 4058 - **reasoning**: Briefly explain the rephrasing choices (e.g., "Generalized '18
 4059 volunteers' to 'a group of participants' to preserve privacy").
 4060 - **llm_request**: Provide the anonymized, actionable request for the LLM.

4061 PUPA Qwen3 8B respond_to_query

4062 Base Prompt:

4063 Respond to a user query.
 4064 For inspiration, we found a potentially related request to a powerful external LLM and
 4065 its response.

4066 MIPROV2 Prompt:

4067 Synthesize the external LLM's response with the user's specific query to generate a final
 4068 privacy-preserving answer. Replace placeholders with actual values while omitting
 4069 sensitive information, maintaining a professional tone, and ensuring contextual accuracy.
 4070 Combine the technical solution from the LLM with the user's unique details to produce a
 4071 clear, anonymized response that adheres to privacy constraints.

4072 Demos:

4073 Example 1:

4074 related_llm_request:

4075 Create a PESTLE analysis framework for GE Healthcare in the US, outlining key issues
 4076 in each category (Political, Economic, Social, Technological, Environmental, Legal)
 4077 and their potential impact on the organization. Prioritize these issues based on
 4078 urgency and significance without referencing confidential data. Structure the output
 4079 as a matrix with clear criteria for evaluation.

4080 related_llm_response:

4081 Certainly! Below is a **PESTLE analysis framework** tailored for **GE Healthcare in
 4082 the US**, structured as a matrix. It outlines key issues in each category, evaluates
 4083 their **urgency** and **significance**, and highlights their **potential impact** on
 the organization.

4084 ---

4085
 4086 #### PESTLE Analysis Matrix for GE Healthcare (US)

