APEX-DATA: AUTOMATED PROMPT ENGINEERING EXPERT WITH DYNAMIC DATA SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models are highly sensitive to prompt formulation, necessitating automatic prompt optimization to unlock their full potential. While evolutionary algorithms have emerged as the dominant paradigm, they suffer from a critical bottleneck: data efficiency. Current methods treat the development dataset as a static benchmark, wasting significant compute budget on uninformative data. In this work, we introduce **APEX-DATA** (**A**utomated **P**rompt **E**ngineering e**X**pert with Dynamic **Data** Selection), a novel framework that optimizes the *data usage* alongside the prompt search. APEX-DATA dynamically stratifies the dataset into `Easy`, `Hard`, and `Mixed` tiers based on the optimization lineage. By prioritizing the `Mixed` tier, which identifies the data where the LLM has mixed performance, we identify two high-leverage subsets: the **addressable frontier** for generating informative mutations and the **rank-sensitive frontier** for distinguishing candidate quality. We evaluate APEX-DATA across three diverse benchmarks: IFBench, SimpleQA Verified, and FACTS Grounding, spanning across distinct task types and evaluation protocols, under a fixed budget of task LLM calls. Due to its data efficiency, APEX-DATA outperforms the initial prompt by an average of 11.2 percentage points on Gemini 2.5 Flash and 6.8 percentage points on Gemma 3 27B, demonstrating that a data-centric approach is key to efficient prompt optimization.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities across instruction following, fact retrieval, and complex reasoning. However, their performance remains notoriously sensitive to the prompts (Kojima et al., 2022). A minor change can often mean the difference between a correct answer and a hallucination. This fragility has given rise to automatic prompt optimization, a field dedicated to treating prompts not as static strings, but as learnable parameters that can be optimized (Zhou et al., 2023b; Deng et al., 2022; Hsieh et al., 2024). Currently, genetic and evolutionary algorithms have emerged as the dominant paradigm in prompt optimization, exemplified by methods like APO (Pryzant et al., 2023) and GEPA (Agrawal et al., 2026). These frameworks typically operate through an iterative two-stage process: (1) *Mutation*, where the optimizer analyzes error cases from the current prompt to generate new candidate prompts, and (2) *Selection*, where these candidates are evaluated and ranked by their performance to identify the better ones.

However, this paradigm faces a critical and often overlooked bottleneck: **data efficiency**. The effectiveness of both stages is heavily dependent on the quality of the data used. The mutation stage relies on identifying *informative* failure cases that point towards fixable errors, while the selection stage requires a rigorous evaluation to accurately rank candidates. Current methods typically resort to naive data strategies: random sampling for mutation, and either random sampling or full-set evaluation for selection. This creates a fundamental dilemma: random sampling leads to rank instability, where superior prompts are discarded due to variance, while full-set evaluation depletes the compute budget rapidly, limiting the search to only a few generations.

We argue that the root cause of this inefficiency is the treatment of the dataset as a *static benchmark*. In reality, the utility of any given data point is not fixed. It changes dynamically throughout the optimization process. As prompts evolve and improve, examples that were once informative discriminators may converge into consistently `Easy` cases (solved by all candidates), while others persist as `Hard` (intractable noise). Continuing to mutate based on unsolvable errors, or evaluating

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

candidates on examples that have lost their discriminatory power, wastes a significant portion of the compute budget on signals that provide zero improvement information.

To address this, we introduce APEX-DATA (**A**utomated **P**rompt **E**ngineering e**X**pert with Dynamic **Data** Selection). APEX shifts the focus from improving the search algorithm to improving the data usage. By maintaining a lineage of prompt performance, APEX-DATA dynamically categorizes data points into `Easy`, `Hard`, and `Mixed` tiers. Our core insight is that the informative mutation and selection signals primarily lie in the `Mixed` tier, where the LLM has mixed performance on recent prompt variants. This dynamic tier constitutes two critical frontiers: the **addressable frontier** indicating volatile errors suitable for mutation, and the **rank-sensitive frontier** with ambiguous cases that differentiate candidates. By prioritizing compute to these high-leverage frontiers, APEX-DATA maximizes the optimization signal per evaluation call.

In summary, our contributions are three-fold. First, we identify the data efficiency bottleneck in current genetic prompt optimization methods and attribute it to the static treatment of dynamic data signals. Second, we propose APEX-DATA, a novel algorithm that enhances the prompt optimization process by stratifying data into dynamic tiers and targeting the addressable and rank-sensitive frontiers. Third, we achieve performance gains across three diverse benchmarks spanning across distinct task types and evaluation protocols (IFBench, SimpleQA Verified, FACTS Grounding). APEX-DATA improves 11.2 percentage points on Gemini 2.5 Flash and 6.8 percentage points on Gemma 3 27B with budget constraints, establishing a data-efficient prompt optimization paradigm.

2 BOTTLENECK OF PROMPT OPTIMIZATION

2.1 PRELIMINARY: PROMPT OPTIMIZATION

Prompts serve as the primary interface for interacting with LLMs at the inference stage. For a given user intent, there exists an expansive space of potential prompt variants, many of which yield significantly different performance outcomes. The goal of prompt optimization is to systematically identify an optimal prompt P^* that maximizes a predefined performance metric on a target LLM:

$$P^* = \arg\max_P \mathbb{E}_{x\sim\mathcal{D}^*}[f(x, \text{LLM}(P, x))], \tag{1}$$

where $\mathcal{D}^*$ represents the oracle data distribution and $f(\cdot)$ is an evaluation function (e.g., string-matching accuracy or model-based grading) that scores the LLM's response relative to input x. The evaluation function $f(\cdot)$ in prompt optimization can take arbitrary forms. For optimization purpose, the evaluation outcome is typically converted into $s(P, x) \in \{0, 1\}$, where $s(P, x) = 0$ marks a failure to be selected as an error case and $s(P, x) = 1$ denotes a pass. The specific threshold or condition defining a failure is flexible across tasks and evaluation protocols. In practice, both the LLM and the evaluation function f are treated as black boxes where gradients are unavailable. Consequently, derivative-free genetic algorithms, such as APO and GEPA, have emerged as leading methods. These methods typically iterate through two primary stages on a collected dataset $\mathcal{D}$:

Mutation: To explore the prompt space, a meta-optimizer LLM_{meta} (typically the target LLM itself) generates a new prompt P_{new} conditioned on an existing prompt P_t and a batch of its error cases $\mathcal{E}_t \subset \{x \in \mathcal{D} \mid s(P_t, x) = 0\}$:

$$P_{\text{new}} \sim \text{LLM}_{\text{meta}}(P_t, \mathcal{E}_t). \tag{2}$$

Selection: The new candidate prompt P_{new} is evaluated alongside the current population $\mathcal{P}_t$. The subsequent generation $\mathcal{P}_{t+1}$ is constructed by selecting the k highest-performing prompts based on their aggregate scores on $\mathcal{D}$:

$$\mathcal{P}_{t+1} = \text{Top-}k\left(\mathcal{P}_t \cup \{P_{\text{new}}\} \mid \mathcal{D}\right). \tag{3}$$

2.2 PROBLEM: DATA EFFICIENCY

Despite their success, data efficiency remains a critical but overlooked bottleneck in both the mutation and selection stages, hindering the scalability of existing frameworks. We formalize the global optimization state as a **sparse score matrix** $\mathbf{S} \in \{0, 1, \emptyset\}^{|\mathcal{P}|\times|\mathcal{D}|}$, where each entry $S_{j,i}$ represents the performance of prompt P_j on datapoint $x_i \in \mathcal{D}$:

$$S_{j,i} = s(P_j, x_i) \in \{0, 1\}, \quad (\text{or } \emptyset \text{ if not applicable}). \tag{4}$$

