FROM REFLECTION TO DISCOVERY: ACTIVE KNOWLEDGE ACQUISITION FOR TRAINING CONTEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing large language models (LLMs) are expensive and time-consuming to adapt after deployment, especially when a task requires newly produced information or niche domain knowledge. Recent work has shown that LLMs can iteratively update their own context to adapt to downstream tasks without updating their weights. However, relying solely on their intrinsic knowledge limits their ability to acquire missing task-relevant information: feedback may identify errors without providing the knowledge needed to correct them. In this paper, we equip these LLMs with Wikipedia search and browser tools to actively acquire missing information during context updates. Simply adding these tools to standard sequential context training can degrade performance. We therefore adopt a beam-search-style procedure that explores multiple candidate contexts and retains states selected by validation feedback, including the previous best context. We evaluate this combination on low-resource translation (FLORES+), healthcare (HealthBench), and reasoning tasks (LiveCodeBench and Humanity's Last Exam), observing gains over the evaluated non-information-seeking baselines. Further analyses examine data efficiency and hyperparameter sensitivity on low-resource translation, and show transfer of the optimized context from Gemini-2.5-Flash to Gemini-3-Flash.

1 INTRODUCTION

Large language models (LLMs) have achieved success across diverse domains (Team, 2025; Singh et al., 2025; Yadav & Bethard, 2018; Zhang et al., 2018; El-Kishky et al., 2025; Balunovic et al., 2025; Li et al., 2025a; Ren et al., 2025), yet adapting them after deployment remains challenging especially when tasks require newly produced information or niche domain knowledge (Gao et al., 2025; Fang et al., 2025). Retraining or fine-tuning models with new data can be costly and time-consuming, hindering timely adaptation. An alternative is to train the model's context: an evolving working memory that supports adaptation while keeping the underlying weights frozen (Wang et al., 2024; Cheng et al., 2024; Liu et al., 2025). In context training, an LLM-based optimizer reflects on a data batch, such as past task attempts and feedback, and updates a persistent context for future use. By selecting, abstracting, and refactoring experiences into a dynamic knowledge base or skill set, such systems can achieve positive transfer on relevant tasks without altering a single parameter (Li et al., 2026). For example, pioneering prompt and context optimization frameworks such as ProTeGi (Pryzant et al., 2023), TextGrad (Yüksekgönül et al., 2024), and DSPy (Khattab et al., 2024) demonstrate the potential of adapting frozen models through their inputs.

Despite their initial success, most existing approaches are constrained by a fundamental drawback: they are *closed* systems. Lacking external grounding and access to external sources of information, these frameworks primarily rearrange and refine the optimizer agent's existing internal knowledge, making it difficult to incorporate task-relevant information that falls outside the model's parametric memory. This creates a critical bottleneck: When the desirable information (*e.g.*, a report released after training or a niche technical fact) lies outside the model's frozen parametric knowledge, the optimizer agent cannot reliably discover it and make effective updates to the context. This issue is especially acute when the training feedback identifies that the executor is wrong but does not itself contain the missing knowledge needed to repair the context. In that case, a closed optimizer can only reorganize or extrapolate from its existing knowledge, and the resulting update is written directly into the context used by future executor calls. Moreover, the system may amplify hallucinations rather

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.

than verify the ground truth. As noted in recent studies on the "curse of recursion" (Shumailov et al., 2024), such self-consuming loops without external data often lead to *context collapse* (Zhang et al., 2025b), where the diversity and utility of the optimized context suddenly degrade.

Nevertheless, simply granting the optimizer external access (e.g., web) does not guarantee success. In the standard sequential context optimization pipeline, we identify two critical failure modes: (1) Context Pollution: Given the uncontrolled nature of web content, the model risks ingesting low-quality or misleading information into the context. According to our preliminary study, the optimizer agents struggle to recover from these context-polluting updates, especially when the context optimizer agent lacks an explicit backtracking mechanism. (2) Local Optima: During training, a greedy optimization strategy may commit to sub-optimal trajectories early on, achieving only marginal gains while missing better-performing alternative solutions.

To address these issues, we adopt a beam-search-style training process that maintains a pool of candidate contexts, explores diverse candidate updates, and discards trajectories that are contaminated by low-quality external data or trapped in weak strategies. We also include the current best context in the candidate pool as a "Do Nothing" option, so the procedure need not replace it with a lower-scoring proposal. This ensures that, if *all* new explorations turn out to be noisy or unhelpful, the optimizer can simply retain the previous best state. We evaluate our approach across diverse domains, including low-resource translation (FLORES+), healthcare (HealthBench), and reasoning-heavy tasks (LiveCodeBench & Humanity's Last Exam). We observe consistent performance gains over the sequential, closed-context training baseline when active information seeking is paired with this search-based training procedure, *without* manual task-specific optimization or prompt tuning. Moreover, our analyses and ablation studies show that the method is data-efficient and robust to hyperparameter choices, and that context optimized for one model transfers well to other models.

2 METHODOLOGY

2.1 PRELIMINARY: LEARNING AS STATE OPTIMIZATION

We begin with a general view of learning as state optimization. This perspective places standard parameter tuning and context training under the same notation. Specifically, a *general learning system* minimizes the divergence between the system's prediction and the desired outcomes with the following components $\Lambda = \langle \mathcal{M}, \mathcal{S}, \mathcal{O}, \mathcal{D}, R \rangle$:

1. $\mathcal{M} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$: An inference function mapping inputs $x \in \mathcal{X}$ and state $S \in \mathcal{S}$ to the system's prediction $\hat{y} \in \mathcal{Y}$.

2. $\mathcal{S}$: The modifiable state space encoding the system's knowledge. In standard deep learning systems, this state is usually the model parameters; in other settings, it may be a soft prompt, cache, memory bank, or input context.

3. $\mathcal{O} : \mathcal{S} \times \mathcal{B} \to \mathcal{S}$: An optimizer that updates the state based on a learnable batch $\mathcal{B}$.

4. $\mathcal{D}$: A distribution over the task space $\mathcal{X}$.

5. R: A reward function evaluating the discrepancy between the prediction $\hat{y}$ and the ideal output.

These components interact within a cyclic learning pipeline. At each learning step t, given a batch of input $X_t \sim \mathcal{D}$, the system generates its corresponding prediction $\hat{Y}_t = \mathcal{M}(X_t; S_t)$ and receives feedback $r_t = R(X_t, \hat{Y}_t)$ from the reward function. This forms a learnable batch $\mathcal{B}_t = (X_t, \hat{Y}_t, r_t)$. The optimizer $\mathcal{O}$ then updates the current state as $S_{t+1} \leftarrow \mathcal{O}(S_t, \mathcal{B}_t)$ to reduce the discrepancy between the generated outputs and the optimal behaviour. The final objective of the system is to determine the optimal state S^* that maximizes the feedback function R over $\mathcal{D}$:

$$S^* = \arg\max_{S \in \mathcal{S}} \mathbb{E}_{x \sim \mathcal{D}}[R(x, \mathcal{M}(x; S))] \tag{1}$$

Standard gradient-based learning represents a specific instantiation of this process, where the modifiable state is the parameter and the optimizer is defined by gradient updates. Other instantiations may optimize continuous prompts or cached states with gradients. In this work, we focus on a frozen-weight instantiation where the modifiable state is a discrete, human-readable context C.

2.2 Context Training as a Frozen-Weight Instantiation

Under this instantiation, *context training* modifies the model's behavior (prediction $\hat{y}$) without altering its weights θ. We refer to the LLM-based components in this pipeline as *agents* because they are invoked with role-specific instructions and tool access. The *executor agent* solves the specific given task instances conditioned on the current established context, while the *optimizer agent* reads trajectories collected and feedback from the *executor agent* and update the context. As shown in Fig. 1, context training involves three steps analogous to the role of optimization in gradient-based learning:

1. **Forward pass:** The executor agent processes the input task conditioned on the existing context.

2. **Loss function:** The outputs from the executor agent are passed to a reward function to quantify the performance gap. This signal may take the form of a scalar score, a verifiable reward, or natural language feedback diagnosing the error.

3. **Update step:** An optimizer agent analyzes the feedback and updates the context. In most prior context optimization works, this step entails rewriting the system prompt to rectify errors for the subsequent iteration. Prompts for both agents are detailed in App. F; importantly, these prompts are kept general-purpose rather than being specially optimized for any one task family.