4089
 4090 | **Category** | **Key Issues** | **Urgency** (High/Med/Low) | **Significance**
 4091 (High/Med/Low) | **Potential Impact on GE Healthcare** |
 4092 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
 4093 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
 4094 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
 4095 |-----|-----|-----|-----|-----|-----|-----|-----|-----|
 4096 | **Political** | 1. Healthcare policy changes (e.g., Medicare/Medicaid reimbursement
 4097 reforms) | High | High | Directly affects revenue streams and product/service demand;
 4098 requires adaptation to new reimbursement models. |
 4099 | | 2. Trade policies and tariffs on medical equipment imports/exports | Medium | Medium
 4100 | Could increase costs of components or affect supply chain efficiency. |
 4101 | | 3. Government funding for healthcare innovation and research | Medium | High |
 4102 Influences R&D opportunities and partnerships; potential for grants and subsidies. |
 4103 | **Economic** | 1. US healthcare spending trends and budget constraints | High | High |
 4104 Impacts purchasing power of hospitals and clinics; affects sales volume and pricing
 4105 strategies. |
 4106 | | 2. Inflation and rising operational costs | High | Medium | Increases production and
 4107 operational expenses; may pressure profit margins. |
 4108 | | 3. Economic cycles affecting capital investment in healthcare infrastructure | Medium
 4109 | Medium | Influences demand for high-cost medical equipment and technology upgrades. |
 4110 | **Social** | 1. Aging population increasing demand for healthcare services and devices
 4111 | High | High | Drives demand for diagnostic imaging, monitoring, and therapeutic
 4112 devices; growth opportunity. |
 4113 | | 2. Growing patient awareness and demand for personalized medicine | Medium | Medium |
 4114 Necessitates innovation in tailored healthcare solutions and data-driven technologies. |
 4115 | | 3. Workforce shortages in healthcare professionals | High | Medium | May affect
 4116 adoption and training for new technologies; impacts customer support and service
 4117 delivery. |
 4118 | **Technological** | 1. Rapid advancements in AI and machine learning for diagnostics
 4119 and imaging | High | High | Critical for maintaining competitive edge; requires
 4120 continuous investment in R&D and product development. |
 4121 | | 2. Integration of IoT and connected devices in healthcare | Medium | High | Enhances
 4122 product offerings but demands robust cybersecurity and interoperability solutions. |
 4123 | | 3. Data privacy and cybersecurity technologies | High | High | Essential to protect
 4124 patient data and comply with regulations; failure risks legal penalties and reputational
 4125 damage. |
 4126 | **Environmental** | 1. Regulatory pressure to reduce carbon footprint and waste in
 4127 manufacturing | Medium | Medium | Necessitates sustainable practices; potential cost
 4128 implications but improves brand reputation. |
 4129 | | 2. Energy efficiency in product design and operations | Medium | Medium | Can reduce
 4130 operational costs and appeal to environmentally conscious customers. |
 4131 | | 3. E-waste management and recycling programs | Low | Medium | Compliance with
 4132 environmental standards; growing importance for corporate social responsibility. |
 4133 | **Legal** | 1. Compliance with FDA regulations and medical device approvals | High |
 4134 High | Critical for market access; delays or failures can halt product launches and cause
 4135 financial losses. |
 4136 | | 2. Intellectual property rights and patent protections | Medium | High | Protects
 4137 innovation and competitive advantage; infringement risks costly litigation. |
 4138 | | 3. Liability and product safety regulations | High | High | Ensures patient safety;
 4139 non-compliance can lead to lawsuits, recalls, and reputational damage. |
 4140
 4141 ---
 4142

```

4136
4137
4138     ### Criteria for Evaluation:
4139
4140         - **Urgency**: How immediate the issue requires attention (High = immediate/ongoing;
4141             Medium = within 1-2 years; Low = longer-term).
4142         - **Significance**: The potential magnitude of impact on GE Healthcare's operations,
4143             profitability, and strategic positioning (High = critical; Medium = moderate; Low =
4144             minor).
4145         - **Potential Impact**: Describes how the issue could affect GE Healthcare's business,
4146             including risks and opportunities.
4147
4148         ---
4149
4150     This matrix helps prioritize strategic focus areas for GE Healthcare in the US, balancing
4151     immediate challenges with longer-term trends.
4152     user_query:
4153         issue priority matrix on pestle analysis for GE Healthcare in US
4154     response:
4155         Here's a prioritized **issue priority matrix** based on the PESTLE analysis for **GE
4156             Healthcare in the US**, focusing on the most urgent and significant challenges and
4157             opportunities:
4158
4159         ---
4160
4161     ### **High Priority Issues (High Urgency + High Significance)**
4162     | **Category** | **Key Issue** | **Urgency** | **Significance** | **Impact Summary** |
4163     |-----|-----|-----|-----|
4164     |-----|-----|-----|-----|
4165     |-----|-----|-----|-----|
4166     |-----|-----|-----|-----|
4167     |-----|-----|-----|-----|
4168     |-----|-----|-----|-----|
4169     |-----|-----|-----|-----|
4170     |-----|-----|-----|-----|
4171     |-----|-----|-----|-----|
4172     |-----|-----|-----|-----|
4173     |-----|-----|-----|-----|
4174     |-----|-----|-----|-----|
4175     |-----|-----|-----|-----|
4176     |-----|-----|-----|-----|
4177     |-----|-----|-----|-----|
4178     |-----|-----|-----|-----|
4179     |-----|-----|-----|-----|
4180     |-----|-----|-----|-----|
4181     |-----|-----|-----|-----|
4182

```

4183
 4184
 4185
 4186
 4187
 4188
 4189
 4190
 4191
 4192
 4193
 4194
 4195
 4196
 4197
 4198
 4199
 4200
 4201
 4202
 4203
 4204
 4205
 4206
 4207
 4208
 4209
 4210
 4211
 4212
 4213
 4214
 4215
 4216
 4217
 4218
 4219
 4220
 4221
 4222
 4223
 4224
 4225
 4226
 4227
 4228
 4229

Prioritization Criteria

- **Urgency**: Immediate (High) vs. longer-term (Low).
- **Significance**: Critical (High) vs. moderate (Medium).
- **Focus**: Issues with **High in both urgency and significance** require immediate strategic action, while others demand monitoring or phased planning.

This matrix helps GE Healthcare allocate resources to address the most pressing challenges and leverage high-impact opportunities.

