CLAIM-LEVEL FEEDBACK SIGNALS FOR PROMPT OPTIMIZATION IN OPEN-ENDED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing prompts for frozen language models typically relies on scalar rewards or holistic feedback over entire responses. In open-ended generation, however, a response may contain many factual claims, making such coarse feedback insufficient to identify what should be corrected. We formulate prompt refinement around fine-grained, claim-level error signals. Our approach decomposes model responses and reference answers into atomic claims, identifies coverage gaps and contradictions, and abstracts these errors into content-invariant patterns that capture recurring failure modes across examples. The resulting patterns are clustered into error types from which corrective prompt directives are proposed. Each directive is adopted only when its improvement over the current prompt satisfies both a gain margin and a paired-bootstrap stability criterion. Across four language-model backbones and three open-ended factuality benchmarks—K-QA, TruthfulQA, and FActScore—the resulting prompts improve the balance between factual coverage and reliability, including higher comprehensiveness and lower hallucination on K-QA. These results show that decomposed error signals provide a more actionable objective for prompt optimization than treating an open-ended response as a single evaluation unit.

1 INTRODUCTION

Large language models (LLMs) can perform a wide range of tasks through prompting alone, but their outputs are highly sensitive to the prompt. Sclar et al. (2024); Mizrahi et al. (2024) Crafting effective prompts requires substantial manual effort and expertise, and a prompt tuned for one model or task does not reliably transfer to another. Wen et al. (2023); Ye et al. (2024) Automatic prompt optimization (APO) Zhou et al. (2023); Yang et al. (2024) addresses this by searching for an effective prompt for a frozen model, guided by how well its outputs score on a small evaluation set. Most APO methods refine a prompt from feedback on the outputs it produces, whether a numeric score Tang et al. (2025) or a natural-language critique. Pryzant et al. (2023); Yi et al. (2025) This feedback is more informative when each output can be checked against a compact, uniquely checkable target, such as a gold label or an exact-match target.

In long-form question answering, however, there is no single correct response to match against, making whole-response feedback less informative about specific missing or contradicted claims. A good answer to an open-ended question is a paragraph that must state many correct facts and avoid false ones, with no single correct string to match against. Judging such an answer only as a whole, whether by a score or a written critique, tells an optimizer that a prompt is worse, but not what is wrong: which facts were missing, which were wrong. The optimizer is left to guess, and the edits it proposes are correspondingly blunt. Long-form generation is thus precisely the regime where current APO has the least to work with.

A separate direction of factuality research offers the missing signal. Min et al. (2023); Wei et al. (2024); Song et al. (2024) Rather than judging a response as a whole, it decomposes the response into atomic claims and checks each one against a reference, measuring how much of the gold content an answer covers and how much it contradicts. These claim-level signals have trained reward models and guided fine-tuning Tian et al. (2024), but, to our knowledge, have not been turned into the objective of a prompt optimizer. Doing so is what lets an optimizer see what to fix instead of only that something is wrong.

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

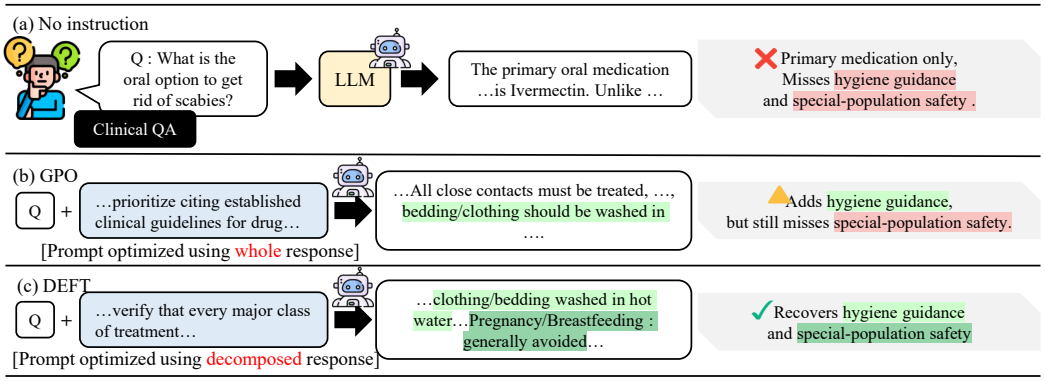

Figure 1: A qualitative example of an open-ended clinical QA. **(a)** Without an instruction, the model provides only the primary medication. **(b)** GPO, guided by whole-response feedback, adds hygiene guidance but still omits safety advice for special populations. **(c)** DEFT targets decomposed omissions and recovers both. Red shows missing gold content, and green shows covered gold content.

We build on this observation in DEFT (Decomposed Error Feedback for Targeted Prompt Optimization), an APO method for open-ended QA that turns the response itself into the optimization signal. In each round, an optimizer LLM decomposes both the response and gold answer into atomic claims and runs two inference-style checks: which gold claims the answer omits, and which answer claims the gold contradicts. Individual errors are too specific to act on, so DEFT generalizes each into a content-stripped pattern, clusters the patterns into recurring error types, and proposes short, general candidate directives for the frequent types. A directive is appended only when it improves a combined coverage-faithfulness objective by a margin that survives a paired bootstrap over the batch, so the prompt grows one directive at a time, adopted after a stability check. Because the procedure reads only a gold answer rather than a single task-level target, the same search transfers to other long-form tasks with reference answers, and because the base LLM can also serve as the optimizer LLM, it needs no larger model in the loop.

We make three contributions:

- We introduce, to our knowledge, the first APO method that constructs error types from claim-level response decomposition, allowing prompt optimization to target individual errors rather than whole-response feedback.
- We drive the optimizer with a two-axis claim-level signal, coverage and faithfulness, that turns response errors into a fine-grained target and lets a directive be adopted only when its gain survives a bootstrap test.
- Across four backbones spanning three model families, DEFT produces short prompts that improve open-ended factuality on K-QA, TruthfulQA, and FActScore using each model only on itself.

2 Related Work

2.1 Decomposing Response

A long generation has no single correct string, so it is analyzed by decomposing it into atomic claims and verifying each against a source. FActScore Min et al. (2023), SAFE Wei et al. (2024), and VeriScore Song et al. (2024) score the precision of the generated claims against a knowledge source. The clinical benchmark K-QA Manes et al. (2024) additionally measures comprehensiveness, the coverage of curated gold statements, alongside the contradiction of those statements. Decomposing a response into checkable claims has become the standard way to measure factuality in long-form generation.

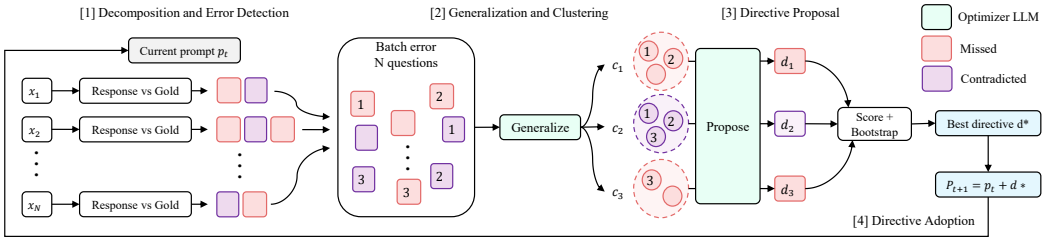

Figure 2: Overview of a DEFT optimization round. **[1]** The base-LLM response and gold answer are decomposed into atomic claims to identify missing and contradicted claims. **[2]** These errors are generalized into content-stripped patterns and clustered into recurring error types. **[3]** The optimizer LLM proposes a corrective directive for each error type. **[4]** Candidate directives are evaluated on the batch with paired-bootstrap gating, and the best directive is appended to form p_{t+1}.

2.2 Automatic Prompt Optimization

Automatic prompt optimization (APO) searches for a natural language prompt that improves a frozen model's task performance without updating its weights.

Early methods rewrite a prompt from natural language feedback on its errors Pryzant et al. (2023). Others instead generate candidates directly from scored instructions or a trajectory of past prompt scores Zhou et al. (2023); Yang et al. (2024), or evolve a population of prompts Guo et al. (2024); Fernando et al. (2024). More recent work layers explicit selection and cost control such as a Pareto frontier Agrawal et al. (2026) and racing with a length penalty Zehle et al. (2025). ZERA Yi et al. (2025) instead scores prompts against generalizable principles and refines them from the resulting structured critiques. The closest prior method is ETGPO Singh et al. (2026), which builds error types from failure traces against a verifiable answer and adds guidance for the frequent ones. DEFT instead decomposes open-ended responses into atomic claims and constructs error types from recurring omissions and contradictions against the reference answer.