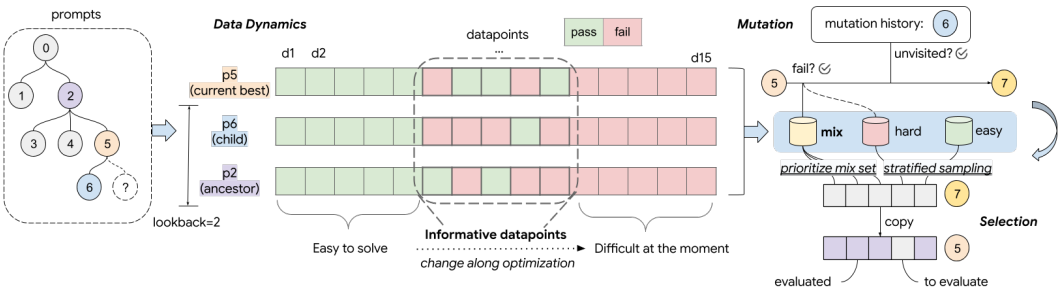


Figure 1: **The APEX-DATA framework overview.** The optimization process begins by tracking prompt lineage (left), where historical performance across ancestors (e.g., p_2) and children (e.g., p_6) is analyzed over a defined lookback window. In the **Data Dynamics** phase (center), datapoints are dynamically categorized into tiers (`Easy`, `Hard`, and `Mixed`) based on pass/fail trajectories. The `Mixed` set identifies informative datapoints where prompts have divergent performance, signaling high potential for optimization. These tiers inform the **Mutation** and **Selection** stages (right), both of which prioritizing the `Mixed` set: the former leverages mutation history to explore unvisited prompt space, while the latter employs stratified sampling to efficiently rank candidate prompts. Evaluation is incremental, skipping previously evaluated datapoints to maximize efficiency.

Inefficiency in Mutation. The mutation distribution produced by LLM_{meta} highly depends on $\mathcal{E}_{\text{curr}}$. Current methods typically rely on random sampling of error cases, which fails to account for the **optimization trajectory**, rendering the mutation process a source of instability rather than monotonic improvement. Optimization performance is often governed by a hierarchy of fixability, where certain errors serve as logical precursors to others. For instance, if error samples' batch $\mathcal{E}_b$ contains errors only addressable after foundational issues in $\mathcal{E}_a$ are resolved, the trajectory must follow the sequence $\mathcal{E}_a \rightarrow \mathcal{E}_b$. Reversing this sequence or sampling randomly leads to high-variance updates and forgetting, as the optimizer attempts to solve high-level failures without the necessary preliminary refinements. This lack of trajectory awareness also frequently precipitates the trap where $\mathcal{E}_{\text{curr}}$ becomes dominated by extremely hard cases that are fundamentally beyond the capacity of the target LLM. Attempting to optimize for these points misleads the optimizer, stalling progress. Notably, the optimal data for mutation is non-static. It changes dynamically along the optimization process as different error types become addressable at different stages of prompt maturity.

Inefficiency in Selection. The selection stage represents the primary computational overhead of the optimization loop, often accounting for more than 90% of total costs.[1] This creates a fundamental trade-off between selection accuracy and the number of iterations T possible within a fixed budget. While comprehensive evaluation ensures precise ranking, it is computationally prohibitive. Conversely, stochastically sampling a subset introduces noise that leads to rank inversion, mistakenly selecting suboptimal prompts. Crucially, absolute performance estimates are often redundant. For selection, the priority is the relative rank of candidates. Because the population consists of related prompts that share significant structural or semantic similarities, their performance is identical for a large portion of $x \in \mathcal{D}$. The true bottleneck is the failure to isolate **discriminative data**:

$$\mathcal{D}_{\text{disc}} = \{x_i \in \mathcal{D} \mid \exists P_a, P_b \in \mathcal{P}_t : s(P_a, x_i) \neq s(P_b, x_i)\}. \tag{5}$$

Existing frameworks waste their budget evaluating $x \notin \mathcal{D}_{\text{disc}}$, where all candidates perform identically. Evaluating these points provides zero information for selection, effectively reducing the number of meaningful updates possible within a fixed compute budget.

3 METHOD: APEX-DATA

The APEX-DATA framework (Figure 1) addresses the core bottlenecks of data inefficiency by dynamically re-evaluating the utility of each datapoint in the optimization loop (Algorithm 1). Instead of treating the dataset as a static benchmark, APEX-DATA categorizes data based on its historical interaction with the prompt lineage to guide both mutation and selection.

[1]Mutation often requires fewer than three API calls, whereas selection requires evaluating across the entire dataset once or multiple subsets, frequently requiring hundreds of API calls, as reported in APO and GEPA.

Algorithm 1: APEX-DATA Workflow

Input : Initial prompt P_0, dataset $\mathcal{D}$, budget N, iterations T
Output : Optimized prompt P^*

$P_{\text{curr}} \leftarrow P_0$; $\alpha \leftarrow \alpha_0$; $\mathcal{U} \leftarrow \emptyset$; $\mathcal{H} \leftarrow \text{Evaluate}(P_0, \mathcal{D})$ `// Initialize state & cache`
for $t = 1 \dots T$ **do**
  `// 1. Dynamic Data Stratification`
  $\mathcal{B} \leftarrow \text{Stratify}(\mathcal{D}, \mathcal{H}, P_{\text{curr}}, k)$ `// Partition into subsets` $\mathcal{B}_{T,s}$ `(Alg. 3)`
  `// 2. Trajectory-Guided Mutation`
  $\mathcal{E} \leftarrow$ Sample m instances from $(\mathcal{B}_{M,0} \cup \mathcal{B}_{H,0}) \setminus \mathcal{U}$; $\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{E}$ `// Eq. (8)`
  $P_{\text{new}} \leftarrow \text{LLM}_{\text{meta}}\big(\text{Mutate}(P_{\text{curr}}, \text{Critique}(P_{\text{curr}}, \mathcal{E}))\big)$ `// Diagnose & propose candidate`
  `// 3. Rank-Sensitive Selection`
  $\mathcal{D}_{\text{eval}} \leftarrow \text{SampleRankSensitive}(\mathcal{B}, N, \alpha)$ `// Budget split via Eqs.(10)`
  if P_{new} *outperforms* P_{curr} *on* $\mathcal{D}_{eval}$ **then**
    $P_{\text{curr}} \leftarrow P_{\text{new}}$; $\alpha \leftarrow \alpha + \beta$ `// Accept candidate & anneal anchor`
  $\mathcal{H} \leftarrow \text{UpdateHistory}(\mathcal{H}, \mathcal{D}_{\text{eval}})$ `// Incremental evaluation cache`
return P_{curr}

3.1 DATA DYNAMICS FOR PROMPT OPTIMIZATION

The core philosophy of APEX-DATA is to move beyond static data by characterizing how individual datapoints influence the optimization trajectory. We categorize the interaction between the prompt lineage and the dataset into three distinct dynamics: `Easy`, `Hard`, and `Mixed` (Algorithm 3).