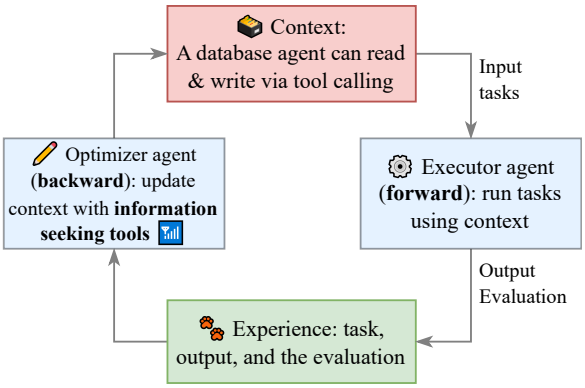


Figure 1: The context training pipeline. In this paper, we propose two main modifications: instantiating the context as a structured database and augmenting the optimizer agent with information-seeking tools.

Furthermore, we introduce two primary modifications in this work, which are introduced in detail in the next section: (1) We instantiate the context as an external structured database that models can read from and write to via function calling; (2) We augment the optimizer agent with information-seeking tools, enabling it to retrieve missing information from the web without being limited by its frozen parametric knowledge.

2.3 Context Management and Information Seeking Tools

Context Management Tool In this work, we instantiate the context as a structured database composed of discrete *resource* items, distinct from the traditional monolithic textual prompt. Each *resource* comprises several attributes: (1) a unique resource ID; (2) a concise TLDR summarizing the item; (3) the raw content; and (4) metadata including the information source, length, keywords, and a text embedding generated by gemini-embedding-001[1]. We implement an interface that enables the optimizer agent to interact with this structured database via tool calls. Functionally, the interface supports essential "write" operations, including initializing an empty context, and adding, deleting, or updating specific resource items. It also facilitates various "read" actions, enabling the model to preview the current context, retrieve specific resources by ID, or search for relevant content via keywords, embeddings, or a dedicated retrieval sub-agent. Compared to standard monolithic textual prompts, this tool offers greater precision in manipulating context. It allows the optimizer agent to surgically update or remove specific content without regenerating or reprocessing the entire context, while enabling the executor to retrieve only the resources most relevant to the current task. A more detailed description of this tool is provided in Tab. 8 and App. E.

Information Seeking Tools To transcend the closed nature of existing context training pipelines, we equip the agent with external grounding capabilities via two specialized tools: (1) **WikipediaSearchTool**: implemented based on the Python `wikipedia` library[2], the tool makes it easy to access and parse data from Wikipedia. (2) **BrowserUseTool**: this tool enables the agent to navigate web pages dynamically. It can parse HTML content to extract code snippets, recent reports, or

[1] Gemini embeddings documentation [2] Wikipedia Python package

documentation that Wikipedia has not yet indexed. This tool is particularly beneficial when the model possesses only vague notions of the desired information. Our implementation leverages the `browser-use` library[3]. We include the WikipediaSearchTool to enable the optimizer agent to query specific concepts easily. It is primarily triggered when the optimizer detects gaps in declarative knowledge (*e.g.*, missing definitions). For more complex information-seeking scenarios, we prompt the model to use browsers, as this is a more general way for agents to retrieve information from the web. By integrating these tools, the optimizer $\mathcal{O}$ transitions from a pure reasoning engine to an active searcher. In our pipeline, before proposing an update S_{t+1}, the optimizer can invoke these tools to verify its internal priors or acquire new evidence, ensuring that the semantic gradients applied to the context are grounded in the external world.

2.4 Pitfalls of the Sequential Context Training

Standard context training typically employs a linear strategy that retains a single context C_{t+1} at each training step and updates it. Nevertheless, as evidenced by our preliminary study, simply incorporating information-seeking tools into the standard linear training could introduce new risks, as detailed below.

Context Pollution Figure 2a illustrates our preliminary study on the *English-to-Chokwe* translation task. We observe that the context could be poisoned with tiny updates, resulting in a severe performance drop, and the optimizer agent struggles to remove these harmful artifacts once they are introduced. As shown in the shaded region (steps 4 to 16), a mild update to the context (about 200 tokens) is associated with a precipitous decline in the performance. Crucially, the system fails to recover from this "pollution". Instead of pruning the toxic content, the optimizer repeatedly adds and removes information (steps 16 to 128). At the same time, the performance remains very low, highlighting the necessity of an explicit backtracking mechanism that helps the model "undo" such mistakes.

Local Optima The second flaw is that the model is prone to getting trapped in local optima, resulting in a repetitive cycle of accumulation and collapse. As shown in Figure 2b (English-to-Buginese translation task), the context length (dashed black line) exhibits a distinct sawtooth pattern: it grows steadily before suffering sudden, sharp declines, termed as the "context collapse" observed in prior studies (Zhang et al., 2025b), where models fail to maintain information density as length increases. A closer inspection of the context composition reveals a more specific failure mode. While the *Dictionary Support* (orange region) consistently dominates the context, the optimizer does periodically attempt to prune these resources. Yet, crucially, these pruned resources are invariably re-added in subsequent steps. This implies that the optimizer is stuck in a loop: it tries to compress the context but fails to discover superior strategies (such as increasing *Parallel Examples*, the blue region), and thus is forced to revert to the "safe" but suboptimal strategy of dictionary expansion. This cyclical inability to escape the current strategy basin underscores the critical lack of effective exploration mechanisms in standard sequential training, especially when the context-optimizer agent has access to external information of varying quality.

2.5 Context Optimization with Beam Search

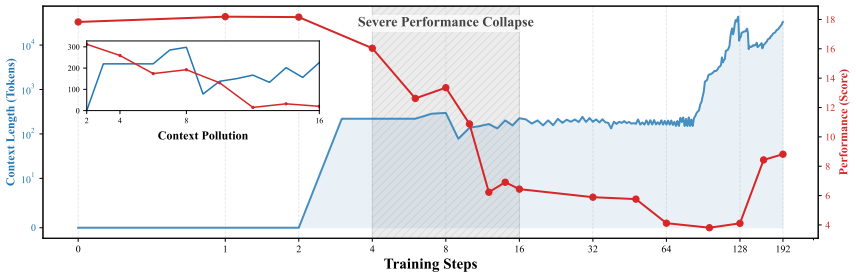
To address the two issues mentioned above, we adopt a beam-search-style training procedure (Vijayakumar et al., 2016). Instead of maintaining a single context trajectory and performing greedy sequential edits, we maintain a small population of K candidate contexts, denoted by $\mathbb{C}_t$, and prune them using validation feedback:

$$\mathbb{C}_t = \{c_t^{(1)}, c_t^{(2)}, \ldots, c_t^{(K)}\}$$

This process is illustrated in the Fig. 3. Specifically, at each training step t, the pipeline proceeds in two phases:

Expansion (Exploration): For each candidate in the current beam, the optimizer agent $\mathcal{O}$ generates M child contexts by running the context-training loop in Fig. 1 (dashed arrows in Fig. 3). Each child is optimized for L update steps on training batches sampled from $\mathcal{D}_{train}$, yielding multiple

[3] Browser-use repository



(a) Context Pollution on English-to-Chokwe: an update is followed by a sharp performance drop, with limited subsequent recovery.



(b) Local optima on English-to-Buginese: the optimizer repeatedly expands and prunes dictionary support, with limited performance gains.

Figure 2: Failure modes in the standard sequential context training pipeline with information-seeking tools.

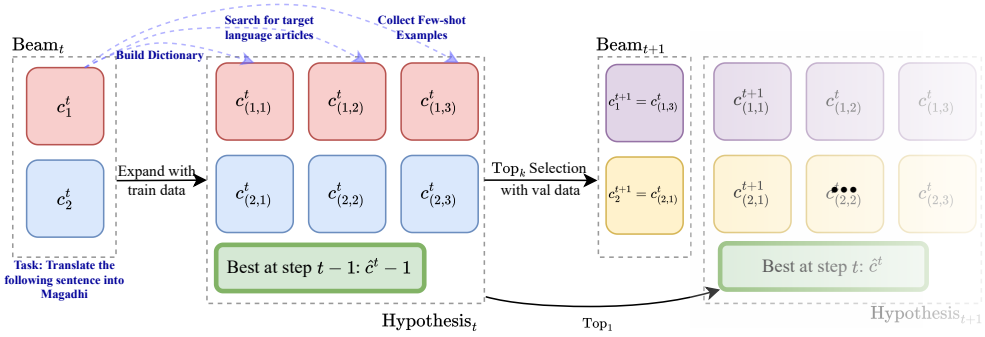


Figure 3: Beam Search guided context training process. The optimizer agent could test different optimization strategies and reject lower-performing ones. We also include the best candidate from the previous step as the "Do Nothing" option.

alternative context updates from the same parent. In machine translation, for example, different branches may emphasize building a dictionary, searching for reference articles, or collecting few-shot examples. These strategies are not hard-coded as a fixed menu; rather, they are discovered by the optimizer during expansion. To encourage diversity, children from the same parent are generated sequentially: when generating the context $c_{(i,j)}^t$, we provide the model with a short summary of the previous explorations $c_{(i,<j)}^t$ and explicitly prompt it to pursue a different update strategy.