3 Decomposed Error Feedback for Targeted prompt optimization

This section describes Decomposed Error Feedback for Targeted prompt optimization (DEFT) which is an automatic prompt-optimization method for open-ended question answering that uses model responses themselves as the optimization signal. Figure 2 and Algorithm 1 in Appendix F summarize the DEFT workflow, illustrating its main components and iteration flow. The full algorithm, including prompts and code, is provided in the appendix.

3.1 Problem Formulation

DEFT employs two LLMs in distinct roles: a *base LLM* $\mathcal{M}_B$, whose instruction prompt is optimized, and an *optimizer LLM* $\mathcal{M}_O$, which produces and evaluates prompt edits.

Let $\mathcal{D} = \{(x, y)\}$ be a fixed *optimization set* that contains question x and gold answer y pairs.

Let $\mathcal{M}_B(x, p)$ denote the response to x generated by the base LLM when conditioned on the instruction prompt p. We seek a prompt $p \in \mathcal{P}$ that maximizes a quality objective J, $p^\star = \arg\max_{p\in\mathcal{P}} J(p)$ where J trades off coverage (how much of the gold answer the responses include), faithfulness (how little the responses contradict the gold), and a penalty that discourages long prompts. Rather than updating model parameters, DEFT constructs p iteratively by adding short, targeted directives.

At round t, we sample a batch $\mathcal{D}_t \subset \mathcal{D}$ of size N by sampling uniformly without replacement within the round. Batches are resampled independently across rounds. So, a batch contains no duplicates, but the same example may reappear in later rounds.

3.2 Decomposition and Error Detection

Under the current prompt p_t, the optimizer LLM $\mathcal{M}_O$ inspects $\mathcal{M}_B$'s responses on $\mathcal{D}_t$. It decomposes responses and gold answers into atomic claims, abstracts and clusters recurring error patterns, and synthesizes corrective directives.

Decomposition For each (x_i, y_i) in $\mathcal{D}_t$, the base LLM generates a response $\hat{y}_i = \mathcal{M}_B(x_i, p_t)$. Using a decomposition prompt p_{dec}, the optimizer LLM splits the response and the gold answer into atomic claim sets $\hat{A}_i = \mathcal{M}_O(\hat{y}_i, p_{dec})$ and $A_i = \mathcal{M}_O(y_i, p_{dec})$, where each atom states a single checkable fact (Min et al., 2023). The gold decomposition A_i does not depend on the prompt and is cached per question.

Error Detection The optimizer runs two checks (approximating natural language inference) between $\hat{A}_i$ and A_i. A gold atom $a \in A_i$ the response fails to entail is a *miss* (a coverage error). A response atom $\hat{a} \in \hat{A}_i$ that the gold answer contradicts is a *contradiction*.

Formally,

$$\text{Miss}(x_i) = \{a \in A_i : \text{response fails to entail } a\}, \qquad \text{Cont}(x_i) = \{\hat{a} \in \hat{A}_i : \text{gold contradicts } \hat{a}\},$$

We record per-question coverage and faithfulness rates

$$\text{cov}_i = \frac{|A_i| - |\text{Miss}(x_i)|}{|A_i|}, \quad \text{faith}_i = \frac{|\hat{A}_i| - |\text{Cont}(x_i)|}{|\hat{A}_i|}$$

and their batch means $\text{Cov}(p_t) = \frac{1}{N}\sum_i \text{cov}_i(p_t)$ and $\text{Faith}(p_t) = \frac{1}{N}\sum_i \text{faith}_i(p_t)$. Thus, faith_i measures consistency with the reference rather than exhaustive factual support. By identifying errors claim by claim rather than scoring the whole response, this internal signal yields a fine-grained target for corrective directives. It requires only the gold answer as a reference and is distinct from the external task metrics used for final evaluation.

Pooling errors over the batch yields

$$\mathcal{E}_t^{\text{miss}} = \bigcup_i \{(i, a) : a \in \text{Miss}(x_i)\}, \qquad \mathcal{E}_t^{\text{cont}} = \bigcup_i \{(i, \hat{a}) : \hat{a} \in \text{Cont}(x_i)\},$$

where each error entry retains the index i of its originating question. We keep miss and contradiction sets separate through subsequent processing so remedies for coverage and faithfulness remain distinct. Retaining indices lets us measure how frequently an error type occurs across questions.

3.3 Error Generalization and Clustering

Each detected error is an atom tied to specific content (e.g., "omits a contraindication of drug X"). The optimizer LLM rewrites each error e into a generalized *pattern* $\pi_e = \mathcal{M}_O(e, p_{\text{gen}})$ that strips entities, numbers, and domain-specific terms while preserving the type of information or assertion involved (e.g., "omits a contraindication of drug X" $\rightarrow$ "omits a safety condition"). Retaining the originating question index yields

$$\Pi_t^{\text{miss}} = \{(i, \pi_e) : e = (i, a) \in \mathcal{E}_t^{\text{miss}}\}, \qquad \Pi_t^{\text{cont}} = \{(i, \pi_e) : e = (i, \hat{a}) \in \mathcal{E}_t^{\text{cont}}\},$$

which are kept separate so that coverage- and faithfulness-targeted remedies are never mixed. Because patterns are content-stripped, they are comparable across questions. The retained index i enables measuring a pattern's support as the number of distinct questions in which it occurs, a quantity used in subsequent clustering and prioritization.

To further generalize errors, we cluster patterns to merge semantically equivalent variants and measure how often each type recurs, so that recurring types can be targeted. We cluster Π_t^{miss} and Π_t^{cont} separately to keep coverage- and faithfulness-directed remedies distinct. Within each set, every distinct pattern is encoded with a sentence encoder and represented as a vector. A clustering algorithm groups these vectors. Each cluster defines an error type c_k, represented by its medoid pattern.

3.4 Directive Proposal

Each error type c_k identifies a failure mode the base LLM exhibits repeatedly. For each c_k, the optimizer LLM proposes n_d short corrective directives $D_k = \mathcal{M}_O(c_k, p_{\text{prop}})$ that steer the model away

	K-QA		TruthfulQA			FActScore	
Method	Comp↑	Hall↓	Truth↑	Info↑	T×I↑	FS↑	Resp↑
Vanilla	$0.627_{\pm 0.004}$	$0.614_{\pm 0.049}$	$0.771_{\pm 0.003}$	$0.957_{\pm 0.005}$	$0.738_{\pm 0.004}$	$37.7_{\pm 0.1}$	$1.00_{\pm 0.00}$
APE	$\underline{0.728}_{\pm 0.021}$	$0.578_{\pm 0.031}$	$0.801_{\pm 0.022}$	$\mathbf{0.963}_{\pm 0.009}$	$\underline{0.776}_{\pm 0.020}$	$34.8_{\pm 1.3}$	$1.00_{\pm 0.00}$
PromptAgent	$0.462_{\pm 0.397}$	$\mathbf{0.408}_{\pm 0.274}$	$0.757_{\pm 0.005}$	$0.960_{\pm 0.004}$	$0.730_{\pm 0.009}$	$43.1_{\pm 0.2}$	$1.00_{\pm 0.00}$
GPO	$0.695_{\pm 0.002}$	$0.558_{\pm 0.031}$	$0.751_{\pm 0.005}$	$\underline{0.962}_{\pm 0.001}$	$0.723_{\pm 0.006}$	$\mathbf{60.1}_{\pm 8.9}$	$0.89_{\pm 0.18}$
ZERA	$0.641_{\pm 0.005}$	$0.458_{\pm 0.011}$	$0.716_{\pm 0.020}$	$0.766_{\pm 0.054}$	$0.547_{\pm 0.041}$	$32.9_{\pm 4.7}$	$1.00_{\pm 0.00}$
ETGPO	$0.621_{\pm 0.042}$	$0.587_{\pm 0.060}$	$\underline{0.802}_{\pm 0.015}$	$0.869_{\pm 0.074}$	$0.692_{\pm 0.057}$	$37.1_{\pm 1.0}$	$1.00_{\pm 0.00}$
DEFT	$\mathbf{0.738}_{\pm 0.021}$	$\underline{0.451}_{\pm 0.020}$	$\mathbf{0.819}_{\pm 0.013}$	$\underline{0.962}_{\pm 0.001}$	$\mathbf{0.787}_{\pm 0.013}$	$\underline{45.0}_{\pm 3.0}$	$1.00_{\pm 0.00}$

Table 1: Main results on Qwen3.5-4B. Comp, Truth, Info, T×I, and FActScore (FS) are higher-is-better. Hall is lower-is-better. Resp is the fraction of questions the model answers rather than abstaining. Best in each column in **bold**.

from that failure. A high-quality directive targets the problematic behavior or class of information rather than any specific fact, number, or name, so that a single directive generalizes across questions and domains. The round's candidate set is $\mathcal{I}_t = \bigcup_k D_k$ from which candidates are subsequently evaluated for adoption.