To accurately capture these dynamics without overfitting to stale data, we treat the evaluation history $\mathcal{H}$ as an ordered sequence of prompts. For any datapoint $x_i \in \mathcal{D}$, let $\mathcal{H}^{(i)}_{\text{valid}}$ be the sub-sequence of historical prompts that have actively evaluated x_i (i.e., $s(P, x_i) \neq \emptyset$). We define its local history $\mathcal{R}_i$ as the set of outcomes from the k most recent prompts in $\mathcal{H}^{(i)}_{\text{valid}}$:

$$\mathcal{R}_i = \left\{ s(P, x_i) \mid P \in \text{last}_k\big(\mathcal{H}^{(i)}_{\text{valid}}\big) \right\}. \tag{6}$$

The lookback window k acts as a dynamic slice on the sparse score matrix $\mathbf{S}$, ensuring the optimizer always has a consistent sample size of relevant, recent behavior rather than outdated signals.

We partition the dataset $\mathcal{D}$ into three semantic tiers based on the variance of $\mathcal{R}_i$:

$$\text{Tier}(i) = \begin{cases} \text{E (Easy)} & \text{if Set}(\mathcal{R}_i) \equiv \{1\} \\ \text{H (Hard)} & \text{if Set}(\mathcal{R}_i) \equiv \{0\} \\ \text{M (Mixed)} & \text{if Set}(\mathcal{R}_i) \equiv \{0, 1\} \end{cases} \tag{7}$$

- **Easy:** Points consistently solved by the lineage. Re-evaluating them provides minimal signal.
- **Hard:** Points consistently failed. These represent data that are currently intractable.
- **Mixed:** Points exhibiting volatility. These represent the rank-sensitive frontier for evaluation and the most probable targets for improvement.

By intersecting these historical tiers $T \in \{E, H, M\}$ with the evaluation outcome $s \in \{1, 0, \emptyset\}$ under the *current* prompt P_{curr}, we partition the dataset $\mathcal{D}$ into nine disjoint subsets, denoted as $\mathcal{B}_{T,s}$. This notation provides a granular view of the data state. For example, $\mathcal{B}_{M,0} \subseteq \mathcal{D}$ represents the subset of historically Mixed instances that are currently failing, while $\mathcal{B}_{E,\emptyset}$ represents Easy instances that were skipped in the current pass.

3.2 TRAJECTORY-GUIDED MUTATION

To resolve the stochastic instability of mutation, APEX-DATA enforces a trajectory-guided strategy. Standard optimization methods uniformly sample errors from the entire failure set, frequently retrieving "hard" cases ($\mathcal{B}_{H,0}$) that the current prompt logic cannot handle, leading to hallucinated corrections. In contrast, APEX-DATA targets the **addressable frontier** by constructing the mutation

error batch primarily from the `Mixed-Fail` bucket ($\mathcal{B}_{M,0}$). These "soft failures" are datapoints where the model has demonstrated capacity in the recent lineage but regressed under the current variation. Correcting these regressions stabilizes the trajectory.

Furthermore, to ensure coverage maximization, we maintain a usage history $\mathcal{U}$ to prevent the optimizer from repeatedly overfitting to the same recurring failures. Errors for mutation are exclusively drawn from unvisited failures, where the usage history will be reset once this pool is exhausted, guaranteeing broad exploration of the error surface:

$$e \in \{x_i \mid x_i \in (\mathcal{B}_{M,0} \cup \mathcal{B}_{H,0}),\, x_i \notin \mathcal{U}\}\,. \tag{8}$$

3.3 Rank-Sensitive Selection

The selection stage reduces evaluation cost by focusing on the **rank-sensitive subset**, which contains datapoints where the new prompt is most likely to diverge from its parent. We construct evaluation dataset $\mathcal{D}_{\text{eval}}$ using a tiered sampling strategy to fill the budget N. First, we identify volatile unknowns, data points that are historically mixed but were skipped during the previous evaluation. These provide the highest information gain and form the required baseline: $\mathcal{D}_{\text{req}} = \mathcal{B}_{M,\emptyset}$. To fill the remaining evaluation budget $R = N - |\mathcal{D}_{\text{req}}|$, we employ stratified sampling to balance stability (anchors) and error correction. We sample a positive set $\mathcal{D}_{\text{pos}}$ (prioritizing $\mathcal{B}_{M,1}$ to catch regressions, followed by $\mathcal{B}_{E,\emptyset}$) and a negative set $\mathcal{D}_{\text{neg}}$ (prioritizing $\mathcal{B}_{M,0}$ to confirm fixes).

The size of these sets is governed by an **anchor ratio** α_t, which dictates the proportion of the budget dedicated to verifying positive stability. This ratio follows an annealing schedule, growing when the updated prompt is selected:

$$\alpha_{t+1} = \alpha_t + \beta \cdot \mathbb{I}(P_{\text{new}} \succ P_{\text{curr}}). \tag{9}$$

As the prompt improves, α increases, effectively preserving mastered logic by dedicating more evaluation budget to preventing regressions. To ensure the anchor size is calibrated to the model's actual competence, we clamp α_t against the pass ratios of the mixed tier (ρ_{mix}) and all data (ρ_{all}):

$$k_{\text{pos}} = \lfloor \min(\alpha_t, \rho_{\text{mix}}, \rho_{\text{all}}) \cdot R \rfloor, \quad k_{\text{neg}} = R - k_{\text{pos}}. \tag{10}$$

Finally, to minimize computational overhead, APEX-DATA employs an **incremental evaluation** scheme. If an outcome $s(P_{\text{curr}}, x_i)$ for a sampled point already exists in the history $\mathcal{H}$, we retrieve it directly from memory, executing full inference only for the new candidate.

4 Experiments

4.1 Experimental Setting

General Setup. We employ **Gemini-2.5-Flash** (Comanici et al., 2025) and **Gemma-3-27B** (Team Gemma et al., 2025) as our underlying LLMs. For each experiment, we utilize the same model for both the meta-optimizer (LLM_{meta}) and the target model (LLM). To enforce a fair comparison across methods with varying sample efficiency, we restrict all optimizers to a fixed global budget of 5,000 calls to the evaluation function.

Datasets. We evaluate APEX-DATA on three distinct tasks representing instruction following, factual question answering, and grounded generation. They span across distinct evaluation protocols, including programmatic verifier, gold-standard answer, and LLM-as-a-judge. **IFBench** (Pyatkin et al., 2025): Assesses precise instruction following on verifiable constraints, using programmatic verifiers to evaluate constraint adherence. Following Agrawal et al. (2026), we sample 500 instances from the IF-RLVR split for optimization and reserve 294 examples from IFBench for testing. Notably, the constraints for optimization and testing have no overlap. **SimpleQA Verified** (Haas et al., 2025): Assesses short-form factuality question answering across diverse topics, evaluating against the gold-standard answer. We randomly partition the data into 700 examples for optimization and 300 for testing. **FACTS Grounding** (Jacovi et al., 2025): Assesses long-form generation grounded to a provided document, evaluating response quality and grounding faithfulness using LLM-as-a-judge. We randomly partition the data into 660 examples for optimization and 200 for testing.

Baselines. We compare against the following leading prompt optimization frameworks. **APO** (Pryzant et al., 2023): A genetic based prompt optimization method that remains the best-performing

Table 1: **Main Results.** Test accuracy (%) ± Std Dev. **Bold** and underline indicates the best and the second-best. APEX-DATA consistently outperforms baselines across models and benchmarks.