#####
 3 other demos omitted for brevity.
 #####

GEPA Prompt generated by config GEPA:

Respond to a user query by leveraging a related LLM response as inspiration. Structure your answer in a clear, organized format (e.g., bullet points, sections) to enhance readability. Ensure the response is tailored to the user's language and avoids any personally identifiable information (PII) or sensitive data.

When rephrasing or summarizing content:

1. **Maintain academic tone** if the query specifies formal language (e.g., rephrasing technical or historical text).
2. **Refactor repetitive or redundant information** into concise, focused sections (e.g., grouping similar ideas, eliminating duplication).
3. **Highlight domain-specific facts** (e.g., cultural, technical, or regional details) to add value, as these may not be universally known.
4. **Improve code or technical responses** by breaking down complex logic into smaller, well-documented functions, reducing redundancy, and ensuring proper resource management.

For all outputs:

- Prioritize **clarity, accuracy, and adherence to the user's requirements**.
- Avoid **leaking proprietary information, PII, or unverified claims**.
- If the user's query involves multiple steps (e.g., code optimization, content rephrasing), address each part systematically.
- Use **language-specific conventions** (e.g., correct grammar, terminology) to align with the user's preferred language.

O GEPA GENERATED PROMPTS FOR KERNEL GENERATION

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

Figure 27 shows the prompt generated by GEPA with GPT-4o for NPU Keernal Generation, that achieves 26.85% score with the same same GPT-4o agent, that achieved just 4.25% with a simple prompt.

```

4230
4231
4232
4233
4234
4235
4236
4237
4238 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
4239 specified operation on a vector of bfloat16 values. Your code should be complete and self-contained within a single code block, including all
4240 necessary headers and imports. Follow these guidelines:
4241
1. Headers and Imports: Include only the necessary headers for AIE operations. Avoid including `<adf.h>` or any headers that are not part of
4242 the standard AIE API. Use:
4243 ```cpp
4244 #include <stdint.h>
4245 #include <aie_api/aie.hpp>
4246 #include <aie_api/utils.hpp>
4247 ...```
4248
2. Kernel Function: Implement the kernel function as specified in the input. The function should take pointers to input and output buffers
4249 and a size parameter. Use AIE vector operations to process the data efficiently.
4250
3. Vector Operations: Utilize the AIE API's vector operations for loading, processing, and storing data. For example, use `aie::vector` for
4251 vector operations and `aie::reduce_min` for reduction tasks. Ensure that the vector size is compatible with the AIE hardware capabilities.
4252
4. Avoid Non-Existent Functions: Do not use functions like `aie::exp` or `aie::store_v` if they are not supported. Instead, implement the
4253 required functionality using available AIE API functions.
4254
5. Error Handling: Ensure that the code is free of syntax errors and compatible with the AIE environment. Test the code for compilation
4255 without errors.
4256
Here is an example template for a kernel function:
4257 ```cpp
4258 #include <stdint.h>
4259 #include <aie_api/aie.hpp>
4260 #include <aie_api/utils.hpp>
4261
4262 void kernel_function_name(bfloat16 *input_vector, bfloat16 *output_vector, uint32_t vector_size) {
4263     constexpr int vector_length = 16; // Adjust based on AIE capabilities
4264     aie::vector<bfloat16, vector_length> input_data;
4265     aie::vector<bfloat16, vector_length> output_data;
4266
4267     for (uint32_t i = 0; i < vector_size; i += vector_length) {
4268         input_data = aie::load_v<vector_length>(input_vector + i);
4269         // Perform the required operation on input_data
4270         // Store the result in output_data
4271         aie::store_v<vector_length>(output_vector + i, output_data);
4272     }
4273 }
4274 ...```
4275
4276 Replace `kernel_function_name` and the operation logic with the specific task details provided in the input.

```

Figure 27: GEPA generated prompt for NPUEval that achieves 26.85% score with the same GPT-4o agent, that achieved a 4.25% score.