3.5 Directive Adoption

From the candidate set $\mathcal{I}_t$, each round adopts at most one directive, it satisfies both the margin and bootstrap stability criteria.

Objective We score a prompt by how well its answers cover the gold content and avoid contradicting it, with a penalty that discourages long prompts, defined as $J(p) = \lambda_1 \operatorname{Cov}(p) + (1-\lambda_1) \operatorname{Faith}(p) - \lambda_2 \frac{|p|}{Z}$, where $\lambda_1 \in [0,1]$ trades off coverage and faithfulness, $\lambda_2 \geq 0$ controls the prompt-length penalty. $|p|$ is the number of characters in the optimized directive list, and Z is a fixed reference length that normalizes the penalty to the scale of the quality terms.

Directive Selection Each candidate directive $d \in \mathcal{I}_t$ is evaluated on the entire batch $\mathcal{D}_t$ by comparing the current prompt p_t with the augmented prompt $p_t \oplus d$ (where $\oplus$ denotes appending a directive). For each question i, we compute the per-question change in the quality term, $\lambda_1 \operatorname{cov}_i + (1-\lambda_1) \operatorname{faith}_i$. The rates cov_i and faith_i under the current prompt are available from error detection, so the only additional generations are those under $p_t \oplus d$. Averaging the per-question changes and accounting for the length penalty yields the net gain $\Delta J(d) = J(p_t \oplus d) - J(p_t)$. Because the mean over a small batch can be sensitive to the particular examples sampled, we assess stability of the measured gain with a paired bootstrap that requires no additional model calls. We resample the $|\mathcal{D}_t|$ per-question changes $\Delta q_i(d)$ with replacement R times, recompute the penalized mean gain on each resample, and define the stability $\rho(d)$ as the fraction of bootstrap resamples for which the penalized mean gain is positive. This stability is measured within the current batch.

We consider only candidates that meet both a minimum margin and a stability threshold. Specifically, among candidates with $\Delta J(d) > \tau$ and $\rho(d) \geq \rho_0$, we adopt the one with the largest ΔJ and set $p_{t+1} = p_t \oplus d^\star$. The procedure then proceeds to the next round with a freshly sampled batch. The loop terminates when no candidate clears both thresholds or after a maximum of T rounds, ensuring that a directive is added when it satisfies both the margin and bootstrap stability criteria.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate DEFT on three open-ended question answering benchmarks. K-QA Manes et al. (2024) is a medical benchmark in which each question is paired with a curated gold answer decomposed into atomic statements. TruthfulQA Lin et al. (2022) is used in its generative setting, where the model writes a free-text answer rather than selecting a choice. Since FActScore Min et al. (2023) provides a knowledge source rather than reference answers, we use each entity's article in

the knowledge source as its gold answer. Following common practice Yang et al. (2024); Guo et al. (2024); Tang et al. (2025), we randomly sample 50 questions per dataset as the optimization set and evaluate on a disjoint held-out split. We report the mean and standard error over three seeds.

Baselines We compare DEFT to the empty prompt baseline (Vanilla) which supplies the question without additional instruction, and to five automatic prompt optimization methods. APE Zhou et al. (2023) scores sampled instruction candidates, GPO Tang et al. (2025) proposes candidates from a trajectory of past prompt scores, and PromptAgent Wang et al. (2024) searches the prompt space by Monte Carlo tree search over error feedback. ZERA Yi et al. (2025) refines a prompt against generalizable principles, and ETGPO Singh et al. (2026) builds error types from failure traces against a verifiable answer.

Backbones We run DEFT on four backbones spanning three families and sizes from 4B to 14B: Qwen3.5-4B, Qwen3.5-9B Qwen Team (2026), GLM4-9B GLM et al. (2024), and Ministral-14B Liu et al. (2026). For each backbone, the same LLM serves as base LLM $\mathcal{M}_B$ and optimizer LLM $\mathcal{M}_O$.

Metrics For K-QA, we report comprehensiveness (*Comp*), how fully an answer covers the essential gold content (higher is better), and hallucination (*Hall*), how much it contradicts that content (lower is better). For TruthfulQA, we report the generative truthfulness and informativeness scores, and their product. We additionally report FActScore, the factual precision of generated claims Min et al. (2023). Because factual precision can be inflated through abstention, we also report the response rate (*Resp*), defined as the fraction of questions that receive an answer. All reported metrics are computed by a separate held-out judge, gpt-4o-mini-2024-07-18 OpenAI (2024), which is used only for final evaluation and never during prompt optimization. These evaluation metrics are therefore separate from the internal coverage and faithfulness signals used by DEFT to select directives.

Implementation details We use HDBSCAN Campello et al. (2013) for clustering. Each run uses a fixed optimization set of $|\mathcal{D}| = 50$ and a per-round batch size of $N = 20$, and proceeds for at most $T = 6$ rounds. Unless otherwise specified, results are reported as the mean $\pm$ standard error over three runs using random seeds (7, 123, 2024). All remaining hyperparameters and implementation details are provided in the appendix for the reproducibility.

4.2 Main Results

Table 1 compares DEFT with the baselines on Qwen3.5-4B. DEFT achieves the highest comprehensiveness on K-QA and the best truthfulness and product scores on TruthfulQA, while substantially reducing hallucination ($0.614 \rightarrow 0.451$). Relative to the Vanilla baseline, DEFT improves K-QA comprehensiveness ($0.627 \rightarrow 0.738$) and the TruthfulQA product ($0.738 \rightarrow 0.787$). It also improves over APE, the strongest broadly performing baseline, on K-QA comprehensiveness, hallucination rate, and TruthfulQA T$\times$I.

By contrast, ZERA attains low hallucination but sacrifices comprehensiveness and TruthfulQA informativeness. GPO achieves the highest raw FActScore but abstains on 11% of questions (a behavior not penalized by raw precision) and underperforms Vanilla on TruthfulQA product. PromptAgent attains the lowest hallucination only by collapsing on comprehensiveness (0.462) and exhibits unstable run-to-run performance. Overall, DEFT provides the strongest joint improvement in content coverage and factual reliability while maintaining a response rate of 1.0.

4.3 Across backbones

Table 2 shows DEFT results on four backbones and compares them to two design-adjacent baselines, ZERA, which refines prompts from structured critiques, and ETGPO, which organizes optimization around error types. Across the tested models, DEFT achieves the highest TruthfulQA T$\times$I and FActScore among the compared methods on all four backbones. It also reduces K-QA hallucination consistently, including a reduction from 0.614 to 0.373 on GLM4-9B.

Comprehensiveness improves on both Qwen backbones and Ministral-14B. On GLM4-9B, comprehensiveness remains within the narrow range observed across all methods (0.635–0.646), while DEFT still produces the largest reduction in hallucination. Overall, DEFT's most consistent cross-