Method	IFBench	SimpleQA Verified	FACTS Grounding	Average
Gemini 2.5 Flash				
Initial Prompt	38.5 ± 1.2	23.6 ± 0.6	85.8 ± 1.1	49.3
APO ($\|D\|$, Pryzant et al. 2023)	43.7 ± 1.2	27.4 ± 2.6	89.7 ± 1.9	53.6
APO ($\|D\|/2$, Pryzant et al. 2023)	43.5 ± 1.9	25.0 ± 3.5	89.9 ± 0.1	52.8
MIPROv2 (Opsahl-Ong et al., 2024)	41.6 ± 3.7	24.3 ± 0.8	93.2 ± 3.0	53.0
TextGrad (Yuksekgonul et al., 2025)	40.6 ± 1.2	28.9 ± 2.0	93.5 ± 1.3	54.3
GEPA (Agrawal et al., 2026)	41.2 ± 0.9	28.8 ± 0.8	93.5 ± 0.0	54.5
APEX-DATA (Ours)	**52.3** ± 1.4	**35.0** ± 2.3	**94.1** ± 0.4	**60.5**
Gemma 3 27B				
Initial Prompt	33.4 ± 0.7	9.4 ± 0.5	80.7 ± 1.6	41.2
APO ($\|D\|$, Pryzant et al. 2023)	35.7 ± 2.4	**11.5** ± 0.1	88.5 ± 0.7	45.2
APO ($\|D\|/2$, Pryzant et al. 2023)	36.1 ± 0.5	8.4 ± 1.4	85.7 ± 0.9	43.4
GEPA (Agrawal et al., 2026)	34.1 ± 0.5	9.4 ± 0.5	91.7 ± 0.5	45.1
APEX-DATA (Ours)	**39.3** ± 2.0	**11.5** ± 1.2	**93.3** ± 0.1	**48.0**

after its release (Wan et al., 2024). It utilizes reflective mutation and employs a sampling-based selection mechanism with a controllable budget per iteration. We include two versions with per-iteration budgets set to the dataset size ($|D|$) and half ($|D|/2$). **MIPROv2** (Opsahl-Ong et al., 2024): A prompt optimization algorithm within DSPy (Khattab et al., 2024) using Bayesian optimization over subsampled minibatches to navigate the search space. **TextGrad** (Yuksekgonul et al., 2025): An automatic text-driven differentiation framework, supporting backpropagate natural language feedback using LLM-driven gradient descent. **GEPA** (Agrawal et al., 2026): A genetic framework employing reflection-based mutation and a Pareto-based selection strategy on the full dataset.

Implementation Details. For APEX-DATA, we employ a lineage lookback window of $k = 5$ to construct the history for data categorization. The mutation batch size is set to $m = 5$. For the rank-sensitive evaluation, we set the evaluation budget per iteration to $N = 100$. The anchor ratio initializes at $\alpha = 0.2$ and increments by $\beta = 0.03$ when a better prompt is found, gradually stabilizing the evaluation set as the prompt converges.

4.2 MAIN RESULTS

Table 1 presents the comparative performance of APEX-DATA against baseline methods. APEX-DATA consistently outperforms baselines across both LLMs and all three benchmarks, achieving the highest average accuracy. On Gemini 2.5 Flash, the method secures a mean accuracy of 60.5% (+11.2% over the initial prompt), obviously surpassing the strongest baseline, GEPA, which reaches 54.5%. This observation is consistent on Gemma 3 27B, where APEX-DATA maintains its lead. Notably, this performance gap is particularly pronounced on complex instruction-following tasks and when optimizing stronger LLMs, highlighting the method's scalability. Broadly, these results validate that the bottleneck in prompt optimization is not merely the search algorithm, but the *effective optimization signals*. While baselines struggle with the trade-off between signal cost and signal noise, APEX-DATA demonstrates that actively shaping the data distribution allows for directed and stable improvement.

A deeper analysis of the experimental results isolates two distinct failure modes in existing prompt selection strategies. First, methods reliant on full-set evaluation (GEPA) suffer from *limited search depth*. While the evaluation signal is reliable, the prohibitive computational cost restricts the depth of optimization trajectories. Second, random subsampling strategies (APO) lack *evaluation stability*. The reliance on small random sets introduces high variance, failing to reliably distinguish between superior and inferior prompts. This instability is evident in that aggressive subsampling with fewer evaluation samples to exchange for more iterations (APO $|D|/2$) may lead to performance regression

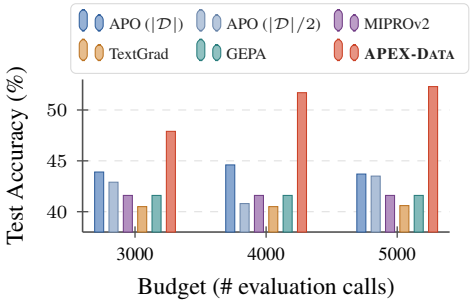


Figure 2: **Performance by budget**.

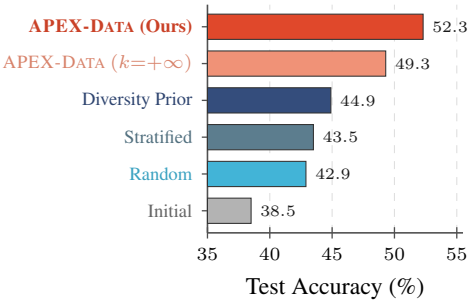


Figure 3: **Ablation of data selection strategies**.

compared to the conservative baseline (APO $|D|$). Furthermore, these evaluation issues exacerbate the inherent limitations of random mutation. GEPA truncates the search before the process can stabilize, while APO exhibits performance regression by corrupting the selection signal with noise. Consequently, baselines fail to achieve *stable and monotonic improvement*. APEX-DATA effectively navigates these trade-offs through its dynamic data selection strategy. By prioritizing the *Addressable Frontier* for mutation and the *Rank-Sensitive Frontier* for selection, the method achieves better performance across multiple scenarios.

4.3 ANALYSIS

To provide an in-depth understanding of the prompt optimization methods, we conduct fine-grained analysis on IFBench with Gemini 2.5 Flash.

APEX-DATA Sustains Improvement Across Budgets. Figure 2 illustrates the test accuracy achieved across different evaluation budgets. Baselines such as GEPA exhibits improvement in early stages but makes little progress thereafter, a result of *insufficient exploration* constrained by high per-iteration costs. Conversely, the aggressive subsampling of APO ($|\mathcal{D}|/2$) results in *instability*. In contrast, APEX-DATA maintains a *sustained improvement* that becomes increasingly pronounced as the budget scales, validating the critical importance of *data efficiency*.

Dynamic Data Selection Outperforms Other Strategies. Figure 3 dissects the contribution of different data selection mechanisms to the final performance. The results highlight a clear hierarchy of efficacy. *Random* provides a baseline improvement over the initial prompt. *Stratified* balances the dataset based on initial prompt performance, marginally improving upon random selection. Explicitly enforcing *Diversity Prior* based on constraint types provides further gains, as it introduces auxiliary guidance information. However, these methods still fall significantly short of APEX-DATA variants. The comparison between APEX-DATA *($k = +\infty$)* and the complete APEX-DATA validates the hypothesis of our dynamic frontier strategy, demonstrating that removing *outdated signals* to target the dynamic frontiers delivers a substantial performance margin.