Selection (Pruning): We evaluate generated candidates on a held-out validation set and compare their scores with that of the previous best candidate ($\hat{c}_{t-1}$, depicted as the green block). The update

Table 1: Performance on low-resource MT (Chrf++) and HealthBench (aggregated score with LLM-judge).

Method	LRMT Avg.	HealthBench
Gemini-2.5-Flash	26.31	0.3793
Gemini-2.5-Pro	30.37	0.5030
Context Training		
BoN	31.94	0.4090
Seq	31.52	0.4629
BeamSearch	31.31	0.4650
Context Training w/ Information Seeking		
Seq-IS	29.68	0.4484
BeamSearch-IS	**34.54**	**0.5026**

Table 2: Pass@1 and pass@8 for LCB, and avg@8 on HLE. Full results are in App. B.

Method	LCB Overall	HLE Avg.
Gemini-2.5-Flash	49.4 / 65.6	6.53
Context Training		
BoN	49.2 / 65.9	6.13
Seq	49.0 / 67.5	6.20
BeamSearch	49.3 / 67.1	6.51
Context Training w/ Information Seeking		
Seq-IS	49.3 / 68.1	6.45
BeamSearch-IS	**52.5 / 72.1**	**8.63**

rule is:

$$\mathbb{C}_{t+1} = \text{Top}_K \left(\underbrace{\{\hat{c}_{t-1}\}}_{\text{Elitism}} \cup \bigcup_{c \in \mathbb{C}_t} \mathcal{O}_{\text{expand}}(c, \mathcal{B}_t) \right) \tag{2}$$

where $\mathcal{O}_{\text{expand}}$ denotes the set of candidate contexts produced from c through the expansion phase. Selection ranks all candidates by their validation scores and retains the top K for further expansion. It also allows the search to abandon strategies that make short-term progress but lead to weaker validation performance. Including the best candidate from the previous step acts as a "Do Nothing" option: if no new candidate achieves a higher validation score, the optimizer simply keeps the previous best state.

Implementation via Version Control: To operationalize this branching pipeline, we implement the context as a version-controlled code repository. The optimizer uses atomic functions to manage context versions, such as `create_branch` (to fork the current context for further expansion), `commit` (to snapshot a specific state of the context), and `check_out` (to revert to a parent node or switch branches). These version-control actions are an implementation mechanism rather than a conceptual component of the method, and are currently hard-coded into the training loop. A more detailed description is shown in Tab. 8.

3 Experiments

3.1 Experiment Settings

We evaluate on low-resource machine translation (FLORES+), healthcare dialogue (HealthBench), competitive coding (LiveCodeBench), and multidisciplinary reasoning (HLE), using Gemini-2.5-Flash unless otherwise specified. We compare the base model, Best-of-N, sequential and beam-search context training, and their information-seeking variants, matching optimizer calls within each tool-access setting. For FLORES+, HealthBench, and LiveCodeBench, context optimization uses 128 training examples and 64 validation examples; HLE uses category-specific splits. Metrics, data splits, and full baseline definitions are in App. D.

3.2 Main results

Results on Low-resource Machine Translation and HealthBench Table 1 shows the results on Low-Resource Machine Translation (LRMT) and HealthBench; detailed per-language and per-theme breakdowns are provided in App. B. On LRMT, directly equipping the sequential model with information-seeking tools hurts average performance, dropping from 31.52 (Seq) to 29.68 (Seq-IS), which provides further evidence of **Context Pollution**. In contrast, BeamSearch-IS reaches **34.54**, improving over the best non-IS context-training baseline (31.94) and the larger Gemini-2.5-Pro

baseline (30.37).The same trend appears on HealthBench. Simply adding information-seeking tools to sequential training degrades the overall score from 0.4629 to 0.4484, whereas BeamSearch-IS improves to **0.5026**, comparable to the much larger and more expensive Gemini-2.5-Pro (0.5030). Across these two domains, information seeking improves performance when combined with beam search, but degrades performance under sequential context training.

Results on Complex Reasoning Tasks Table 2 summarizes the performance on LiveCodeBench (LCB) and the Humanity's Last Exam (HLE). We observe three key trends: (1) Methods relying solely on internal reasoning (BoN, Seq, BeamSearch) yield negligible improvements over the Gemini-2.5-Flash baseline. The overall pass@1 on LCB hovers around 49%, and HLE accuracy remains approximately 6.5%, indicating that optimizing reasoning paths without external information yields limited gains in our setting. (2) The Seq-IS method shows inconsistent results across domains. While it remains robust on LCB (improving pass@8 to 68.1%), its average HLE accuracy decreases slightly from 6.53% to 6.45%, with larger declines in Physics and Math. (3) In contrast, BeamSearch-IS achieves consistent gains across both benchmarks. It improves LCB overall performance to 52.5 / 72.1 and achieves the highest average accuracy on HLE (8.63%), effectively outperforming both the baseline and other context training methods.

4 Analysis

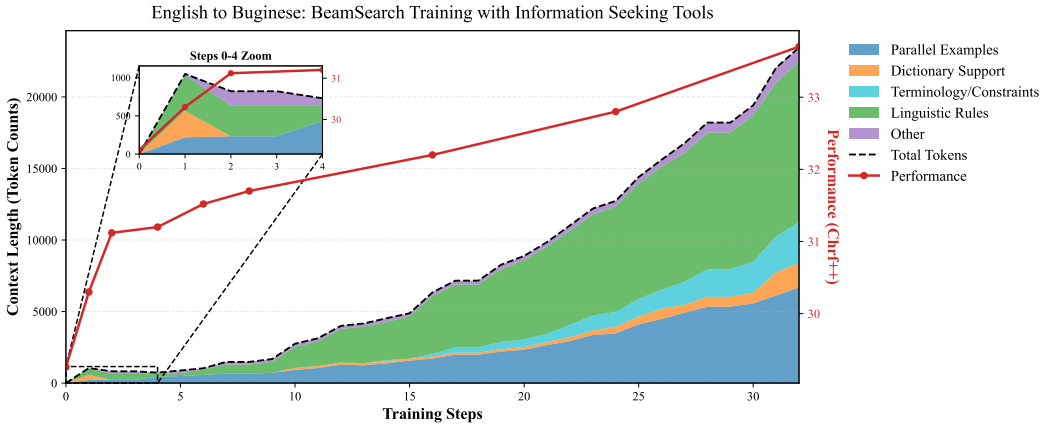
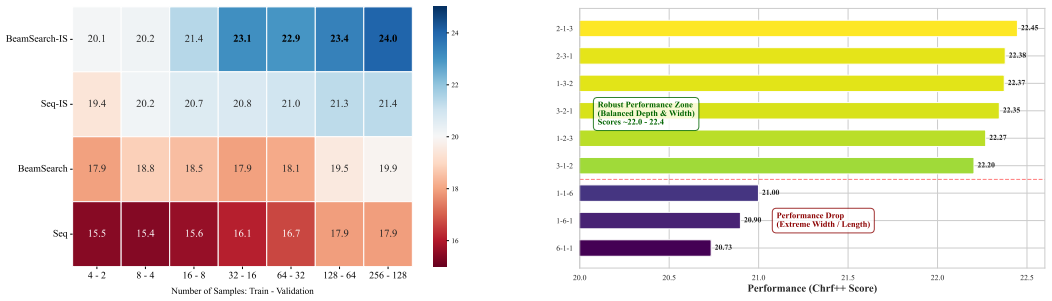


Figure 4: Context composition and translation performance (red line) during BeamSearch-IS training on English-to-Buginese. The zoom-in window shows an initial increase in *Dictionary Support* (orange) at Step 1, followed by a shift toward *Linguistic Rules* (green) and *Parallel Examples* (blue) at Step 2. The trajectory illustrates changes in the selected context as training proceeds.