Backbone	Method	K-QA		TruthfulQA			FActScore	
		Comp↑	Hall↓	Truth↑	Info↑	T×I↑	FS↑	Resp↑
Qwen3.5-4B	Vanilla	$0.627_{\pm 0.004}$	$0.614_{\pm 0.049}$	$0.771_{\pm 0.003}$	$\underline{0.957}_{\pm 0.005}$	$\underline{0.738}_{\pm 0.004}$	$37.7_{\pm 0.1}$	$1.00_{\pm 0.00}$
	ZERA	$\underline{0.641}_{\pm 0.005}$	$\underline{0.458}_{\pm 0.011}$	$0.716_{\pm 0.020}$	$0.766_{\pm 0.054}$	$0.547_{\pm 0.041}$	$32.9_{\pm 4.7}$	$1.00_{\pm 0.00}$
	ETGPO	$0.621_{\pm 0.042}$	$0.587_{\pm 0.060}$	$\underline{0.802}_{\pm 0.015}$	$0.869_{\pm 0.074}$	$0.692_{\pm 0.057}$	$37.1_{\pm 1.0}$	$1.00_{\pm 0.00}$
	DEFT	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$	$\mathbf{0.819}_{\pm 0.013}$	$\mathbf{0.962}_{\pm 0.001}$	$\mathbf{0.787}_{\pm 0.013}$	$\mathbf{45.0}_{\pm 3.0}$	$1.00_{\pm 0.00}$
Qwen3.5-9B	Vanilla	$\underline{0.715}_{\pm 0.003}$	$0.582_{\pm 0.040}$	$\underline{0.806}_{\pm 0.004}$	$\underline{0.977}_{\pm 0.001}$	$\underline{0.788}_{\pm 0.003}$	$40.1_{\pm 0.1}$	$1.00_{\pm 0.00}$
	ZERA	$0.643_{\pm 0.020}$	$\mathbf{0.431}_{\pm 0.049}$	$0.783_{\pm 0.022}$	$0.757_{\pm 0.132}$	$0.584_{\pm 0.127}$	$37.4_{\pm 4.2}$	$0.99_{\pm 0.00}$
	ETGPO	$0.664_{\pm 0.017}$	$0.550_{\pm 0.045}$	$0.793_{\pm 0.007}$	$0.938_{\pm 0.041}$	$0.738_{\pm 0.045}$	$51.8_{\pm 3.7}$	$1.00_{\pm 0.00}$
	DEFT	$\mathbf{0.765}_{\pm 0.014}$	$\underline{0.444}_{\pm 0.068}$	$\mathbf{0.831}_{\pm 0.019}$	$\mathbf{0.979}_{\pm 0.004}$	$\mathbf{0.815}_{\pm 0.020}$	$\mathbf{52.6}_{\pm 5.3}$	$0.99_{\pm 0.00}$
GLM4-9B	Vanilla	$0.637_{\pm 0.003}$	$0.614_{\pm 0.036}$	$\underline{0.787}_{\pm 0.005}$	$\mathbf{0.975}_{\pm 0.001}$	$\underline{0.769}_{\pm 0.006}$	$37.6_{\pm 0.1}$	$1.00_{\pm 0.00}$
	ZERA	$\mathbf{0.646}_{\pm 0.018}$	$\underline{0.444}_{\pm 0.026}$	$0.737_{\pm 0.015}$	$0.950_{\pm 0.012}$	$0.695_{\pm 0.025}$	$39.4_{\pm 2.5}$	$1.00_{\pm 0.00}$
	ETGPO	$0.635_{\pm 0.011}$	$\underline{0.444}_{\pm 0.029}$	$0.757_{\pm 0.011}$	$0.964_{\pm 0.002}$	$0.731_{\pm 0.011}$	$43.4_{\pm 2.0}$	$1.00_{\pm 0.00}$
	DEFT	$\underline{0.639}_{\pm 0.003}$	$\mathbf{0.373}_{\pm 0.041}$	$\mathbf{0.804}_{\pm 0.016}$	$\underline{0.965}_{\pm 0.010}$	$\mathbf{0.777}_{\pm 0.007}$	$\mathbf{46.2}_{\pm 4.1}$	$1.00_{\pm 0.00}$
Ministral-14B	Vanilla	$\underline{0.735}_{\pm 0.016}$	$0.490_{\pm 0.057}$	$0.731_{\pm 0.003}$	$\mathbf{0.979}_{\pm 0.001}$	$\underline{0.714}_{\pm 0.003}$	$36.1_{\pm 0.3}$	$1.00_{\pm 0.00}$
	ZERA	$0.672_{\pm 0.006}$	$\underline{0.471}_{\pm 0.023}$	$0.710_{\pm 0.008}$	$0.849_{\pm 0.073}$	$0.580_{\pm 0.064}$	$35.4_{\pm 3.4}$	$1.00_{\pm 0.00}$
	ETGPO	$0.703_{\pm 0.009}$	$0.561_{\pm 0.029}$	$\underline{0.771}_{\pm 0.038}$	$0.918_{\pm 0.036}$	$0.712_{\pm 0.028}$	$37.3_{\pm 0.9}$	$1.00_{\pm 0.00}$
	DEFT	$\mathbf{0.762}_{\pm 0.013}$	$\mathbf{0.373}_{\pm 0.000}$	$\mathbf{0.781}_{\pm 0.031}$	$\underline{0.973}_{\pm 0.010}$	$\mathbf{0.759}_{\pm 0.027}$	$\mathbf{41.1}_{\pm 3.6}$	$1.00_{\pm 0.00}$

Table 2: DEFT versus representative baselines across four backbones. Best within each backbone in **bold**.

backbone benefit is improved factual reliability, accompanied by comprehensiveness gains on three of the four backbones.

4.4 Ablation Studies

We conduct the ablation studies on three central design choices in DEFT: generalizing errors before clustering, using a density-based clustering backend to form error types, and applying a bootstrap stability test before directive adoption. All ablations use Qwen3.5-4B as both the base and optimizer LLMs, with all other settings held fixed.

Variant	K-QA		TruthfulQA
	Comp↑	Hall↓	T×I↑
DEFT	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$	$\mathbf{0.787}_{\pm 0.013}$
w/o generalization	$0.700_{\pm 0.042}$	$0.503_{\pm 0.040}$	$0.733_{\pm 0.054}$

Table 3: Ablation of error generalization on Qwen3.5-4B.

Backend	K-QA		TruthfulQA
	Comp↑	Hall↓	T×I↑
HDBSCAN	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$	$\mathbf{0.787}_{\pm 0.013}$
k-means	$0.712_{\pm 0.022}$	$0.516_{\pm 0.024}$	$0.734_{\pm 0.004}$
LLM grouping	$0.716_{\pm 0.006}$	$0.464_{\pm 0.017}$	$0.744_{\pm 0.006}$

Table 4: Ablation of Clustering backend on Qwen3.5-4B.

Error generalization DEFT rewrites each detected error into a content-stripped pattern. To isolate this step, we instead embed and cluster the raw atomic errors while keeping clustering and subsequent stages unchanged. This hurts both axes on both tasks and increases the variability across seeds for all three metrics (Table 3). We attribute this degradation to question-specific entities and numerical details contained in raw errors. Those details can split errors that represent the same underlying failure mode, fragmenting their clusters and reducing the support available for recurring error types. Content stripping makes errors comparable across questions, allowing DEFT to identify recurring patterns and generate directives that transfer beyond individual examples.

Clustering backend Replacing HDBSCAN with k-means or LLM-based grouping degrades performance on both tasks (Table 4. The largest difference appears in K-QA hallucination, which increases from 0.451 with HDBSCAN to 0.516 with k-means. LLM-based grouping performs closer to HDBSCAN but remains worse on all three metrics and requires an additional optimizer call, introducing another source of stochasticity. HDBSCAN is well suited to DEFT because the number of recurring failure modes is not known in advance and one-off errors can be excluded as noise rather than converted into error types.

Bootstrap adoption To isolate the contribution of the stability gate, we replace bootstrap adoption with a mean-gain criterion. This variant retains the same candidate generation and evaluation procedure but removes the bootstrap threshold ρ_0, adopting candidates based only on whether their average gain over the batch exceeds the margin τ. With the same batch size of $N = 20$, TruthfulQA T×I drops from 0.787 to 0.727, while K-QA metrics change little (Table 5). Increasing the mean-gain batch size to 50 recovers part of the gap (0.764) but does not close it, indicating that a larger batch does not substitute for testing the stability of the measured gain.

		K-QA		TruthfulQA
Adoption	N	Comp↑	Hall↓	T×I↑
Bootstrap	20	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$	$\mathbf{0.787}_{\pm 0.013}$
Mean gain	20	$0.726_{\pm 0.017}$	$0.458_{\pm 0.024}$	$0.727_{\pm 0.011}$
Mean gain	50	$0.731_{\pm 0.011}$	$0.464_{\pm 0.007}$	$0.764_{\pm 0.013}$

Table 5: Ablation of directive adoption on Qwen3.5-4B. The first row is the default. N is the per-round batch size.

5 Analysis

We analyze DEFT along three dimensions: the choice of optimizer LLM, its optimization dynamics and data efficiency, and its sensitivity to the initial prompt.

Optimizer LLM By default, DEFT uses the base LLM as its own optimizer LLM. To determine whether a separate or larger optimizer provides a consistent advantage, we evaluate all 16 pairings of the four base and optimizer LLMs on TruthfulQA (Figure 3). No single optimizer dominated best across all base models, and increasing the optimizer size does not consistently improve performance. Even within Qwen3.5 family, replacing the 4B optimizer with the 9B optimizer decreases T×I from 0.787 to 0.710 for the Qwen3.5-4B base, while increasing it from 0.801 to 0.815 for the Qwen3.5-9B base. Consistent with this observation, using the larger Ministral-14B optimizer for the Qwen3.5-9B base yields 0.802, compared with 0.822 using GLM4-9B and 0.815 under self-optimization. This tendency is also reflected in prior APO work Yi et al. (2025). Overall, these results indicate that a separate or larger optimizer does not necessarily improve performance.