Prioritizing Mixed Data Drives Dominant Optimization Gains. Figure 4 highlights the critical role of data tiers during optimization. Prioritizing `Hard` instances severely degrades performance by overfitting to intractable noise, while focusing on `Easy` examples yields negligible learning signal. While uniform random sampling across `All` data produces a modest gain, concentrating optimization on `Mixed` instances achieves a substantial improvement. This confirms that the volatile frontier provides the vital grounding signal for generalizable prompt improvement.

Mixed Data Preserves Candidate Ranking Stability. Iterative prompt optimization fundamentally relies on accurate relative prompt comparison. When evaluating on a small subset of data, noisy estimates can invert this relative ordering. As shown in Figure 5, APEX-DATA's strategy of prioritizing `Mixed`-tier instances substantially improves pairwise ranking agreement against the full-set ground truth compared to the uniform random sampling baseline. By prioritizing `Mixed` data, APEX-DATA concentrates budget on instances where competing prompts actually diverge, ensuring reliable candidate selection at a fraction of the evaluation cost.

Informative Mixed Instances Emerge Dynamically Throughout Optimization. Figure 6 tracks the composition of data tiers across optimization iterations. As optimization progresses, a significant fraction of instances transitions into the `Mixed` tier, stabilizing around 20–25% of the dataset. This dynamic emergence confirms that data utility is not static. Instances that appear intractable initially

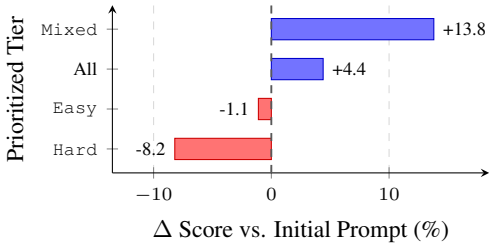


Figure 4: Performance gain over the initial prompt when prioritizing different data tiers during optimization.

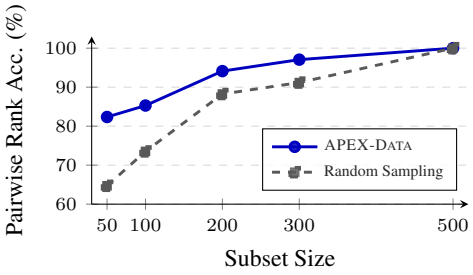


Figure 5: Pairwise candidate prompt selection accuracy measured against full-set ground truth ($N = 500$) across subset sizes.

Table 2: Independent contributions of APEX-DATA's components.

Trajectory-Guided Mutation	Rank-Sensitive Selection	Score (%)
✓	✓	**52.3**
✗	✓	50.2
✓	✗	48.3
✗	✗	42.9

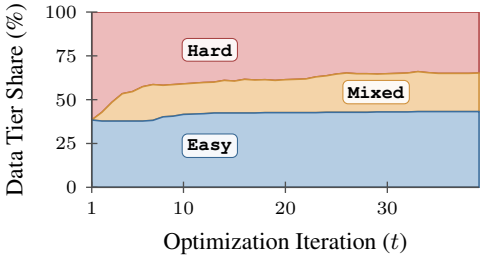


Figure 6: Evolution of data tier composition. `Mixed` instances emerge dynamically.

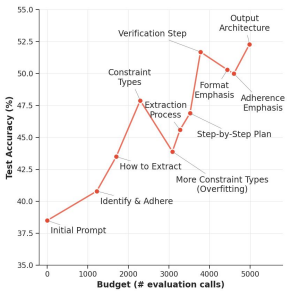
become addressable as prompt quality matures. By continuously tracking this shifting frontier over a moving lookback window, APEX-DATA maintains a fresh pool of informative candidates for mutation and selection rather than stalling on static partitions.

APEX-DATA's Mutation and Selection Components Yields Synergistic Gains. We conduct an ablation study to isolate and quantify the individual contributions of APEX-DATA's prompt mutation and selection mechanisms. As detailed in Table 2, replacing both components with random baselines results in a severe performance drop. Introducing APEX-DATA's selection strategy independently yields a greater improvement than applying the mutation strategy alone. Crucially, the combination of both components achieves the peak overall score, highlighting a strong synergistic effect between trajectory-guided mutation and rank-sensitive selection.

Qualitative Example: APEX-DATA Accumulates Improvements Over Iterations. The evolution of the APEX-DATA prompt in Figure 7 demonstrates a clear shift from simple instruction-giving to establishing a rigid meta-cognitive scaffolding for the model. Early iterations focus on *input analysis*, explicitly teaching the model what a constraint is. The middle phase introduces *process control*, forcing a plan and verify mechanism that separates the reasoning process from the final generation. The final phase creates a strict *output architecture*. By mandating a specific response structure (*Constraints* → *Plan* → *Execution* → *Answer*), the prompt acts as a forcing function.

5 RELATED WORK

Automatic Prompt Optimization. Early research explored diverse search strategies for prompt engineering, ranging from gradient-based methods (Shin et al., 2020; Deng et al., 2022; Chen et al., 2024a) and edit-based heuristics (Prasad et al., 2023), to instruction induction (Honovich et al., 2023). They have paved the way for LLM-driven optimization frameworks, which formalize prompt optimization by leveraging LLMs to generate candidate instructions (Zhou et al., 2023b; Yang et al., 2024). As the field matured, genetic (or evolutionary) algorithms utilizing natural language feedback emerged as the most effective paradigm. A representative method is APO (Pryzant et al., 2023), which iteratively refines prompts by treating error feedback as a signal for mutation. Parallel work has explored other directions to improve the genetic approach, by integrating history guidance, hybrid search strategies, and carefully curated meta-prompts (Hsieh et al., 2024; Fernando et al., 2024; Wang et al., 2024b; Ye et al., 2024; Guo et al., 2024). Notably, AELP (Hsieh et al., 2024) is designed



Carefully analyze the given query. First, meticulously identify and list[1222] *every single instruction* within the 'Query' that imposes a rule or requirement on the final output's form, style, length, specific wording (e.g., required/forbidden words/phrases, unique words), character/word counts, punctuation, case, structure, or content (e.g., starting/ending text, inclusion/exclusion of specific characters/numbers, postscripts, or any directive about copying/ignoring parts of the input).[2292/3085] To do this, systematically scan the 'Query' for keywords and phrases related to form, style, length, specific wording (e.g., 'must include', 'cannot contain', 'only use'), counts (e.g., 'at least', 'exactly', 'number of'), punctuation, case, structure, and content directives. For each identified rule, rephrase it clearly as a distinct constraint. List these as distinct constraints.[3272] If no such constraints are found, state 'No specific constraints identified.'[1222] Before generating the response, create a detailed, step-by-step plan to ensure every identified constraint is met. Include checks for each constraint within your plan. Then, execute this plan to construct your response.[3519] During this execution, and especially when formulating the final content to be enclosed in <answer> tags, *strictly apply all identified constraints as absolute rules guiding the generation process*.[4599] Before outputting the final response, perform a comprehensive check to verify all identified output format, style, length, and content constraints have been met.[3777] The content of your final response should strictly follow this structure: First, output 'Constraints:', followed by your identified constraints. Next, output 'Plan:', followed by your detailed plan. Then, output 'Execution:', followed by your execution details. Finally, and only then, output the fully compliant response to the 'Query' *exclusively* within `<answer>` tags, ensuring that *no other content* (not even a title or paragraphs unless explicitly part of the *Query's* final output requirement) appears within these tags.[4984]

Figure 7: **Qualitative analysis of APEX-DATA prompt evolution.** The **left panel** tracks the optimization trajectory with key strategies discovered. The **right panel** shows the color-coded changes to match these milestones. APEX-DATA accumulates improvements over time.

for long prompts, utilizing history guidance and limiting the mutation scope to ensure stability. Recent research has also expanded the scope of prompt optimization to agentic workflows (Agrawal et al., 2026; Opsahl-Ong et al., 2024), in-context learning (Wu et al., 2024; Wan et al., 2024), and multimodal scenarios (Liu et al., 2024; Wan et al., 2025). Among them, GEPA (Agrawal et al., 2026) employs a Genetic-Pareto framework to optimize agents towards a multi-objective frontier. Despite these advancements, the field has largely overlooked the data-centric perspective of prompt optimization. Most existing methods treat the dataset as a static benchmark, ignoring the varying informational value of different examples. APEX-DATA address this fundamental inefficiency by improving the data selection process itself. By dynamically tracking the rank-sensitivity and solvability of datapoints, APEX-DATA allocates compute wisely.