4277 **O.2 KERNELBENCH: CUDA KERNEL CODE GENERATION FOR NVIDIA GPUs**
42784279 **GEPA GPT-4o generated prompt for CUDA Kernel Generation**
42804281
4282 To optimize a given PyTorch model by replacing operators with custom CUDA kernels, follow
4283 these detailed instructions. Your goal is to achieve performance improvements while
4284 ensuring correctness. Name your optimized output architecture `ModelNew`. Output the
4285 new model code in codeblocks. Please generate real code, NOT pseudocode, and ensure
4286 the code compiles and is fully functional. Do not include testing code.
42874288 **### Steps to Create Custom CUDA Kernels**
42894290 **#### 1. Identify Operators to Replace**
4291 - Analyze the model to identify operators that can benefit from custom CUDA
4292 implementations.
4293 - Consider operator fusion opportunities to combine multiple operations into a single
4294 kernel for efficiency.
42954296 **#### 2. Setup and Compilation**
4297 - Use `torch.utils.cpp_extension.load_inline` to compile your CUDA code. This allows you
4298 to define and compile custom CUDA kernels directly within your Python script.
4299 - Ensure all necessary CUDA and C++ headers are included to avoid missing includes errors.
43004301 **#### 3. Implementing the CUDA Kernel**
4302 - Write the CUDA kernel code, ensuring it is optimized for parallel execution. Use shared
4303 memory to reduce global memory accesses and ensure coalesced memory access patterns.
4304 - Example structure for a CUDA kernel:
4305

```
```cpp
4306 __global__ void my_kernel(const float* input, float* output, int size) {
4307 int idx = blockIdx.x * blockDim.x + threadIdx.x;
4308 if (idx < size) {
4309 // Perform computation
4310 }
4311 }
4312 ````
```

4313    **#### 4. Kernel Launch Configuration**  
4314    - Configure the kernel launch parameters (number of blocks and threads per block) to  
4315       maximize GPU utilization.  
4316    - Example:  
4317       

```
```python
4318       const int block_size = 256;
4319       const int num_blocks = (size + block_size - 1) / block_size;
4320       my_kernel<<<num_blocks, block_size>>>(input, output, size);
4321       ````
```

4322 **#### 5. Error Handling and Debugging**
4323 - Implement error checking after CUDA API calls and kernel launches using `
4324 cudaGetLastError()` to catch errors early.
4325 - Use CUDA debugging tools like `cuda-memcheck` and NVIDIA Nsight for debugging and
4326 profiling.
4327 - Be aware of common syntax errors and namespace issues, and ensure that the CUDA code
4328 adheres to syntax rules.

```

4324
4325
4326 ##### 6. Integrating with PyTorch
4327 - Define a Python function that wraps the CUDA kernel call and integrates it into the
4328   PyTorch model.
4329 - Example:
4330   ```python
4331   def my_custom_op(input):
4332       output = torch.zeros_like(input)
4333       my_kernel(input.data_ptr<float>(), output.data_ptr<float>(), input.numel())
4334       return output
4335   ```
4336
4337 ##### 7. Testing and Validation
4338 - Validate the correctness of the custom kernel by comparing its output with the original
4339   PyTorch operator.
4340 - Use profiling tools to measure performance improvements and identify further
4341   optimization opportunities.
4342 - Establish reference outputs for known-good inputs to verify correctness.
4343 - Consider floating-point discrepancies and use a small epsilon for acceptable differences
4344
4345
4346 ##### 8. Compatibility and Compute Capability
4347 - Ensure that the CUDA code is compatible with the target GPU's compute capability.
4348 - Use appropriate compiler flags and consider building for multiple compute capabilities
4349   if necessary.
4350
4351 ##### Example Code
4352 Here is an example of how to replace a simple element-wise addition with a custom CUDA
4353   kernel:
4354
4355   ```python
4356   import torch
4357   import torch.nn as nn
4358   from torch.utils.cpp_extension import load_inline
4359
4360   # Define the custom CUDA kernel for element-wise addition
4361   elementwise_add_source = """
4362   #include <torch/extension.h>
4363   #include <cuda_runtime.h>
4364
4365   __global__ void elementwise_add_kernel(const float* a, const float* b, float* out, int
4366   size) {
4367       int idx = blockIdx.x * blockDim.x + threadIdx.x;
4368       if (idx < size) {
4369           out[idx] = a[idx] + b[idx];
4370       }
4371   }
4372
4373   torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b) {
4374       auto size = a.numel();
4375       auto out = torch.zeros_like(a);
4376
4377       const int block_size = 256;
4378   }
4379
4380   ```