How Beam Search Enables Exploration Figure 4 illustrates exploration and retention during context training. Compared with the sequential trajectory in Fig. 2b, which repeatedly expands *Dictionary Support*, BeamSearch-IS shifts toward *Linguistic Rules* and *Parallel Examples*, consistent with observations by Aycock et al. (2025). The zoom-in window shows this shift between Steps 1 and 2. The "Do Nothing" candidate also allows the previous best context to survive when proposed updates do not improve its validation score. Intervals with unchanged performance and resource composition are consistent with retaining that context rather than accepting an update. These observations illustrate the combined exploration and retention process, without isolating the contribution of rollback from that of branching.

Data Efficiency Analysis Acquiring annotated data is often prohibitively expensive in real-world scenarios. Therefore, we analyze performance across data regimes ranging from 4 to 256 training samples to evaluate the data-efficiency of our approach. We chose the English-to-Southwestern Dinka (dik_Latn) translation task for this analysis. Figure 5a illustrates a clear performance hierarchy. While standard methods (e.g., Seq, bottom rows) remain stuck in low-performance regions (indicated by red hues) even as data volume increases, our BeamSearch-IS method demonstrates remarkable data efficiency. Notably, it achieves near-optimal performance (score $>$ 23.0, dark blue) with as few as 32



(a) Data efficiency on English-to-Southwestern Dinka. Darker blue indicates higher ChrF++ scores.

(b) Hyperparameter ablation. The yellow region marks robust high-performance configurations.

Figure 5: Data efficiency and hyperparameter robustness of BeamSearch-IS.

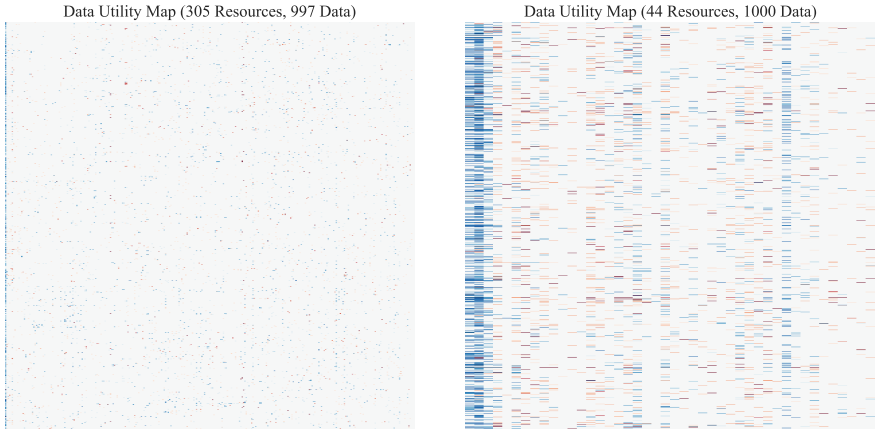


Figure 6: The heatmaps visualize the utility of retrieved resources (x-axis) for test samples (y-axis). Blue indicates positive transfer.

training samples. This suggests that Beam Search acts as a signal amplifier. By exploring multiple potential context modifications for each training example, the optimizer extracts more signal from limited data, allowing it to converge to high-quality strategies with a fraction of the data required by the baseline methods.

Ablation Study on Hyper-parameters We further assess the framework's sensitivity to the hyperparameters of Beam Search: Beam Width, Number of Hypotheses per step, and training epochs. Using the English-to-Southwestern Dinka (dik_Latn) translation task for this ablation study, Figure 5b illustrates the performance variance across distinct combinations under a similar training budget, denoted as (Width-Hypotheses-Epochs). As highlighted by the "Robust Performance Zone" (yellow bars), the system maintains high performance ($\sim$22.2–22.45) across a diverse range of balanced configurations (e.g., 2-1-3 or 3-2-1). This indicates that BeamSearch-IS is relatively insensitive to specific parameter settings. Significant performance degradation occurs only in "extreme, unbalanced settings" (purple bars), such as the 6-1-1 configuration (prioritizing extreme width over training epochs), where the score drops to 20.73. This confirms that as long as a reasonable balance between exploration (width) and exploitation (depth) is maintained, the method yields stable results.

Data Utility Analysis To better understand how the context works, we use Gemini-3-Flash as an annotator to quantify the utility of each resource for each test sample. The specific prompt we used for annotation is in App. F. Figure 6 visualizes this utility map, where the x-axis represents unique context resources (sorted from longer to shorter ones from left to right) and the y-axis represents test data points. Our observations include: (1) The overall maps are predominantly sparse and modular, indicating that the majority of retrieved contexts are highly instance-specific and only effective for a narrow subset of queries. (2) We also observe that dominant resources emerge in both domains, visualized as distinct continuous vertical blue lines on the left. These lines correspond to a small set

of high-quality contexts that provide positive transfer across nearly the entire test set. (3) Crucially, to ensure these dominant resources were not merely leaking answers, we rigorously screened them for data contamination (defined as the presence of a question and its ground-truth answer). We again use Gemini-3-Flash as the data contamination detector; the specific prompt is in App. F. Our inspection confirmed **zero instances of such overlap**, ensuring that the broad utility of these resources stems from genuine knowledge applicability rather than data leakage.

Generalization of the trained context Finally, we investigate a critical question: Does the trained context discover genuinely helpful knowledge, or does it merely learn model-specific artifacts? To test this, we evaluate the *transferability* of the context by applying context optimized on Gemini-2.5-Flash directly to the newer, more capable Gemini-3-Flash without *any* further tuning. Table 3 reveals that the standard closed Seq method exhibits poor transferability and yields no performance gain in most cases. For instance, Seq causes regression on HealthBench (a slight drop from 0.6164 to 0.6011) and struggles with reasoning-heavy HLE subjects like CS/AI and Math. This suggests that these baselines tend to learn executor-specific patterns that do not generalize. Conversely, BeamSearch-IS demonstrates remarkable generalization. It drives consistent gains across all tasks, boosting the Magahi translation score **by nearly 10 points** (to **52.12**) and achieving **0.6624** on HealthBench. Notably, in the HLE benchmark, it consistently delivers gains across diverse fields, and those gains are even larger than on the Gemini-2.5-Flash model. These results show that context optimized for Gemini-2.5-Flash can also benefit Gemini-3-Flash on the evaluated tasks without further context training.

Table 3: Final performance when applying context optimized by Gemini-2.5-Flash to Gemini-3-Flash. Full breakdowns are in App. B.

Method	LRMT Avg.	HLE Avg.	HealthBench
Gemini-3-Flash	37.58	33.24	0.6164
Transferred Context			
Seq	39.54	32.80	0.6011
BeamSearch	40.15	33.21	0.6218
Seq-IS	39.93	34.10	0.6118
BeamSearch-IS	**43.26**	**36.88**	**0.6624**

5 Related Work

Prompt and context optimization methods such as ProTeGi (Pryzant et al., 2023), TextGrad (Yuksekgönül et al., 2024), and DSPy (Khattab et al., 2024) adapt frozen LLMs through their inputs. Prompt engineering (Schulhoff et al., 2024; Sahoo et al., 2024) includes Chain-of-Thought (Wei et al., 2022), genetic search (Fernando et al., 2024), and beam search (Pryzant et al., 2023). Memory systems distill experience into reusable context (Wang et al., 2024; Liu et al., 2025), while Agentic Context Engineering develops structured, incremental updates (Zhang et al., 2025b). Reflection-based methods refine outputs using feedback and the model's existing knowledge (Shinn et al., 2023; Madaan et al., 2023). Feedback may expose errors without explaining their correction; our optimizer seeks the missing knowledge externally. Retrieval-Augmented Generation (RAG) typically retrieves evidence from an existing corpus for a given query (Zhang et al., 2025a; Amugongo et al., 2025; Li et al., 2025b). Our optimizer acquires knowledge during training and evaluates candidate context updates before retention. The executor retrieves from this persistent context across subsequent tasks. We study knowledge acquisition together with candidate exploration and validation-based retention, since external access alone may not yield useful updates. A full discussion appears in App. A.

6 Conclusion

Recent work employing LLMs as context optimizers suggests a promising path for specializing models at test time without updating their weights. However, most existing context optimization methods are passive and closed, which limits their ability to acquire missing external knowledge. In this paper, we study context optimization with active information seeking by augmenting the optimizer with Wikipedia search and browser-based tools. We show that naively adding these tools to sequential context training can be harmful, but that a simple search-based training procedure makes them effective in practice. Across low-resource machine translation, healthcare, and reasoning benchmarks, this combination yields consistent improvements over passive baselines, while also remaining data-efficient and transferring well across models. Limitations and directions for future work are discussed in App. G.

USE OF AI ASSISTANTS

During manuscript preparation, we used AI assistants for writing, language polishing, and figure creation. Separately, the research uses LLMs as executor and optimizer agents, and for HLE question-relevance scoring, resource-utility annotation, and contamination detection, as described in the methods and appendices. The authors are responsible for the research design, interpretation of results, and final manuscript.