Data Selection for LLM Alignment. The importance of data quality over quantity for LLM alignment was highlighted by the LIMA hypothesis (Zhou et al., 2023a), which provided empirical evidence that a small, curated set of examples is sufficient for alignment. This insight catalyzed a wave of automated data selection methods. AlpaGasus (Chen et al., 2024b) filters low-quality data to improve fine-tuning efficiency, while LESS (Xia et al., 2024) selects data based on gradient similarity. Additionally, influence-based approaches (Choe et al., 2025) utilize influence functions to identify which training examples most affect downstream capabilities. More recent work has shifted towards dynamic and active data curation, such as Data Advisor (Wang et al., 2024a), which dynamically curates data to improve safety alignment. Parallel to these training-time advances, data selection has recently garnered attention in prompt optimization. Notable approaches, such as those by Diao et al. (2024) and Dong et al. (2025), employ active learning or combine data similarity and model confidence heuristics to curate examples. However, the application scope of these methods is limited by restrictive dependencies, such as human-in-the-loop annotation, reliance on internal model signals (e.g., logits), or restriction to classification tasks. These requirements render them unsuitable for general generative tasks under fully automated, black-box API settings, and non-comparable to our work. APEX-DATA extends the principles of identifying informative data and dynamic curation to prompt optimization, offering a fully automated, black-box solution.

6 CONCLUSION

In this work, we identified the static treatment of data as the fundamental efficiency bottleneck in automated prompt optimization. Existing evolutionary methods waste improved model capabilities on fixed benchmarks, treating informative and uninformative data equally. We addressed this by proposing APEX-DATA, a data-centric framework that transforms the development set from a static ruler into a dynamic pool. By stratifying data into dynamic tiers, APEX-DATA focuses the optimization budget on the addressable frontier for prompt mutation and the rank-sensitive frontier for prompt selection. Our empirical results across instruction following, factuality, and grounding tasks demonstrate that APEX-DATA achieves better performance under the same compute budget. Beyond prompt engineering, these findings suggest a broader principle for black-box optimization. As candidates evolve, the data used to optimize them must evolve together. We expect APEX-DATA to inspire future work to further explore the intersection of dynamic data curation and agent optimization.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, and support qualitative and thematic data analysis. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, and interpret results. and generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, and assist in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for create or edit software code, and edit a research paper to improve readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work significantly improves accessibility to high-performance LLMs by automating complex prompt engineering. By removing the bottleneck of manual expertise, APEX-DATA allows a broader audience, including non-experts and smaller organizations, to achieve state-of-the-art results previously reserved for specialized teams. Furthermore, this efficiency drives a critical shift in computational resource allocation. By demonstrating that the performance gap between optimized prompts and expensive parameter updates is rapidly narrowing, APEX-DATA encourages a move towards more sustainable, energy-efficient model adaptation strategies, fundamentally reducing the computational overhead of AI deployment. However, we acknowledge that automated prompt optimization is a dual-use capability. The same mechanisms that efficiently navigate the prompt space could be misused. APEX-DATA operates as an orchestration layer built on top of LLMs as the final execution engines. As LLMs continue to grow safer, they inherently ensure that optimization tools like APEX-DATA remain securely bounded, regardless of how a prompt is engineered.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we provide complete algorithmic specifications in Algorithm 1 and Appendix B. Full experimental details, including dataset splits, evaluation protocols, models, and hyperparameter configurations, are outlined in Section 4.1. The prompt templates are documented in Appendix C and Appendix D. All baseline frameworks are executed using their official open-source implementations with recommended default settings, including APO[2], MIPROv2[3], TextGrad[4], and GEPA[5].

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 8479–8565, 2026.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. In *International Conference on Machine Learning*, pages 6503–6518. PMLR, 2024a.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024b.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Baker Grosse, and Eric P. Xing. What is your data worth to GPT? LLM-scale data valuation with influence

[2] https://github.com/microsoft/LMOps/tree/main/prompt_optimization

[3] https://github.com/stanfordnlp/dspy

[4] https://github.com/zou-group/TextGrad

[5] https://github.com/gepa-ai/gepa

functions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zPKeJAEo27.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 3369–3391, 2022.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1350, 2024.

Ximing Dong, Shaowei Wang, Dayi Lin, and Ahmed Hassan. Model performance-guided evaluation data selection for effective prompt optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2844–2859, 2025.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 13481–13544, 2024.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZG3RaNIsO8.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge. *arXiv preprint arXiv:2509.07968*, 2025.

Or Honovich, Uri Shaham, Samuel Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, 2023.

Cho-Jui Hsieh, Si Si, Felix Yu, and Inderjit Dhillon. Automatic engineering of long prompts. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10672–10685, 2024.

Alon Jacovi, Andrew Wang, Chris Alberti, Connie Tao, Jon Lipovetz, Kate Olszewska, Lukas Haas, Michelle Liu, Nate Keating, Adam Bloniarz, et al. The facts grounding leaderboard: Benchmarking llms' ability to ground responses to long-form input. *arXiv preprint arXiv:2501.03200*, 2025.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12687–12697, 2024.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Krista Opsahl-Ong, Michael Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, 2024.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, 2023.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, 2023.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

Zhuoyu Shi and Fred Morstatter. The diffusion of causal language in social networks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 18, pages 1422–1435, 2024.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pages 10131–10165, 2025.

Team Gemma, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, and Sercan Arik. Teach better or show smarter? on instructions and exemplars in automatic prompt optimization. *Advances in Neural Information Processing Systems*, 37:58174–58244, 2024.

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, Rajarishi Sinha, and Sercan Ö Arık. Maestro: Self-improving text-to-image generation via agent orchestration. *arXiv preprint arXiv:2509.10704*, 2025.

Fei Wang, Ninareh Mehrabi, Palash Goyal, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. Data advisor: Dynamic data curation for safety alignment of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8089–8100, 2024a.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*, 2024b.

Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Prompt optimization with ease? efficient ordering-aware automated selection of exemplars. In *Advances in Neural Information Processing Systems*, volume 37, pages 122706–122740, 2024.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning*, pages 54104–54132. PMLR, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 355–385, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021, 2023a.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2023b.

A ANALYSIS

Impact of the Lookback Window. A sensitivity analysis demonstrates APEX-DATA's robustness across various lookback configurations. As detailed in Table 3, the default setting of a lookback window of 5 achieves a score of 52.3. Narrowing the window to 3 restricts the historical context, yielding a lower score of 50.3. Conversely, expanding the window to 10 degrades performance slightly to 50.6, as the broader scope begins to reintroduce the outdated, stale signals that our dynamic frontier strategy aims to eliminate. This confirms that a moderately sized lookback window provides the ideal balance between context retention and recency bias.