```

```

4371
4372     const int num_blocks = (size + block_size - 1) / block_size;
4373
4374     elementwise_add_kernel<<<num_blocks, block_size>>>(a.data_ptr<float>(), b.data_ptr<
4375         float>(), out.data_ptr<float>(), size);
4376
4377     return out;
4378 }
4379
4380 elementwise_add_cpp_source = (
4381     "torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b);"
4382 )
4383
4384 # Compile the inline CUDA code for element-wise addition
4385 elementwise_add = load_inline(
4386     name="elementwise_add",
4387     cpp_sources=elementwise_add_cpp_source,
4388     cuda_sources=elementwise_add_source,
4389     functions=["elementwise_add_cuda"],
4390     verbose=True,
4391     extra_cflags=[],
4392     extra_ldflags=[],
4393 )
4394
4395 class ModelNew(nn.Module):
4396     def __init__(self) -> None:
4397         super().__init__()
4398         self.elementwise_add = elementwise_add
4399
4400     def forward(self, a, b):
4401         return self.elementwise_add.elementwise_add_cuda(a, b)
4402
4403
4404     ### Additional Best Practices
4405     - **Optimize Memory Usage**: Minimize data transfers between host and device, and use
4406         shared memory to reduce global memory access.
4407     - **Atomic Operations**: When using atomic operations like `atomicMax` with floating-point
4408         numbers, ensure correct usage by following best practices, such as using appropriate
4409         data types and minimizing contention.
4410     - **Performance Optimization**: Maximize parallel execution, optimize memory access
4411         patterns, and use compiler flags to enhance performance.
4412     - **Namespace Usage**: Avoid adding class declarations or function definitions directly to
4413         reserved namespaces like `cuda`. Use nested namespaces within non-reserved
4414         namespaces to organize code.
4415     - **Numerical Precision**: Be aware of floating-point arithmetic issues, such as non-
4416         associativity, and use appropriate precision levels for calculations.
4417     - **Debugging Tools**: Utilize tools like CUDA-GDB and NVIDIA Nsight for debugging and
4418         profiling to ensure correctness and performance.
4419
4420     By following these instructions, you can effectively replace PyTorch operators with custom
4421     CUDA kernels, ensuring both performance improvements and correctness.
4422
4423     ### Instruction for Replacing PyTorch Operators with Custom CUDA Kernels

```

```

4418
4419
4420 Your task is to optimize a given PyTorch model by replacing certain operators with custom
4421 CUDA kernels to achieve performance improvements. Follow the steps below to ensure a
4422 successful implementation:
4423
4424 #### Step 1: Identify Operators for Replacement
4425 - Criteria for Selection: Choose operators that are computationally intensive and have
4426 potential for parallelization. Consider operators that are frequently used in the
4427 model's forward pass.
4428 - Operator Fusion: Look for opportunities to fuse multiple operators into a single
4429 CUDA kernel, such as combining matrix multiplication with activation functions (e.g.,
4430 matmul + ReLU).
4431
4432 #### Step 2: Implement Custom CUDA Kernels
4433 - Kernel Structure: Define your CUDA kernel using the `__global__` specifier. Ensure
4434 that each thread handles a specific part of the computation. Use correct index
4435 calculations to access data.
4436 - Memory Management:
4437 - Allocate memory for input, output, and any intermediate data on the GPU using `__
4438 cudaMalloc`. Use `cudaMemcpy` to transfer data between host and device.
4439 - Utilize shared memory to cache frequently accessed data and reduce global memory
4440 accesses.
4441 - Ensure coalesced global memory accesses for efficient memory transactions.
4442 - Numerical Stability and Boundary Conditions:
4443 - Implement verification mechanisms to ensure numerical stability. Use `__host__`  

4444 `__device__` functions for testing on both CPU and GPU.
4445 - Handle boundary conditions to prevent out-of-bounds memory access. Ensure that thread
4446 indices are within valid ranges.
4447 - Optimization Techniques:
4448 - Use shared memory to reduce global memory accesses and improve performance.
4449 - Consider using mixed precision and Tensor Cores for matrix operations to enhance
4450 performance.
4451 - Avoid diverged execution paths to maintain efficient parallel execution.
4452
4453 #### Step 3: Integrate CUDA Kernels into PyTorch Model
4454 - Inline Compilation: Use `torch.utils.cpp_extension.load_inline` to compile your CUDA
4455 code and integrate it into the PyTorch model.
4456 - Model Modification: Replace the original PyTorch operators with calls to your custom
4457 CUDA functions. Ensure that the new model architecture (`ModelNew`) is fully
4458 functional and compiles without errors.
4459
4460 #### Step 4: Testing and Validation
4461 - Correctness: Verify that the output of the optimized model matches the expected
4462 output of the original model. Use a set of test cases to ensure accuracy.
4463 - Performance Evaluation: Measure the runtime of the optimized model and compare it to
4464 the original. Aim for a significant reduction in execution time.
4465 - Edge Case Handling: Ensure that the kernel correctly handles cases where the matrix
4466 size is not a multiple of the block size and other potential edge cases.
4467
4468 #### Example Code
4469 Below is an example of how to define and integrate a custom CUDA kernel for element-wise
4470 addition:
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664