REFERENCES

Lameck Mbangula Amugongo, Pietro Mascheroni, Steven Brooks, Stefan Doering, and Jan Seidel. Retrieval augmented generation for large language models in healthcare: A systematic review. *PLOS Digital Health*, 4(6):e0000877, 2025.

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health. *CoRR*, abs/2505.08775, 2025.

Seth Aycock, David Stap, Di Wu, Christof Monz, and Khalil Sima'an. Can llms really learn to translate a low-resource language from one grammar book? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanovic, and Martin T. Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *CoRR*, abs/2505.23281, 2025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

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

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 1107–1128. Association for Computational Linguistics, 2024.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera Gilaberte, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, o3 contributors, Oleg Mürk, Rhythm Garg, Rui Shu, Szymon Sidor, Vineet Kosaraju, and Wenda Zhou. Competitive programming with large reasoning models. *CoRR*, abs/2502.06807, 2025. doi: 10.48550/ARXIV.2502.06807. URL https://doi.org/10.48550/arXiv.2502.06807.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems. *CoRR*, abs/2508.07407, 2025. URL https://doi.org/10.48550/arXiv.2508.07407.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence. *CoRR*, abs/2507.21046, 2025. URL https://doi.org/10.48550/arXiv.2507.21046.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10, 2022.

Yufei He, Juncheng Liu, Yue Liu, Yibo Li, Tri Cao, Zhiyuan Hu, Xinxing Xu, and Bryan Hooi. Evotest: Evolutionary test-time learning for self-improving agentic systems. *arXiv preprint arXiv:2510.13220*, 2025.

Yuanzhe Hu, Yu Wang, and Julian J. McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. *CoRR*, abs/2507.05257, 2025. URL https://doi.org/10.48550/arXiv.2507.05257.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=syt0rnhg7Z.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, K.p. Subbalakshmi, Jimin Huang, Lingfei Qian, Xueqing Peng, Jordan W. Suchow, and Qianqian Xie. INVESTORBENCH: A benchmark for financial decision-making tasks with LLM-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria, July 2025a. Association for Computational Linguistics.

Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in llm agents without gradient updates. *arXiv preprint arXiv:2601.18510*, 2026.

Zongxi Li, Zijian Wang, Weiming Wang, Kevin Hung, Haoran Xie, and Fu Lee Wang. Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence*, pp. 100417, 2025b.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.694/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual*

Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

Saleh Momeni, Sahisnu Mazumder, Zixuan Ke, and Bing Liu. In-context continual learning assisted by an external continual learner. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7292–7306, 2025.

Ryumei Nakada, Wenlong Ji, Tianxi Cai, James Zou, and Linjun Zhang. A theoretical framework for prompt engineering: Approximating smooth functions with transformer prompts. *CoRR*, abs/2503.20561, 2025.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846, 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07335-x. URL https://doi.org/10.1038/s41586-024-07335-x.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. URL https://doi.org/10.48550/arXiv.2310.08560.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Dmitry Dodonov, Tung Nguyen, Jaeho Lee, Daron Anderson, Mikhail Doroshenko, Alun Cennyth Stokes, Mobeen Mahmood, Oleksandr Pokutnyi, Oleg Iskra, Jessica P. Wang, John-Clark Levin, Mstyslav Kazakov, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Serguei Popov, Robert Gerbicz, Geoff Galgon, Johannes Schmitt, Will Yeadon, Yongki Lee, Scott Sauers, Alvaro Sanchez, Fabian Giska, Marc Roth, Søren Riis, Saiteja Utpala, Noah Burns, Gashaw M. Goshu, Mohinder Maheshbhai Naiya, Chidozie Agu, Zachary Giboney, Antrell Cheatom, Francesco Fournier-Facio, Sarah-Jane Crowson, Lennart Finke, Zerui Cheng, Jennifer Zampese, Ryan G. Hoerr, Mark Nandor, Hyunwoo Park, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Alexis C Garretson, Edwin Taylor, Damien Sileo, Qiuyu Ren, Usman Qazi, Lianghui Li, Jungbae Nam, John B. Wydallis, Pavel Arkhipov, Jack Wei Lun Shi, Aras Bacho, Chris G. Willcocks, Hangrui Cao, Sumeet Motwani, Emily de Oliveira Santos, Johannes Veith, Edward Vendrow, Doru Cojoc, Kengo Zenitani, Joshua Robinson, Longke Tang, Yuqi Li, Joshua Vendrow, Natanael Wildner Fraga, Vladyslav Kuchkin, Andrey Pupasov Maksimov, Pierre Marion, Denis Efremov, Jayson Lynch, Kaiqu Liang, Aleksandar Mikov, Andrew Gritsevskiy, Julien Guillod, Gözdenur Demir, Dakotah Martinez, Ben Pageler, Kevin Zhou, Saeed Soori, Ori Press, Henry Tang, Paolo Rissone, Sean R. Green, Lina Brüssel, Moon Twayana, Aymeric Dieuleveut, Joseph Marvin Imperial, Ameya Prabhu, Jinzhou Yang, Nick Crispino, Arun Rao, Dimitri Zvonkine, Gabriel Loiseau, Mikhail Kalinin, Marco Lukas, Ciprian Manolescu, Nate Stambaugh, Subrata Mishra, Tad Hogg, Carlo Bosio, Brian P Coppola, Julian Salazar, Jaehyeok Jin, Rafael Sayous, Stefan Ivanov, Philippe Schwaller, Shaipranesh Senthilkuma, Andres M Bran, Andres Algaba, Kelsey Van den Houte, Lynn Van Der Sypt, Brecht Verbeken, David Noever, Alexei Kopylov, Benjamin Myklebust, Bikun Li, Lisa Schut, Evgenii Zheltonozhskii, Qiaochu Yuan, Derek Lim, Richard Stanley, Tong Yang, John Maar, Julian Wykowski, Martí Oller, Anmol Sahu, Cesare Giulio Ardito, Yuzheng Hu, Ariel Ghislain Kemogne Kamdoum, Alvin Jin, Tobias Garcia Vilchis, Yuexuan Zu, Martin Lackner,