Table 3: Sensitivity of the lookback window in APEX-DATA on IFBench with Gemini 2.5 Flash.

Setting	Score
lookback = 3	50.3
lookback = 5 (default)	52.3
lookback = 10	50.6

Token Cost. Figure 8 illustrates the trade-off between test accuracy and token consumption for API calls. The results demonstrate that APEX-DATA establishes a clear Pareto frontier, achieving higher test accuracy than the baseline methods.

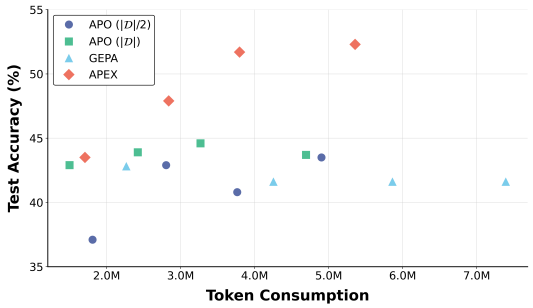


Figure 8: Test accuracy versus token consumption on IFBench with Gemini 2.5 Flash.

Comparison of Prompt Length. Table 4 demonstrates that APEX-DATA's superior performance is not a byproduct of generating longer prompts. While baseline methods like GEPA tend to continuously accumulate instructions, resulting in highly inflated prompt lengths, APEX-DATA achieves the highest overall accuracy with the most concise prompt. This confirms that APEX-DATA does not derive its performance gains from simply introducing more text. Instead, its optimization process successfully distills the instructions to their most critical and precise components, providing highly effective model guidance without redundant bloat.

Table 4: Prompt lengths optimized by different methods on IFBench with Gemini 2.5 Flash.

Method	Prompt Length
APEX-DATA	257
APO ($\|D\|$)	275
APO ($\|D\|/2$)	342
GEPA	662

B ALGORITHM DETAILS

To provide more details on APEX-DATA algorithm and support better reproducibility, we provide the detailed algorithms in Algorithm 2 and Algorithm 3.

Algorithm 2: APEX-DATA Optimization Loop

Input : Initial P_0, Iterations T, Budget N, Batch m, Init anchor α_0, Lookback k
Output : Optimized Prompt P^*

$P_{\text{curr}} \leftarrow P_0$; $\alpha \leftarrow \alpha_0$
$\mathcal{H} \leftarrow \{\text{EvaluateFull}(P_0, \mathcal{D})\}$
for $t = 1 \dots T$ **do**
  // 1. Dynamic Data Stratification
  $\mathcal{B} \leftarrow \text{Stratify}(\mathcal{D}, \mathcal{H}, k)$ // Alg. 3
  // 2. Trajectory-Guided Mutation
  $\mathcal{E} \leftarrow$ Sample m unused from $\mathcal{B}_{M,0} \cup \mathcal{B}_{H,0}$
  $C \leftarrow \text{LLM}_{\text{meta}}(\text{Critique}(P_{\text{curr}}, \mathcal{E}))$
  $P_{\text{new}} \leftarrow \text{LLM}_{\text{meta}}(\text{Mutate}(P_{\text{curr}}, C))$
  // 3. Rank-Sensitive Selection
  $\mathcal{D}_{\text{req}} \leftarrow \mathcal{B}_{M,\emptyset}$
  $R \leftarrow N - |\mathcal{D}_{\text{req}}|$ // Remaining budget
  $\rho_{\text{mix}} \leftarrow \text{PassRate}(\mathcal{B}_M)$; $\rho_{\text{all}} \leftarrow \text{PassRate}(\mathcal{D})$
  $k_{\text{pos}} \leftarrow \lfloor \min(\alpha, \rho_{\text{mix}}, \rho_{\text{all}}) \cdot R \rfloor$; $k_{\text{neg}} \leftarrow R - k_{\text{pos}}$
  $\mathcal{D}_{\text{pos}} \leftarrow$ Sample k_{pos} prioritizing $\mathcal{B}_{M,1}$ then $\mathcal{B}_{E,\emptyset}$
  $\mathcal{D}_{\text{neg}} \leftarrow$ Sample k_{neg} prioritizing $\mathcal{B}_{M,0}$ then $\mathcal{B}_{H,\emptyset}$
  $\mathcal{D}_{\text{eval}} \leftarrow \mathcal{D}_{\text{req}} \cup \mathcal{D}_{\text{pos}} \cup \mathcal{D}_{\text{neg}}$
  Evaluate P_{new} and P_{curr} on $\mathcal{D}_{\text{eval}}$
  if *P_{new} is better than P_{curr} on $\mathcal{D}_{eval}$* **then**
    $P_{\text{curr}} \leftarrow P_{\text{new}}$
    $\alpha \leftarrow \alpha + \beta$ // Anneal anchor
  Update $\mathcal{H}$ with Evaluation of P_{new} and P_{curr} on $\mathcal{D}_{\text{eval}}$
return P_{curr}

Algorithm 3: Dynamic Data Stratification

Input : Dataset $\mathcal{D}$, History $\mathcal{H}$, Lookback k
Output : Granular Data Buckets $\mathcal{B}_{t,s}$

Initialize empty buckets $\mathcal{B}_{t,s}$
for *each* $x_i \in \mathcal{D}$ **do**
  // Retrieve the last k non-empty outcomes
  $\mathcal{R}_i \leftarrow$ Last k valid scores $s(P, x_i)$ from $\mathcal{H}$
  // Determine semantic tier
  if *Set*$(\mathcal{R}_i) \equiv \{1\}$ **then**
    $t \leftarrow$ E (Easy)
  else if *Set*$(\mathcal{R}_i) \equiv \{0\}$ **then**
    $t \leftarrow$ H (Hard)
  else
    $t \leftarrow$ M (Mixed)
  // Lookup cached status (yields $\emptyset$ if uneval)
  $s \leftarrow$ Retrieve $s(P_{\text{curr}}, x_i)$ from $\mathcal{H}$
  $\mathcal{B}_{t,s} \leftarrow \mathcal{B}_{t,s} \cup \{x_i\}$
return $\mathcal{B}$

C INITIAL PROMPTS

```
Respond to the given query. Enclose the final response in <answer> tags to distinguish it from
     the rest of your output.

Query: {query}
```

Listing 1: Initial Prompt for IFBench

```
Respond to the given query. Your final answer must be in <answer> tags.

Query: {query}
```

Listing 2: Initial Prompt for SimpleQA Verified

```
Answer the question based on the given context.
{user_request}
{context_document}
```