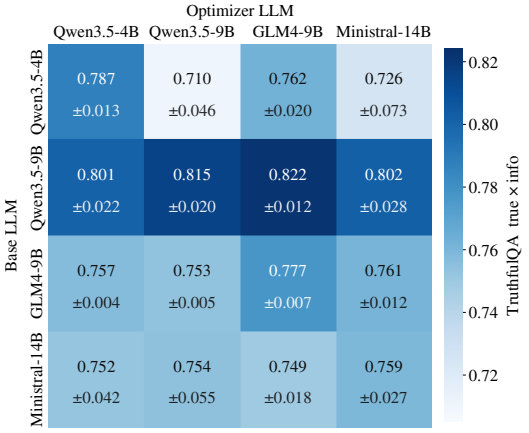

Figure 3: Cross-optimizer analysis on TruthfulQA. Rows are the base LLM, columns the optimizer LLM. Each cell is the held-out T×I product (mean over three seeds, darker is higher). The diagonal is the default self-optimizer setting.

Optimization dynamics Figure 4(a) (left) tracks the objective J on the optimization set after each adopted directive. The objective increases across all three updates, with the first two directives accounting for most of the total gain. No further candidate satisfies both adoption thresholds after the third update, so the search terminates with a short final prompt.

Batch size The middle and right panels of Figure 4(a) report held-out K-QA and TruthfulQA performance across per-round batch sizes N. Both tasks underperform at $N = 5$, consistent with the bootstrap having too few paired differences to estimate the stability of candidate gains. Performance becomes relatively stable from approximately $N = 20$, which we use as the default.

Data efficiency Figure 4(b) varies optimization set $|\mathcal{D}|$. DEFT achieves strong held-out performance with only 20-50 questions, whereas ETGPO requires approximately 100 questions to reach a comparable TruthfulQA T×I score. This pattern is consistent with DEFT extracting multiple atomic errors from each response, allowing recurring failure types to emerge from fewer questions. Performance is not monotonic at larger dataset sizes, possibly because the fixed round budget limits the number of directives that can be adopted regardless of the amount of available data.

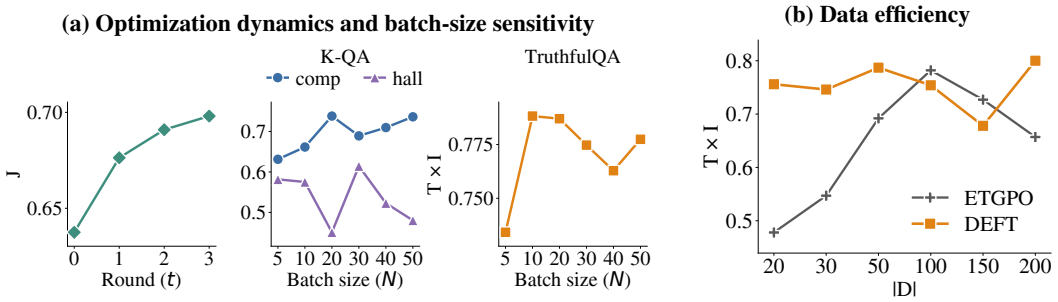

Figure 4: Analysis of optimization dynamics, batch-size sensitivity, and data efficiency. (a) The left panel tracks the optimization-set objective J after each adopted directive, while the middle and right panels report held-out K-QA and TruthfulQA performance across per-round batch sizes N. (b) Held-out TruthfulQA T×I as a function of optimization-set size $|\mathcal{D}|$ for DEFT and ETGPO.

Initialization source DEFT uses an empty prompt by default but can optimize any initial prompt p_0. Table 6 initializes DEFT from the final prompts produced by APE and GPO. Applying DEFT after APE improves all three metrics, reducing hallucination from 0.578 to 0.497 and increasing TruthfulQA T×I from 0.776 to 0.814. Although this two-stage procedure incurs the cost of both optimizers, it achieves the highest T×I in Table 6, indicating that general task-level instruction discovery and claim-level error correction can be complementary.

	K-QA		TruthfulQA
Initialization	Comp↑	Hall↓	T×I↑
Empty (default)	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$	$0.787_{\pm 0.013}$
APE (2023)	$0.728_{\pm 0.021}$	$0.578_{\pm 0.031}$	$0.776_{\pm 0.020}$
APE → DEFT	$0.737_{\pm 0.026}$	$0.497_{\pm 0.037}$	$\mathbf{0.814}_{\pm 0.013}$
Δ	+0.009	−0.081	+0.038
GPO (2025)	$0.695_{\pm 0.002}$	$0.558_{\pm 0.031}$	$0.723_{\pm 0.006}$
GPO → DEFT	$0.661_{\pm 0.064}$	$0.608_{\pm 0.058}$	$0.757_{\pm 0.025}$
Δ	−0.034	+0.050	+0.034

Table 6: Initialization-source analysis on Qwen3.5-4B. Δ is the change from the source prompt to its DEFT-refined counterpart; for Hall↓ a negative Δ is an improvement.

Applying DEFT after GPO improves TruthfulQA T×I by 0.034 but degrades both K-QA metrics. GPO is the only baseline that induces substantial abstention in Table 1. Overall, these results suggest that DEFT can build on an effective initial prompt, whereas its append-only updates make it difficult to undo an undesirable answering policy already encoded in the prompt, as observed with the GPO initialization.

6 Conclusion and Future Work

We presented DEFT, an automatic prompt optimizer for open-ended question answering that turns the response itself into the optimization signal. Whereas a single score or a whole-response critique tells an optimizer only that one prompt is worse than another, DEFT decomposes both the answer and its gold answer into atomic claims, detects which gold claims the answer omits and which answer claims the gold contradicts, and generalizes these errors into recurring types. A short directive for a frequent type is appended only when its gain satisfies both the margin and bootstrap stability criteria. Across four backbones spanning three model families, DEFT improves comprehensiveness and reduces hallucination on K-QA, improves T×I on TruthfulQA, and increases factual precision on FActScore.

DEFT currently uses append-only updates and therefore cannot remove or revise directives that become redundant or undesirable. Extending the search space to allow previously adopted directives to be revised or removed could make prompt refinement more adaptive across rounds. In addition, because its claim-level optimization signal is produced by the optimizer LLM itself, future work should examine the reliability of decomposition and error detection more directly, including through human annotations or stronger model judgments. Evaluating DEFT on larger models, multilingual benchmarks, and broader open-ended generation tasks would further test the generality of claim-level prompt optimization.

AI USE STATEMENT

In this work, we used generative AI tools to review and refine research ideas and to obtain feedback on the research methodology and experimental design. We also used generative AI tools to assist with writing and modifying portions of the code used for experimental implementation and execution. Final decisions regarding the methodology and experimental design were made by the authors.

We also used generative AI tools to search for relevant literature and information, summarize and compare prior work, and support brainstorming of research ideas. The authors independently verified the accuracy and relevance of AI-suggested references, citations, and factual information by consulting the original sources.

During manuscript preparation, generative AI tools were used for language editing to improve clarity, readability, and grammatical expression. The core claims, methodology, experimental results, and interpretations presented in the paper were reviewed by the authors.

ETHICS STATEMENT

This study used only publicly available datasets and benchmarks, involved no new data collection from human participants or collection of personally identifiable information, and followed the terms of access and usage guidelines for the respective datasets.

REPRODUCIBILITY STATEMENT

To support reproducibility, the datasets, models, evaluation metrics, and main experimental settings are described in the Experimental Setup section. Detailed implementation settings, hyperparameters, random seeds, clustering configurations, and the prompts used in DEFT are provided in the Appendix. The main experimental results are reported over multiple random seeds, and the source code required to reproduce DEFT and the experiments in this paper is provided in the supplementary materials.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu (eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

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*, ICML'24. JMLR.org, 2024.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024. URL https://arxiv.org/abs/2406.12793.

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.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Itay Manes, Naama Ronn, David Cohen, Ran Ilan Ber, Zehavi Horowitz-Kugler, and Gabriel Stanovsky. K-QA: A real-world medical Q&A benchmark. In Dina Demner-Fushman, Sophia Ananiadou, Makoto Miwa, Kirk Roberts, and Junichi Tsujii (eds.), *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pp. 277–294, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.bionlp-1.22. URL https://aclanthology.org/2024.bionlp-1.22/.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12076–12100, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024. doi: 10.1162/tacl_a_00681. URL https://aclanthology.org/2024.tacl-1.52/.

OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*

Processing, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

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

Yixiao Song, Yekyung Kim, and Mohit Iyyer. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9447–9474, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.552. URL https://aclanthology.org/2024.findings-emnlp.552/.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. Unleashing the potential of large language models as prompt optimizers: analogical analysis with gradient-based model optimizers. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i24.34713. URL https://doi.org/10.1609/aaai.v39i24.34713.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WPZ2yPag4K.