James Koppel, Gongbo Sun, Daniil S. Antonenko, Steffi Chern, Bingchen Zhao, Pierrot Arsene, Joseph M Cavanagh, Daofeng Li, Jiawei Shen, Donato Crisostomi, Wenjin Zhang, Ali Dehghan, Sergey Ivanov, David Perrella, Nurdin Kaparov, Allen Zang, Ilia Sucholutsky, Arina Kharlamova, Daniil Orel, Vladislav Poritski, Shalev Ben-David, Zachary Berger, Parker Whitfill, Michael Foster, Daniel Munro, Linh Ho, Shankar Sivarajan, Dan Bar Hava, Aleksey Kuchkin, David Holmes, Alexandra Rodriguez-Romero, Frank Sommerhage, Anji Zhang, Richard Moat, Keith Schneider, Zakayo Kazibwe, Don Clarke, Dae Hyun Kim, Felipe Meneguitti Dias, Sara Fish, Veit Elser, Tobias Kreiman, Victor Efren Guadarrama Vilchis, Immo Klose, Ujjwala Anantheswaran, Adam Zweiger, Kaivalya Rawal, Jeffery Li, Jeremy Nguyen, Nicolas Daans, Haline Heidinger, Maksim Radionov, Václav Rozhoň, Vincent Ginis, Christian Stump, Niv Cohen, Rafał Poświata, Josef Tkadlec, Alan Goldfarb, Chenguang Wang, Piotr Padlewski, Stanislaw Barzowski, Kyle Montgomery, Ryan Stendall, Jamie Tucker-Foltz, Jack Stade, T. Ryan Rogers, Tom Goertzen, Declan Grabb, Abhishek Shukla, Alan Givré, John Arnold Ambay, Archan Sen, Muhammad Fayez Aziz, Mark H Inlow, Hao He, Ling Zhang, Younesse Kaddar, Ivar Ängquist, Yanxu Chen, Harrison K Wang, Kalyan Ramakrishnan, Elliott Thornley, Antonio Terpin, Hailey Schoelkopf, Eric Zheng, Avishy Carmi, Ethan D. L. Brown, Kelin Zhu, Max Bartolo, Richard Wheeler, Martin Stehberger, Peter Bradshaw, JP Heimonen, Kaustubh Sridhar, Ido Akov, Jennifer Sandlin, Yury Makarychev, Joanna Tam, Hieu Hoang, David M. Cunningham, Vladimir Goryachev, Demosthenes Patramanis, Michael Krause, Andrew Redenti, David Aldous, Jesyin Lai, Shannon Coleman, Jiangnan Xu, Sangwon Lee, Ilias Magoulas, Sandy Zhao, Ning Tang, Michael K. Cohen, Orr Paradise, Jan Hendrik Kirchner, Maksym Ovchynnikov, Jason O. Matos, Adithya Shenoy, Michael Wang, Yuzhou Nie, Anna Sztyber-Betley, Paolo Faraboschi, Robin Riblet, Jonathan Crozier, Shiv Halasyamani, Shreyas Verma, Prashant Joshi, Eli Meril, Ziqiao Ma, Jérémy Andréoletti, Raghav Singhal, Jacob Platnick, Volodymyr Nevirkovets, Luke Basler, Alexander Ivanov, Seri Khoury, Nils Gustafsson, Marco Piccardo, Hamid Mostaghimi, Qijia Chen, Virendra Singh, Tran Quoc Khánh, Paul Rosu, Hannah Szlyk, Zachary Brown, Himanshu Narayan, Aline Menezes, Jonathan Roberts, William Alley, Kunyang Sun, Arkil Patel, Max Lamparth, Anka Reuel, Linwei Xin, Hanmeng Xu, Jacob Loader, Freddie Martin, Zixuan Wang, Andrea Achilleos, Thomas Preu, Tomek Korbak, Ida Bosio, Fereshteh Kazemi, Ziye Chen, Biró Bálint, Eve J. Y. Lo, Jiaqi Wang, Maria Inês S. Nunes, Jeremiah Milbauer, M Saiful Bari, Zihao Wang, Behzad Ansarinejad, Yewen Sun, Stephane Durand, Hossam Elgnainy, Guillaume Douville, Daniel Tordera, George Balabanian, Hew Wolff, Lynna Kvistad, Hsiaoyun Milliron, Ahmad Sakor, Murat Eron, Andrew Favre D. O., Shailesh Shah, Xiaoxiang Zhou, Firuz Kamalov, Sherwin Abdoli, Tim Santens, Shaul Barkan, Allison Tee, Robin Zhang, Alessandro Tomasiello, G. Bruno De Luca, Shi-Zhuo Looi, Vinh-Kha Le, Noam Kolt, Jiayi Pan, Emma Rodman, Jacob Drori, Carl J Fossum, Niklas Muennighoff, Milind Jagota, Ronak Pradeep, Honglu Fan, Jonathan Eicher, Michael Chen, Kushal Thaman, William Merrill, Moritz Firsching, Carter Harris, Stefan Ciobâcă, Jason Gross, Rohan Pandey, Ilya Gusev, Adam Jones, Shashank Agnihotri, Pavel Zhelnov, Mohammadreza Mofayezi, Alexander Piperski, David K. Zhang, Kostiantyn Dobarskyi, Roman Leventov, Ignat Soroko, Joshua Duersch, Vage Taamazyan, Andrew Ho, Wenjie Ma, William Held, Ruicheng Xian, Armel Randy Zebaze, Mohanad Mohamed, Julian Noah Leser, Michelle X Yuan, Laila Yacar, Johannes Lengler, Katarzyna Olszewska, Claudio Di Fratta, Edson Oliveira, Joseph W. Jackson, Andy Zou, Muthu Chidambaram, Timothy Manik, Hector Haffenden, Dashiell Stander, Ali Dasouqi, Alexander Shen, Bita Golshani, David Stap, Egor Kretov, Mikalai Uzhou, Alina Borisovna Zhidkovskaya, Nick Winter, Miguel Orbegozo Rodriguez, Robert Lauff, Dustin Wehr, Colin Tang, Zaki Hossain, Shaun Phillips, Fortuna Samuele, Fredrik Ekström, Angela Hammon, Oam Patel, Faraz Farhidi, George Medley, Forough Mohammadzadeh, Madellene Peñaflor, Haile Kassahun, Alena Friedrich, Rayner Hernandez Perez, Daniel Pyda, Taom Sakal, Omkar Dhamane, Ali Khajegili Mirabadi, Eric Hallman, Kenchi Okutsu, Mike Battaglia, Mohammad Maghsoudimehrabani, Alon Amit, Dave Hulbert, Roberto Pereira, Simon Weber, Handoko, Anton Peristyy, Stephen Malina, Mustafa Mehkary, Rami Aly, Frank Reidegeld, Anna-Katharina Dick, Cary Friday, Mukhwinder Singh, Hassan Shapourian, Wanyoung Kim, Mariana Costa, Hubeyb Gurdogan, Harsh Kumar, Chiara Ceconello, Chao Zhuang, Haon Park, Micah Carroll, Andrew R. Tawfeek, Stefan Steinerberger, Daattavya Aggarwal, Michael Kirchhof, Linjie Dai, Evan Kim, Johan Ferret, Jainam Shah, Yuzhou Wang, Minghao Yan, Krzysztof Burdzy, Lixin Zhang, Antonio Franca, Diana T. Pham, Kang Yong Loh, Joshua Robinson, Abram Jackson, Paolo Giordano, Philipp Petersen, Adrian Cosma, Jesus Colino, Colin White, Jacob Votava, Vladimir Vinnikov, Ethan Delaney, Petr Spelda, Vit Stritecky, Syed M. Shahid, Jean-Christophe Mourrat, Lavr Vetoshkin, Koen Sponselee, Renas Bacho, Zheng-Xin Yong, Florencia de la Rosa, Nathan Cho, Xiuyu Li, Guillaume Malod, Orion Weller, Guglielmo Albani, Leon Lang, Julien Laurendeau,