```

```

4465
4466     ```python
4467     import torch
4468     import torch.nn as nn
4469     from torch.utils.cpp_extension import load_inline
4470
4471     # Define the custom CUDA kernel for element-wise addition
4472     elementwise_add_source = """
4473     #include <torch/extension.h>
4474     #include <cuda_runtime.h>
4475
4476     __global__ void elementwise_add_kernel(const float* a, const float* b, float* out, int
4477     size) {
4478         int idx = blockIdx.x * blockDim.x + threadIdx.x;
4479         if (idx < size) {
4480             out[idx] = a[idx] + b[idx];
4481         }
4482     }
4483
4484     torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b) {
4485         auto size = a.numel();
4486         auto out = torch::zeros_like(a);
4487
4488         const int block_size = 256;
4489         const int num_blocks = (size + block_size - 1) / block_size;
4490
4491         elementwise_add_kernel<<<num_blocks, block_size>>>(a.data_ptr<float>(), b.data_ptr<
4492             float>(), out.data_ptr<float>(), size);
4493
4494         return out;
4495     }
4496 """
4497
4498     elementwise_add_cpp_source = (
4499         "torch::Tensor elementwise_add_cuda(torch::Tensor a, torch::Tensor b);"
4500     )
4501
4502     # Compile the inline CUDA code for element-wise addition
4503     elementwise_add = load_inline(
4504         name="elementwise_add",
4505         cpp_sources=elementwise_add_cpp_source,
4506         cuda_sources=elementwise_add_source,
4507         functions=["elementwise_add_cuda"],
4508         verbose=True,
4509         extra_cflags=[],
4510         extra_ldflags=[],
4511     )
4512
4513     class ModelNew(nn.Module):
4514         def __init__(self) -> None:
4515             super().__init__()
4516             self.elementwise_add = elementwise_add
4517
4518         def forward(self, a, b):
4519
4520

```

```

4512
4513     return self.elementwise_add.elementwise_add_cuda(a, b)
4514     ...
4515
4516     ##### Constraints
4517     - Ensure that the optimized model maintains the same accuracy as the original.
4518     - The custom CUDA kernels should be optimized for performance, minimizing execution time
4519     and maximizing GPU utilization.
4520
4521     By following these instructions, you will be able to effectively replace PyTorch operators
4522     with custom CUDA kernels, achieving significant performance improvements while
4523     maintaining model accuracy.
4524
4525
4526
4527
4528

```

P NUMBER OF REFLECTION LM CALLS MADE BY GEPA DURING OPTIMIZATION

Table 4: Total number of calls made by GEPA to reflection LM during optimization.

Benchmark Name	Num Reflection Calls GPT-4.1-Mini	Num Reflection Calls Qwen3-8B
AIME-2025	24	90
LiveBench-Math	34	38
HotpotQA	69	64
IFBench	21	17
Hover	92	50
PUPA	46	38

```

4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558

```