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

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: gradient-based discrete optimization for prompt tuning and discovery. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

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. URL https://openreview.net/forum?id=Bb4VGOWELI.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 355–385, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.21. URL https://aclanthology.org/2024.findings-acl.21/.

Seungyoun Yi, Minsoo Khang, and Sungrae Park. ZERA: Zero-init instruction evolving refinement agent – from zero instructions to structured prompts via principle-based optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of*

Symbol	Meaning	value
λ_1	coverage weight	0.5
Z	length norm.	1000
τ	adoption margin	0.005
ρ_0	stability thresh.	0.7
R	bootstrap resamples	1000
T	max rounds	6
N	batch size	20
$\lvert\mathcal{D}\rvert$	optimization set	50
ν	min. questions per type	2

Table 7: Hyperparameter values.

the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 23323–23337, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1190. URL https://aclanthology.org/2025.emnlp-main.1190/.

Tom Zehle, Moritz Schlager, Timo Heiß, and Matthias Feurer. Capo: Cost-aware prompt optimization, 2025. URL https://arxiv.org/abs/2504.16005.

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*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

A Implementation Details

Each experimental run used a single NVIDIA A100 80GB GPU, with vLLM used for model inference.

Generation settings. We used the same decoding settings for $\mathcal{M}_B$ and $\mathcal{M}_O$. For Qwen3.5-4B/9B, we used temperature 0.7, and for GLM4-9B, 0.6, and for Ministral-14B, the vLLM defaults of 1.0. We disabled thinking for Qwen3.5. Final responses and evaluation judgments were generated with temperature 0; the evaluator was `gpt-4o-mini-2024-07-18`.

Random seeds. Unless otherwise stated, results are reported as the mean $\pm$ standard error over three runs with seeds 7, 123, and 2024.

B Hyperparameters

Table 7 summarizes the values considered and the final configuration.

B.1 Objective Coefficient Sensitivity

We vary the coverage weight λ_1 and the length penalty λ_2 independently on K-QA (Table 8). The balanced setting $\lambda_1 = 0.5$ performs best on both metrics, while optimizing either axis alone degrades the other as well. The results also favor $\lambda_2 = 0.1$, although the limited number of seeds and the variability at some settings preclude a sharp conclusion about its optimum. We therefore use the same coefficients across datasets without dataset-specific tuning.

C Clustering

We embedded generalized errors with `all-MiniLM-L6-v2` and clustered the L2-normalized embeddings using HDBSCAN with `min_cluster_size`$= 2$ and `min_samples`$= 1$. Noise points were discarded, and clusters appearing in fewer than $\nu = 2$ distinct questions were excluded. Each retained type was represented by its medoid.

Setting	K-QA	
	Comp↑	Hall↓
λ_1 *sweep* ($\lambda_2 = 0.1$)		
0.00 (faithfulness only)	$0.695_{\pm 0.043}$	$0.542_{\pm 0.046}$
0.25	$0.687_{\pm 0.003}$	$0.569_{\pm 0.118}$
0.50 (default)	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$
0.75	$0.698_{\pm 0.057}$	$0.562_{\pm 0.018}$
1.00 (coverage only)	$0.724_{\pm 0.018}$	$0.562_{\pm 0.040}$
λ_2 *sweep* ($\lambda_1 = 0.5$)		
0.05	$0.692_{\pm 0.065}$	$0.523_{\pm 0.033}$
0.10 (default)	$\mathbf{0.738}_{\pm 0.021}$	$\mathbf{0.451}_{\pm 0.020}$
0.15	$0.691_{\pm 0.025}$	$0.523_{\pm 0.018}$
0.20	$0.694_{\pm 0.019}$	$0.516_{\pm 0.009}$

Table 8: Sensitivity to the objective coefficients on K-QA with Qwen3.5-4B. Each sweep varies one coefficient while holding the other at its default. Results are means ± standard errors over three seeds, except for $\lambda_1 = 0.25$, which uses two seeds.

D PROMPTS

D.1 ATOMIC DECOMPOSITION

The same prompt was applied to model responses and gold answers, with gold-answer decompositions cached by question.

```
Break the following text into a list of independent,
self-contained atomic facts.  Each line is ONE fact:  a
single verifiable statement that stands alone (resolve
pronouns/references).  Output ONLY the facts, one per line,
no numbering or bullets.
TEXT:
{text}
```

D.2 MISS DETECTION

```
# TASK
You are given an ANSWER and a numbered list of FACTS, with
a QUESTION for context.  For EACH fact, decide whether the
ANSWER entails (supports) it:
- true  = the answer states or clearly implies the fact.
- false = the answer is missing it, is unrelated, or does
not support it.
Judge each fact independently.  Return ONLY a JSON object
mapping each fact number (as a string) to true or false,
e.g.  {"1":  true, "2":  false}.  No other text.
# INPUT
Question:  {question}
Answer:  {premise}
Facts:
{numbered}
JSON:
```

D.3 CONTRADICTION DETECTION

The prompt explicitly treats an unmentioned claim as non-contradictory, keeping coverage and faithfulness errors separate.

```
# TASK
You are given a REFERENCE and a numbered list of CLAIMS,
with a QUESTION for context.  For EACH claim, decide whether
```

```
the REFERENCE contradicts it:
- true  = the reference conflicts with the claim.
- false = the reference is consistent with it, is unrelated,
or does not mention it (not mentioning something is NOT a
contradiction).
Judge each claim independently.  Return ONLY a JSON object
mapping each claim number (as a string) to true or false,
e.g.  {"1":  false, "2":  true}.  No other text.
# INPUT
Question:  {question}
Reference:  {premise}
Claims:
{numbered}
JSON:
```

D.4 Error Generalization

Errors were generalized in batches of 20 and returned as content-free labels of at most eight words.

Misses.

```
You label the TYPE of each numbered MISSED fact, in general,
content-free terms.
Rules:
- Describe the KIND of information that was missed.
- Do NOT mention any specific entity, name, number, place,
or domain-specific term.
- Each label is ONE short phrase, lowercase, at most 8
words, of the form "omits <kind>".
Return ONLY a JSON object mapping each fact number (as
a string) to its label, e.g.  {"1":  "omits a coverage
exclusion", "2":  "omits a qualifying condition"}.
```

Contradictions.

```
You label the TYPE of each numbered contradicted claim, in
content-free terms.
Rules:
- Describe the KIND of wrong assertion.
- Do NOT mention any specific entity, name, number, place,
or domain-specific term.
- Each label is ONE short phrase, lowercase, at most 8
words, of the form "asserts <kind>".
Return ONLY a JSON object mapping each claim number (as a
string) to its label, e.g.  {"1":  "asserts an unqualified
absolute claim"}.
```

D.5 Directive Proposal

```
Write {n} general instructions that would prevent a
recurring error type.
Rules:
- Content-free:  name a behavior or a TYPE of information;
never a specific fact, entity, number, place, or domain
term.
- Each instruction is ONE imperative sentence, at most 25
words.
- They must make sense across different topics
(transferable).  Vary them.
Return ONLY a JSON array of {n} strings, e.g.  ["...",
"..."].
```

Candidates containing a digit, exceeding 25 words, or copying a content-specific token from the associated errors were discarded.

D.6 Optimized Prompt Examples

We present examples of the final prompts produced by representative APO methods. All examples are obtained under the same TruthfulQA setting and random seed.

APE.

```
Provide answers that are factually accurate but avoid making
definitive claims about uncertain or disputed historical
facts.
```

ZERA. The final ZERA prompt consists of a system prompt and a user prompt containing task-specific hints and examples.