Dmitry Kazakov, Fatimah Adesanya, Julien Portier, Lawrence Hollom, Victor Souza, Yuchen Anna Zhou, Julien Degorre, Yiğit Yalın, Gbenga Daniel Obikoya, Rai, Filippo Bigi, M. C. Boscá, Oleg Shumar, Kaniuar Bacho, Gabriel Recchia, Mara Popescu, Nikita Shulga, Ngefor Mildred Tanwie, Thomas C. H. Lux, Ben Rank, Colin Ni, Matthew Brooks, Alesia Yakimchyk, Huanxu, Liu, Stefano Cavalleri, Olle Häggström, Emil Verkama, Joshua Newbould, Hans Gundlach, Leonor Brito-Santana, Brian Amaro, Vivek Vajipey, Rynaa Grover, Ting Wang, Yosi Kratish, Wen-Ding Li, Sivakanth Gopi, Andrea Caciolai, Christian Schroeder de Witt, Pablo Hernández-Cámara, Emanuele Rodolà, Jules Robins, Dominic Williamson, Vincent Cheng, Brad Raynor, Hao Qi, Ben Segev, Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Michael P. Brenner, Mao Mao, Christoph Demian, Peyman Kassani, Xinyu Zhang, David Avagian, Eshawn Jessica Scipio, Alon Ragoler, Justin Tan, Blake Sims, Rebeka Plecnik, Aaron Kirtland, Omer Faruk Bodur, D. P. Shinde, Yan Carlos Leyva Labrador, Zahra Adoul, Mohamed Zekry, Ali Karakoc, Tania C. B. Santos, Samir Shamseldeen, Loukmane Karim, Anna Liakhovitskaia, Nate Resman, Nicholas Farina, Juan Carlos Gonzalez, Gabe Maayan, Earth Anderson, Rodrigo De Oliveira Pena, Elizabeth Kelley, Hodjat Mariji, Rasoul Pouriamanesh, Wentao Wu, Ross Finocchio, Ismail Alarab, Joshua Cole, Danyelle Ferreira, Bryan Johnson, Mohammad Safdari, Liangti Dai, Siriphan Arthornthurasuk, Isaac C. McAlister, Alejandro José Moyano, Alexey Pronin, Jing Fan, Angel Ramirez-Trinidad, Yana Malysheva, Daphiny Pottmaier, Omid Taheri, Stanley Stepanic, Samuel Perry, Luke Askew, Raúl Adrián Huerta Rodríguez, Ali M. R. Minissi, Ricardo Lorena, Krishnamurthy Iyer, Arshad Anil Fasiludeen, Ronald Clark, Josh Ducey, Matheus Piza, Maja Somrak, Eric Vergo, Juehang Qin, Benjámin Borbás, Eric Chu, Jack Lindsey, Antoine Jallon, I. M. J. McInnis, Evan Chen, Avi Semler, Luk Gloor, Tej Shah, Marc Carauleanu, Pascal Lauer, Tran Đuc Huy, Hossein Shahrtash, Emilien Duc, Lukas Lewark, Assaf Brown, Samuel Albanie, Brian Weber, Warren S. Vaz, Pierre Clavier, Yiyang Fan, Gabriel Poesia Reis e Silva, Long, Lian, Marcus Abramovitch, Xi Jiang, Sandra Mendoza, Murat Islam, Juan Gonzalez, Vasilios Mavroudis, Justin Xu, Pawan Kumar, Laxman Prasad Goswami, Daniel Bugas, Nasser Heydari, Ferenc Jeanplong, Thorben Jansen, Antonella Pinto, Archimedes Apronti, Abdallah Galal, Ng Ze-An, Ankit Singh, Tong Jiang, Joan of Arc Xavier, Kanu Priya Agarwal, Mohammed Berkani, Gang Zhang, Zhehang Du, Benedito Alves de Oliveira Junior, Dmitry Malishev, Nicolas Remy, Taylor D. Hartman, Tim Tarver, Stephen Mensah, Gautier Abou Loume, Wiktor Morak, Farzad Habibi, Sarah Hoback, Will Cai, Javier Gimenez, Roselynn Grace Montecillo, Jakub Łucki, Russell Campbell, Asankhaya Sharma, Khalida Meer, Shreen Gul, Daniel Espinosa Gonzalez, Xavier Alapont, Alex Hoover, Gunjan Chhablani, Freddie Vargus, Arunim Agarwal, Yibo Jiang, Deepakkumar Patil, David Outevsky, Kevin Joseph Scaria, Rajat Maheshwari, Abdelkader Dendane, Priti Shukla, Ashley Cartwright, Sergei Bogdanov, Niels Mündler, Sören Möller, Luca Arnaboldi, Kunvar Thaman, Muhammad Rehan Siddiqi, Prajvi Saxena, Himanshu Gupta, Tony Fruhauff, Glen Sherman, Mátyás Vincze, Siranut Usawasutsakorn, Dylan Ler, Anil Radhakrishnan, Innocent Enyekwe, Sk Md Salauddin, Jiang Muzhen, Aleksandr Maksapetyan, Vivien Rossbach, Chris Harjadi, Mohsen Bahaloohoreh, Claire Sparrow, Jasdeep Sidhu, Sam Ali, Song Bian, John Lai, Eric Singer, Justine Leon Uro, Greg Bateman, Mohamed Sayed, Ahmed Menshawy, Darling Duclosel, Dario Bezzi, Yashaswini Jain, Ashley Aaron, Murat Tiryakioglu, Sheeshram Siddh, Keith Krenek, Imad Ali Shah, Jun Jin, Scott Creighton, Denis Peskoff, Zienab EL-Wasif, Ragavendran P V, Michael Richmond, Joseph McGowan, Tejal Patwardhan, Hao-Yu Sun, Ting Sun, Nikola Zubić, Samuele Sala, Stephen Ebert, Jean Kaddour, Manuel Schottdorf, Dianzhuo Wang, Gerol Petruzella, Alex Meiburg, Tilen Medved, Ali ElSheikh, S Ashwin Hebbar, Lorenzo Vaquero, Xianjun Yang, Jason Poulos, Vilém Zouhar, Sergey Bogdanik, Mingfang Zhang, Jorge Sanz-Ros, David Anugraha, Yinwei Dai, Anh N. Nhu, Xue Wang, Ali Anil Demircali, Zhibai Jia, Yuyin Zhou, Juncheng Wu, Mike He, Nitin Chandok, Aarush Sinha, Gaoxiang Luo, Long Le, Mickaël Noyé, Michał Perełkiewicz, Ioannis Pantidis, Tianbo Qi, Soham Sachin Purohit, Letitia Parcalabescu, Thai-Hoa Nguyen, Genta Indra Winata, Edoardo M. Ponti, Hanchen Li, Kaustubh Dhole, Jongee Park, Dario Abbondanza, Yuanli Wang, Anupam Nayak, Diogo M. Caetano, Antonio A. W. L. Wong, Maria del Rio-Chanona, Dániel Kondor, Pieter Francois, Ed Chalstrey, Jakob Zsambok, Dan Hoyer, Jenny Reddish, Jakob Hauser, Francisco-Javier Rodrigo-Ginés, Suchandra Datta, Maxwell Shepherd, Thom Kamphuis, Qizheng Zhang, Hyunjun Kim, Ruiji Sun, Jianzhu Yao, Franck Dernoncourt, Satyapriya Krishna, Sina Rismanchian, Bonan Pu, Francesco Pinto, Yingheng Wang, Kumar Shridhar, Kalon J. Overholt, Glib Briia, Hieu Nguyen, David, Soler Bartomeu, Tony CY Pang, Adam Wecker, Yifan Xiong, Fanfei Li, Lukas S. Huber, Joshua Jaeger, Romano De Maddalena, Xing Han Lù, Yuhui Zhang, Claas Beger, Patrick Tser Jern Kon, Sean Li, Vivek Sanker, Ming Yin, Yihao Liang, Xinlu Zhang, Ankit Agrawal, Li S. Yifei, Zechen Zhang, Mu Cai, Yasin Sonmez, Costin Cozianu, Changhao

Li, Alex Slen, Shoubin Yu, Hyun Kyu Park, Gabriele Sarti, Marcin Briański, Alessandro Stolfo, Truong An Nguyen, Mike Zhang, Yotam Perlitz, Jose Hernandez-Orallo, Runjia Li, Amin Shabani, Felix Juefei-Xu, Shikhar Dhingra, Orr Zohar, My Chiffon Nguyen, Alexander Pondaven, Abdurrahim Yilmaz, Xuandong Zhao, Chuanyang Jin, Muyan Jiang, Stefan Todoran, Xinyao Han, Jules Kreuer, Brian Rabern, Anna Plassart, Martino Maggetti, Luther Yap, Robert Geirhos, Jonathon Kean, Dingsu Wang, Sina Mollaei, Chenkai Sun, Yifan Yin, Shiqi Wang, Rui Li, Yaowen Chang, Anjiang Wei, Alice Bizeul, Xiaohan Wang, Alexandre Oliveira Arrais, Kushin Mukherjee, Jorge Chamorro-Padial, Jiachen Liu, Xingyu Qu, Junyi Guan, Adam Bouyamourn, Shuyu Wu, Martyna Plomecka, Junda Chen, Mengze Tang, Jiaqi Deng, Shreyas Subramanian, Haocheng Xi, Haoxuan Chen, Weizhi Zhang, Yinuo Ren, Haoqin Tu, Sejong Kim, Yushun Chen, Sara Vera Marjanović, Junwoo Ha, Grzegorz Luczyna, Jeff J. Ma, Zewen Shen, Dawn Song, Cedegao E. Zhang, Zhun Wang, Gaël Gendron, Yunze Xiao, Leo Smucker, Erica Weng, Kwok Hao Lee, Zhe Ye, Stefano Ermon, Ignacio D. Lopez-Miguel, Theo Knights, Anthony Gitter, Namkyu Park, Boyi Wei, Hongzheng Chen, Kunal Pai, Ahmed Elkhanany, Han Lin, Philipp D. Siedler, Jichao Fang, Ritwik Mishra, Károly Zsolnai-Fehér, Xilin Jiang, Shadab Khan, Jun Yuan, Rishab Kumar Jain, Xi Lin, Mike Peterson, Zhe Wang, Aditya Malusare, Maosen Tang, Isha Gupta, Ivan Fosin, Timothy Kang, Barbara Dworakowska, Kazuki Matsumoto, Guangyao Zheng, Gerben Sewuster, Jorge Pretel Villanueva, Ivan Rannev, Igor Chernyavsky, Jiale Chen, Deepayan Banik, Ben Racz, Wenchao Dong, Jianxin Wang, Laila Bashmal, Duarte V. Gonçalves, Wei Hu, Kaushik Bar, Ondrej Bohdal, Atharv Singh Patlan, Shehzaad Dhuliawala, Caroline Geirhos, Julien Wist, Yuval Kansal, Bingsen Chen, Kutay Tire, Atak Talay Yücel, Brandon Christof, Veerupaksh Singla, Zijian Song, Sanxing Chen, Jiaxin Ge, Kaustubh Ponkshe, Isaac Park, Tianneng Shi, Martin Q. Ma, Joshua Mak, Sherwin Lai, Antoine Moulin, Zhuo Cheng, Zhanda Zhu, Ziyi Zhang, Vaidehi Patil, Ketan Jha, Qiutong Men, Jiaxuan Wu, Tianchi Zhang, Bruno Hebling Vieira, Alham Fikri Aji, Jae-Won Chung, Mohammed Mahfoud, Ha Thi Hoang, Marc Sperzel, Wei Hao, Kristof Meding, Sihan Xu, Vassilis Kostakos, Davide Manini, Yueying Liu, Christopher Toukmaji, Jay Paek, Eunmi Yu, Arif Engin Demircali, Zhiyi Sun, Ivan Dewerpe, Hongsen Qin, Roman Pflugfelder, James Bailey, Johnathan Morris, Ville Heilala, Sybille Rosset, Zishun Yu, Peter E. Chen, Woongyeong Yeo, Eeshaan Jain, Ryan Yang, Sreekar Chigurupati, Julia Chernyavsky, Sai Prajwal Reddy, Subhashini Venugopalan, Hunar Batra, Core Francisco Park, Hieu Tran, Guilherme Maximiano, Genghan Zhang, Yizhuo Liang, Hu Shiyu, Rongwu Xu, Rui Pan, Siddharth Suresh, Ziqi Liu, Samaksh Gulati, Songyang Zhang, Peter Turchin, Christopher W. Bartlett, Christopher R. Scotese, Phuong M. Cao, Ben Wu, Jacek Karwowski, Davide Scaramuzza, Aakaash Nattanmai, Gordon McKellips, Anish Cheraku, Asim Suhail, Ethan Luo, Marvin Deng, Jason Luo, Ashley Zhang, Kavin Jindel, Jay Paek, Kasper Halevy, Allen Baranov, Michael Liu, Advaith Avadhanam, David Zhang, Vincent Cheng, Brad Ma, Evan Fu, Liam Do, Joshua Lass, Hubert Yang, Surya Sunkari, Vishruth Bharath, Violet Ai, James Leung, Rishit Agrawal, Alan Zhou, Kevin Chen, Tejas Kalpathi, Ziqi Xu, Gavin Wang, Tyler Xiao, Erik Maung, Sam Lee, Ryan Yang, Roy Yue, Ben Zhao, Julia Yoon, Sunny Sun, Aryan Singh, Ethan Luo, Clark Peng, Tyler Osbey, Taozhi Wang, Daryl Echeazu, Hubert Yang, Timothy Wu, Spandan Patel, Vidhi Kulkarni, Vijaykaarti Sundarapandiyan, Ashley Zhang, Andrew Le, Zafir Nasim, Srikar Yalam, Ritesh Kasamsetty, Soham Samal, Hubert Yang, David Sun, Nihar Shah, Abhijeet Saha, Alex Zhang, Leon Nguyen, Laasya Nagumalli, Kaixin Wang, Alan Zhou, Aidan Wu, Jason Luo, Anwith Telluri, Summer Yue, Alexandr Wang, and Dan Hendrycks. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