Listing 3: Initial Prompt for FACTS Grounding

D APEX PROMPT TEMPLATES

```
You are an Adaptive Prompt Editor. Your goal is to rewrite a prompt based on targeted
     structured feedback.

### Input Data
**1. Original Prompt:**
<current_prompt>
{prompt}
</current_prompt>

**2. Critical Feedback:**
```

```
<feedback>
{feedback}
</feedback>
*(This feedback contains a Locator, a Diagnosis, and an Instruction.)*

---
### Execution Protocol

**1. Analysis Phase**
Before rewriting, explicitly plan your edit based on the input structure:
* **Target:** Find the specific text cited in the feedback's **Locator**.
* **Context:** Read the **Diagnosis** to understand the intent (this ensures you don't fix the
     grammar but miss the point).
* **Constraint Verification:** Ensure your planned edit does not accidentally remove critical
    constraints, negative instructions, or variable placeholders (e.g., `{{variable}}`).

**2. Revision Phase**
Rewrite the prompt using the following **"Logic vs. Syntax"** rules:
* **Variable Lockdown (Strict):** Treat all placeholders (e.g., `{{variable}}`) as immutable
    constants. Do not introduce any new placeholders, and do not add, remove, rename, or
    reformat existing ones. The content represented by the placeholder is non-editable.
* **Logic Lock (Strict):** Do not remove or alter instructions, constraints, or steps that are
     *not* targeted by the **Locator**.
* **Contextual Integration (Flexible):** You *are* permitted to adjust the wording,
    transitions, and grammar of the sentences surrounding the **Locator** to ensure the new
    changes blend naturally. The final result should read as a unified document.

---
### Output Format

**Part 1: Edit Strategy**
Provide a comprehensive, step-by-step plan. You must include:
* **The specific text/section you will modify.**
* **How you will rephrase it to satisfy the Instruction.**
* **How you will ensure surrounding transitions remain smooth.**
* **Confirmation that specific variables/constraints are preserved.**

**Part 2: Revised Prompt**
Output the full text of the revised prompt strictly enclosed within `<new_instruction>` tags.
```

Listing 4: APEX Mutation Prompt

```
You are an expert prompt engineer. Your sole function is to analyze a faulty prompt and
    recommend the single most impactful, generalizable change for the next optimization
    iteration. Adhere strictly to the following process.

### The Prompt Under Analysis
The prompt being evaluated.
<current_prompt>
{prompt}
</current_prompt>

---
### Failure Case Analysis
Specific examples where the prompt produced a suboptimal output.
<failure_cases>
{error_cases}
</failure_cases>

---
### Instructions

**Step 1: Diagnose the Root Cause**
Analyze the **Failure Case Analysis** to identify the primary, underlying reasons for most of
    the errors. Classify the root cause into one of two major types:

* **Type 1: Weak Decision Boundaries (Defining the "What")**
    * **Ambiguity & Vague Definitions:** Terms, tone, or success criteria are open to
        interpretation.
    * **Constraint Loopholes:** Missing exclusionary constraints allow unwanted behaviors.
    * **Input Confusion:** The prompt lacks delimiters (e.g., XML tags, quotes), causing the
        model to confuse user input with instructions.
    * **Missing Context/Grounding:** The prompt fails to explicitly bind the model to provided
         source material (leading to hallucinations).

* **Type 2: Missing Process Instructions (Defining the "How")**
    * **Cognitive Overload:** The task is too complex for a single instruction and requires
        explicit decomposition (breaking the problem into distinct sub-tasks).
    * **Implicit Logic:** The prompt assumes the model knows the specific algorithm required
        to transform input to output.

**Step 2: Formulate the Recommendation**
Based on your diagnosis in Step 1, construct the recommendation according to these principles:

* **1. Apply the Correct Remediation Strategy:**
```

```
    * *If Type 1 (Boundary Issues):*
        * **Operationalize Definitions:** Change subjective adjectives to objective metrics.
        * **Enforce Delimiters:** Recommend wrapping input data in explicit tags (e.g., <input
            >...</input>) to separate it from instructions. Note that the placeholders and
            the corresponding input content are non-editable.
        * **Strengthen Constraints:** Add negative constraints or "Grounding" instructions (e.
            g., "Answer only using the provided text").
    * *If Type 2 (Process Issues):*
        * **Request Rationale:** Require the model to "show its work" or think step-by-step
            before answering. It is allowed to generate intermediate outputs before the final
             answer if a clear output format is specified for answer extraction.
        * **Decompose the Task:** Break the prompt into sequential, modular steps or sub-
            prompts.
        * **Few-Shot Prompting:** If the logic is abstract, strictly require "Input -> Output"
             examples to demonstrate the pattern.

* **2. Provide Generalizable Principles:**
    * Your recommendation must address the root cause, not just the specific failure examples.
    * **Crucially, do not quote or directly reference the provided `<failure_cases>`.** Your
         feedback must be independent of the specific content of the examples.
    * If an example is absolutely essential to illustrate your point, **you must invent a new,
         concise, and clear one** that demonstrates the principle effectively.

**Step 3: Construct the Feedback Object**
Translate your diagnosis into a structured directive for the Editor. You must use these exact
     fields:
* **Locator:** Quote the exact text, section header, or placeholder in the `<current_prompt>`
     where the fix should be applied.
* **Diagnosis:** Explain the specific weakness identified in Step 1 (e.g., "Type 1 Ambiguity:
     The adjective 'short' is subjective.").
* **Instruction:** The specific action for the Editor to take (e.g., "Replace 'short' with '
     maximum 50 words'.").

---
### Required Output Format

You may provide explanatory text or rationale first (e.g., "Analysis: ...").
Your final response must be enclosed in `<actionable_feedback>` tags.
Inside these tags, you must strictly follow this format:

<actionable_feedback>
**Locator:** [Quote text or Header in the `<current_prompt>`]
**Diagnosis:** [Brief Type 1/Type 2 explanation]
**Instruction:** [Precise editing instruction]
</actionable_feedback>
```

Listing 5: APEX Critique Prompt

```
### Failure Example (Score: {score})

**1. Input Context:**
<input>
{query}
</input>

**2. Actual Model Output:**
<actual_output>
{response}
</actual_output>

**3. Evaluation Feedback (Why this failed):**
<critique>
{feedback}
</critique>
```

Listing 6: APEX Error Case Template

E LIMITATION

While APEX-DATA improves the data efficiency of prompt optimization, it shares some fundamental limitations inherent to automated prompt engineering. First, the framework heavily relies on the availability of a representative dataset and a reliable, programmatic evaluation function, which can be challenging to obtain for low-resource or highly subjective tasks. Second, APEX-DATA's effectiveness is bounded by the target model's underlying capacities. It assumes the model possesses a reasonable baseline capability to perform the task when properly instructed, and it cannot overcome a fundamental lack of model knowledge or reasoning. Finally, our current evaluation primarily focuses on text-based LLMs. Future work could extend APEX-DATA to more diverse and complex scenarios, such as agentic workflows and multimodal tasks.

F FUTURE WORK

The APEX-DATA framework opens up several promising dimensions for subsequent research. First, APEX-DATA can serve as a versatile, plug-and-play core optimization layer within established LLM programming frameworks (Yuksekgonul et al., 2025; Khattab et al., 2024). Second, APEX-DATA's underlying mechanism can be leveraged to improve inference-time scaling techniques (Novikov et al., 2025; Snell et al., 2025) by dynamically guiding test-time compute allocation and refining intermediate search paths. Finally, future work can extend APEX-DATA beyond static, text-based scenarios to multimodal tasks, complex agentic workflows, and highly interactive environments. For example, APEX-DATA could be utilized to optimize contextual text rephrasing to shape intricate downstream behaviors, such as prompting knowledge sharing via causal language in social networks (Shi and Morstatter, 2024).

More broadly, our findings highlight a fundamental, yet underexplored principle for black-box LLM optimization. As candidate solutions evolve, the target data distribution dynamically co-evolves. Successfully modeling, tracking, and stabilizing this shifting landscape represents a crucial frontier for the robust optimization of black-box LLMs.