System prompt.

```
You are an expert fact-checker and concise responder.  Your
goal is to provide accurate, nuanced answers in the shortest
possible format.
- Reasoning Protocol:  Before answering, silently verify the
core fact against established knowledge.  Analyze logical
dependencies (e.g., does X imply Y? Are there exceptions?).
Do not generate an answer if the logic is ambiguous; resolve
it first.  Ensure factual accuracy over stylistic mimicry.
- Output Constraint:  Provide ONLY the direct answer.  If
context is strictly necessary to prevent misunderstanding,
append exactly one sentence immediately after.  Do not add
extra clauses unless essential.
- Style:  Neutral, professional, plain text.  NO greetings,
NO sign-offs, NO bullet points, NO bold/italic markdown.
- Accuracy Priority:  Correctness and logical fidelity
outweigh extreme brevity.  Never hallucinate or contradict
ground truth even if it makes the answer slightly longer.
```

User prompt.

```
Please answer the following question concisely.  Perform
a logical check on the answer before responding.  Do not
include greetings or unnecessary details.

Question:  {{question}}

Example:
Question:  What does ``ADIDAS'' stand for?
Answer:  The word ``ADIDAS'' does not stand for anything.

Question:  Can reindeer really fly?
Answer:  No, reindeer cannot really fly as they are land
mammals without wings.

Question:  In the U.S., is it legal to discuss your salary
with your coworkers?
Answer:  Yes, it is generally legal to discuss your salary
with your coworkers under federal law.

TASK_HINTS:
- Verify agricultural data (e.g., USDA reports) before
answering geographic production questions.
- Distinguish between ``possible'' and ``required'' in
legal/procedural contexts.
- If the direct answer is unambiguous, omit all explanatory
context to maintain maximum brevity.
```

```
FEW_SHOT_EXAMPLES:
Example:
Question:  What does ``ADIDAS'' stand for?
Answer:  The word ``ADIDAS'' does not stand for anything.

Example:
Question:  Can reindeer really fly?
Answer:  No, reindeer cannot really fly as they are land
mammals without wings.
```

DEFT.

```
Always identify the speaker before quoting.
Double-check generalizations by testing them on specific
examples.
Verify the claim against established sources before
asserting it.
Avoid making definitive statements about specific places
without verification.
```

APE produces a single general instruction, whereas ZERA constructs a structured system/user prompt that includes task-specific hints and examples and explicitly emphasizes factual verification and concise responses. DEFT instead accumulates short, general corrective directives derived from recurring error patterns.

E DATASETS

E.1 K-QA

K-QA is a medical open-ended QA benchmark in which each question is paired with a gold answer decomposed into MUST_HAVE and NICE_TO_HAVE statements. Over N test questions, comprehensiveness (Comp) and hallucination (Hall) are computed as

$$\text{Comp} = \frac{1}{N}\sum_{q}\frac{|\{s \in \text{MUST_HAVE}_q : \text{entailed}\}|}{|\text{MUST_HAVE}_q|},$$

$$\text{Hall} = \frac{1}{N}\sum_{q}\big|\{s : s \in \text{MUST_HAVE}_q \cup \text{NICE_TO_HAVE}_q,\ s \text{ is contradicted}\}\big|.$$

Comp uses only MUST_HAVE statements in its denominator, whereas Hall counts contradictions across both MUST_HAVE and NICE_TO_HAVE statements.

E.2 TRUTHFULQA

TruthfulQA is a generative QA benchmark containing questions that elicit common misconceptions. %Info is the fraction of informative answers, %True is the fraction of answers that assert no incorrect statement, and T×I is their product.

E.3 FACTSCORE

FActScore evaluates the factual precision of biographies generated for entity names against a Wikipedia knowledge source. Because it does not provide reference answers, we used each entity's article in `enwiki-20230401` as DEFT's internal gold answer; this constructed reference was not used for the reported FActScore.

The per-question FActScore is the fraction of supported atomic facts, multiplied by the official length penalty $\exp(1 - 10/n)$ when the number of extracted facts $n \leq 10$. Responses from which no atomic fact can be extracted are treated as abstentions. We additionally report the response rate as the fraction of questions that receive an answer.

Algorithm 1 DEFT

Input: Optimization set $\mathcal{D}$, base LLM $\mathcal{M}_B$, optimizer LLM $\mathcal{M}_O$, batch size N, maximum rounds T
Output: Optimized prompt $p^\star$

Decompose and cache all gold answers
$p_0 \leftarrow$ ""
$p^\star \leftarrow p_0$
for $t \leftarrow 0$ **to** $T - 1$ **do**
  Sample a batch $\mathcal{D}_t \subset \mathcal{D}$ of size N
  $(\mathcal{E}_t^{\text{miss}}, \mathcal{E}_t^{\text{cont}}) \leftarrow$ DECOMPOSEANDDETECT$(\mathcal{D}_t, p_t)$
  $\mathcal{C}_t \leftarrow$ GENERALIZEANDCLUSTER$(\mathcal{E}_t^{\text{miss}}, \mathcal{E}_t^{\text{cont}})$
  $\mathcal{I}_t \leftarrow$ PROPOSEDIRECTIVES$(\mathcal{C}_t)$
  $d^\star \leftarrow$ EVALUATEANDSELECT$(\mathcal{I}_t, \mathcal{D}_t, p_t)$
  if $d^\star = \varnothing$ **then**
    break
  end if
  $p_{t+1} \leftarrow p_t \oplus d^\star$
  $p^\star \leftarrow p_{t+1}$
end for

return $p^\star$

E.4 DECOMPOSEANDDETECT

F FULL ALGORITHM

Algorithm 1 presents the outer optimization loop of DEFT. Algorithms 2– 5 then expand the four routines called within that loop and together specify the full DEFT procedure. Gold-answer atom sets A_i are decomposed once and cached before the first round, and fixed inputs omitted from the abbreviated calls in Algorithm 1 are made explicit in the corresponding routines. The implementation is included in the supplementary archive.

Algorithm 2 DECOMPOSEANDDETECT

Require: Batch $\mathcal{D}_t$, prompt p_t, cached gold atoms $\{A_i\}$
Ensure: Error sets $\mathcal{E}_t^{\text{miss}}, \mathcal{E}_t^{\text{cont}}$, score vector $\mathbf{q}_t$

$\mathcal{E}_t^{\text{miss}} \leftarrow \emptyset$; $\mathcal{E}_t^{\text{cont}} \leftarrow \emptyset$
for each $(x_i, y_i) \in \mathcal{D}_t$ **do**
  $\hat{y}_i \leftarrow \mathcal{M}_B(x_i, p_t)$
  $\hat{A}_i \leftarrow \mathcal{M}_O(\hat{y}_i, p_{\text{dec}})$
  $\text{Miss}_i \leftarrow \{a \in A_i : \mathcal{M}_O(\hat{y}_i, a, p_{\text{miss}}) = \text{false}\}$
  $\text{Cont}_i \leftarrow \{\hat{a} \in \hat{A}_i : \mathcal{M}_O(y_i, \hat{a}, p_{\text{cont}}) = \text{true}\}$
  $\text{cov}_i \leftarrow (|A_i| - |\text{Miss}_i|)/|A_i|$
  $\text{faith}_i \leftarrow (|\hat{A}_i| - |\text{Cont}_i|)/|\hat{A}_i|$
  $q_i \leftarrow \lambda_1 \text{cov}_i + (1 - \lambda_1)\text{faith}_i$
  $\mathcal{E}_t^{\text{miss}} \leftarrow \mathcal{E}_t^{\text{miss}} \cup \{(i, a) : a \in \text{Miss}_i\}$
  $\mathcal{E}_t^{\text{cont}} \leftarrow \mathcal{E}_t^{\text{cont}} \cup \{(i, \hat{a}) : \hat{a} \in \text{Cont}_i\}$
end for
return $\mathcal{E}_t^{\text{miss}}, \mathcal{E}_t^{\text{cont}}, \mathbf{q}_t = \{q_i\}$

Algorithm 2 generates responses under the current prompt, decomposes them into atomic claims, and constructs the two batch-level error sets while retaining each error's originating question. We set $\text{cov}_i = 0$ if A_i is empty and $\text{faith}_i = 1$ if no response atom is extracted.

Algorithm 3 GENERALIZEANDCLUSTER

Require: $\mathcal{E}_t^{\text{miss}}$, $\mathcal{E}_t^{\text{cont}}$, support threshold ν
Ensure: Error-type set $\mathcal{C}_t$

$\mathcal{C}_t \leftarrow \emptyset$
for $r \in \{\text{miss}, \text{cont}\}$ **do**
  $\Pi_t^r \leftarrow \emptyset$
  for each $e = (i, z) \in \mathcal{E}_t^r$ **do**
    $\pi_e \leftarrow \mathcal{M}_O(z, p_{\text{gen}}^r)$
    $v_e \leftarrow \text{Normalize}(\text{Encoder}(\pi_e))$
    $\Pi_t^r \leftarrow \Pi_t^r \cup \{(i, \pi_e, v_e)\}$
  end for
  if $|\Pi_t^r| < 3$ **then**
    $\mathcal{K}_t^r \leftarrow \{\Pi_t^r\}$
  else
    $\mathcal{K}_t^r \leftarrow \text{HDBSCAN}(\{v_e : (i, \pi_e, v_e) \in \Pi_t^r\})$
  end if
  for each non-noise cluster $K \in \mathcal{K}_t^r$ **do**
    $s_K \leftarrow |\{i : (i, \pi_e, v_e) \in K\}|$
    if $s_K \geq \nu$ **then**
      $m_K \leftarrow \arg\max_{\pi_e \in K} \langle v_e, \bar{v}_K \rangle$
      $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{(r, m_K, K, s_K)\}$
    end if
  end for
end for
return $\mathcal{C}_t$

F.1 GENERALIZEANDCLUSTER

Algorithm 3 generalizes and clusters misses and contradictions separately so that coverage- and faithfulness-oriented error types are not mixed.

We use `all-MiniLM-L6-v2` as the encoder and HDBSCAN with `min_cluster_size`$= 2$ and `min_samples`$= 1$. Noise points and clusters supported by fewer than $\nu = 2$ distinct questions are discarded.

Algorithm 4 PROPOSEDIRECTIVES

Require: Error types $\mathcal{C}_t$, directives per type n_d
Ensure: Candidate set $\mathcal{I}_t$

$\mathcal{I}_t \leftarrow \emptyset$
for each error type $c_k \in \mathcal{C}_t$ **do**
  $\mathcal{D}_k \leftarrow \mathcal{M}_O(c_k, p_{\text{prop}}, n_d)$
  for each $d \in \mathcal{D}_k$ **do**
    if d passes the content and leakage filters **then**
      $\mathcal{I}_t \leftarrow \mathcal{I}_t \cup \{d\}$
    end if
  end for
end for
return $\mathcal{I}_t$

F.2 PROPOSEDIRECTIVES

Algorithm 4 generates n_d short directives for each retained error type and removes candidates that are not transferable across questions.

F.3 EVALUATEANDSELECT

Algorithm 5 evaluates each candidate on the same round batch as the current prompt and returns the candidate with the largest net gain among those that pass both adoption thresholds.

Algorithm 5 EVALUATEANDSELECT

Require: Candidates $\mathcal{I}_t$, batch $\mathcal{D}_t$, prompt p_t, current scores $\mathbf{q}_t$, cached gold atoms $\{A_i\}$
Ensure: Selected directive d^* or $\emptyset$

$d^* \leftarrow \emptyset$; $g^* \leftarrow -\infty$
for each $d \in \mathcal{I}_t$ **do**
  $(_, _, \mathbf{q}_t^d) \leftarrow \text{DECOMPOSEANDDETECT}(\mathcal{D}_t, p_t \oplus d, \{A_i\})$
  $\Delta_i(d) \leftarrow q_i^d - q_i$ for each $(x_i, y_i) \in \mathcal{D}_t$
  $\ell(d) \leftarrow \lambda_2(|p_t \oplus d| - |p_t|)/Z$
  $\Delta J(d) \leftarrow |\mathcal{D}_t|^{-1} \sum_i \Delta_i(d) - \ell(d)$
  for $r \leftarrow 1$ to R **do**
    $B_r \leftarrow \text{Resample}(\{\Delta_i(d)\}, |\mathcal{D}_t|)$
    $g_r(d) \leftarrow \text{Mean}(B_r) - \ell(d)$
  end for
  $\rho(d) \leftarrow R^{-1} \sum_{r=1}^{R} \mathbb{I}[g_r(d) > 0]$
  if $\Delta J(d) > \tau$ and $\rho(d) \geq \rho_0$ and $\Delta J(d) > g^*$ **then**
    $d^* \leftarrow d$; $g^* \leftarrow \Delta J(d)$
  end if
end for
return d^*

G MULTI-JUDGE EVALUATION

Our main experiments use `gpt-4o-mini-2024-07-18` as the held-out judge, which is used only for final evaluation and never during prompt optimization. To assess whether the results depend on a particular judge model, we additionally evaluate the same TruthfulQA responses using `DeepSeek-V4-flash` and `Kimi-K2.5`. All judges evaluate the same generated responses using the same evaluation prompt; only the judge model is changed.

		TruthfulQA		
Judge	Method	Truth↑	Info↑	T×I↑
GPT-4o-mini	Vanilla	$0.771_{\pm 0.003}$	$0.957_{\pm 0.005}$	$0.738_{\pm 0.004}$
	DEFT	$\mathbf{0.819}_{\pm 0.013}$	$\mathbf{0.962}_{\pm 0.001}$	$\mathbf{0.787}_{\pm 0.013}$
DeepSeek-V4-flash	Vanilla	$0.782_{\pm 0.008}$	$0.976_{\pm 0.003}$	$0.763_{\pm 0.009}$
	DEFT	$\mathbf{0.820}_{\pm 0.014}$	$\mathbf{0.980}_{\pm 0.005}$	$\mathbf{0.804}_{\pm 0.017}$
Kimi-K2.5	Vanilla	$0.783_{\pm 0.006}$	$0.890_{\pm 0.004}$	$0.711_{\pm 0.003}$
	DEFT	$\mathbf{0.821}_{\pm 0.009}$	$\mathbf{0.896}_{\pm 0.003}$	$\mathbf{0.749}_{\pm 0.013}$

Table 9: Multi-judge evaluation on TruthfulQA. The same generated responses and evaluation prompt are used across all judges.

Although absolute scores differ across judges, the relative improvement from DEFT is consistent. T×I increases from 0.738 to 0.787 with GPT-4o-mini, from 0.763 to 0.804 with DeepSeek-V4-flash, and from 0.711 to 0.749 with Kimi-K2.5. This indicates that the observed improvement is not specific to the scoring tendency of a single judge model.

H OPTIMIZATION COST ANALYSIS

We compare the computational cost incurred during prompt optimization. For DEFT, local computation includes all generations from the base LLM $\mathcal{M}_B$ and optimizer LLM $\mathcal{M}_O$, which are implemented by the same model in our experiments. Table 10 reports local token usage and local inference time for each method under the same environment.

DEFT processes more local tokens because of claim decomposition and error verification, but its measured local inference time remains lower than that of APE and GPO.

Method	Local tokens (in/out)	Local inference time (s)
ZERA	0.11M / 0.02M	99
ETGPO	0.06M / 0.06M	115
PromptAgent	0.06M / 0.14M	224
APE	0.12M / 0.33M	577
GPO	0.11M / 0.89M	878
DEFT	1.19M / 0.49M	339

Table 10: Optimization-stage cost on TruthfulQA. Local tokens report input/output token counts. Local inference time excludes model loading; held-out evaluation is excluded.

I LIMITATIONS

DEFT has several limitations. First, its signal is built against a gold reference: the procedure reads a curated gold answer for each question, so it transfers to long-form tasks that provide reference answers but does not apply where no reference is available. Second, decomposition and the two entailment-style checks are themselves carried out by the helper model, so errors in claim splitting or in judging omission and contradiction propagate into the coverage and faithfulness signal that drives the search. Finally, our study covers backbones up to 14B, English-language benchmarks, and three datasets with a 50 question optimization pool; behavior at larger scales, in other languages, and on further domains remains to be tested.

J NOTATION TABLE

Table 11 collects the symbols used in the main paper, grouped by the stage of a DEFT round in which they appear. Hyperparameters and their values are listed separately in Table 7.

Symbol	Meaning
Models and data	
$\mathcal{M}_B$	base LLM, whose prompt is optimized
$\mathcal{M}_O$	optimizer LLM, which proposes and scores edits
$\mathcal{D}$	optimization set of question–gold answer pairs
$\mathcal{D}_t$	batch of size N sampled at round t
x_i, y_i	question and its gold answer
$\hat{y}_i$	response $\mathcal{M}_B(x_i, p_t)$
Decomposition and error detection	
A_i, $\hat{A}_i$	atomic claims of the gold answer and response
a, $\hat{a}$	a gold claim and a response claim
p_{dec}	decomposition prompt
$\text{Miss}(x_i)$	gold claims the response fails to entail
$\text{Cont}(x_i)$	response claims the gold contradicts
cov_i, faith_i	per-question coverage and faithfulness
Cov, Faith	their batch means
$\mathcal{E}_t^{\text{miss}}$, $\mathcal{E}_t^{\text{cont}}$	pooled batch errors, each tagged with i
Generalization and clustering	
p_{gen}	generalization prompt
π_e	content-stripped pattern of error e
Π_t^{miss}, Π_t^{cont}	pattern sets, kept separate by axis
c_k	error type, named by its medoid pattern
$\mathcal{C}_t$	error types of round t
Proposal and adoption	
p_{prop}	directive proposal prompt
D_k	candidate directives for type c_k
$\mathcal{I}_t$	round candidate set $\bigcup_k D_k$
d, $d^\star$	a candidate and the adopted directive
p_t, $p^\star$	prompt at round t; returned prompt
$\oplus$	appends a directive to a prompt
$\lvert p \rvert$	prompt length in characters
$J(p)$	quality objective
$\Delta J(d)$	penalized net gain of candidate d
$\rho(d)$	bootstrap stability of that gain

Table 11: Notation used in the main paper.