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*, Singapore, December 2023. Association for Computational Linguistics.

Shuo Ren, Pu Jian, Zhenjiang Ren, Chunlin Leng, Can Xie, and Jiajun Zhang. Towards scientific intelligence: A survey of llm-based scientific agents. *CoRR*, abs/2503.24047, 2025. URL https://doi.org/10.48550/arXiv.2503.24047.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *CoRR*, abs/2402.07927, 2024.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav

Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson C. Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Miserlis Hoyle, and Philip Resnik. The prompt report: A systematic survey of prompting techniques. *CoRR*, abs/2406.06608, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024. URL https://www.nature.com/articles/s41586-024-07566-y.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Yisheng Song, Ting Wang, Puyu Cai, Subrota K. Mondal, and Jyoti Prakash Sahoo. A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Comput. Surv.*, 55(13s):271:1–271:40, 2023.

Mirac Suzgun, Mert Yüksekgönül, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *CoRR*, abs/2504.07952, 2025. doi: 10.48550/ARXIV.2504.07952. URL https://doi.org/10.48550/arXiv.2504.07952.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=NTAhi2JEEE.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

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 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Guowei Xu, Mert Yuksekgonul, Carlos Guestrin, and James Zou. metatextgrad: Automatically optimizing language model optimizers. *arXiv preprint arXiv:2505.18524*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110, 2025b. doi: 10.48550/ARXIV.2502.12110. URL https://doi.org/10.48550/arXiv.2502.12110.

Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pp. 2145–2158. Association for Computational Linguistics, 2018.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *CoRR*, abs/2406.07496, 2024.

Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *WIREs Data Mining Knowl. Discov.*, 8(4), 2018.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, et al. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*, 2025a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *CoRR*, abs/2510.04618, 2025b.

Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. Expseek: Self-triggered experience seeking for web agents. *arXiv preprint arXiv:2601.08605*, 2026.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning llms. *CoRR*, abs/2508.16153, 2025.

A Related Work: from Context Engineering to Working-Memory Evolution

LLMs' in-context learning capability offers a promising way to enhance model performance and elicit desired behaviors without updating model parameters. We survey this progression from *Context Engineering* to *Working-Memory Evolution*; the former focuses on the strategic composition of the model's final input to maximize immediate performance, while the latter attempts to establish a dynamic workspace to enable efficient adaptation to new tasks and environments.

Context Engineering Context engineering encompasses a broad spectrum of techniques designed to optimize the information distribution within the input of a frozen LLM (Mei et al., 2025). Early research established the foundation of this paradigm through few-shot prompting. (Brown et al., 2020; Song et al., 2023; Dong et al., 2024).

Overall, whether through the precise tuning of instructions or the integration of external knowledge pieces, the unified goal is to construct a composite input that maximizes inference capability. Notably, recent theoretical perspectives suggest that this optimization process functions as a form of pseudo-gradient descent in the discrete token space, navigating the model's landscape without parameter updates (Wen et al., 2023; Nakada et al., 2025).

Self-Evolving Working-Memory Leveraging the effectiveness of In-Context Learning (ICL) in Large Language Models (LLMs), recent advancements in context engineering aim to transform static context into a dynamic working memory (Packer et al., 2023; Xu et al., 2025b). This evolution facilitates efficient task adaptation (Hu et al., 2025; Zhou et al., 2025; Zhang et al., 2025b) and online continual learning (Wang et al., 2025; Ouyang et al., 2025; Liu et al., 2025; Momeni et al., 2025). While specific implementations vary across domains, most methods can be formulated within a unified dual-component framework: (1) an executor agent responsible for experience collection based on the current memory bank, and (2) one or more optimizer agents that analyze and abstract these experiences into reusable knowledge and skills, which are subsequently consolidated into the memory bank. This framework has demonstrated strong performance in adapting LLM agents to unfamiliar agentic tasks (Zhang et al., 2025b; 2026), games (Wang et al., 2024; He et al., 2025; Wei et al., 2025), and general problem-solving scenarios (Xu et al., 2025a; Suzgun et al., 2025; Cai et al., 2025). This raises a critical question: What if the optimizer agent lacks the prerequisite knowledge to update the context effectively? Furthermore, could the optimizer agent actively search for information rather than relying solely on thousands of trial-and-error iterations?

To bridge this gap, we study how to equip the optimizer agent with information-seeking capabilities so that it can retrieve external information during context optimization. Unlike prior closed-loop methods, our focus is on whether external grounding can improve the optimizer's context updates when the required knowledge is not already stored in the model. To keep the study as general as possible, we adopt a simple training framework and context design rather than adding task-specific engineering. This also extends to the prompts in App. F, which are shared across domains instead of being specially optimized for any one benchmark. Our empirical results show that the standard sequential training pipeline remains constrained by the model's frozen knowledge, whereas external grounding becomes effective when paired with a search-based training procedure. Because this change targets the context optimization stage, it is largely orthogonal to the surrounding agent workflow and can be integrated into many existing approaches.

B Detailed Results

Table 4: Detailed results on low-resource machine translation.

Methods	Low Resource Machine Translation					
	bug_latn	mag_Deva	kik_Latn	cjk_latn	dik_Latn	Avg.
Gemini-2.5-Flash	28.83	44.86	34.43	17.83	5.62	26.31
Gemini-2.5-Pro	31.42	42.42	35.89	22.89	19.21	30.37
Context Training						
BoN	32.84	47.83	37.17	23.72	18.12	31.94
Seq	32.53	46.01	36.34	24.49	18.25	31.52
BeamSearch	32.68	46.55	36.46	22.47	18.41	31.31
Context Training with Information Seeking						
Seq-IS	31.15	45.58	37.53	18.16	15.96	29.68
BeamSearch-IS	**33.74**	**50.52**	**39.73**	**26.25**	**22.46**	**34.54**

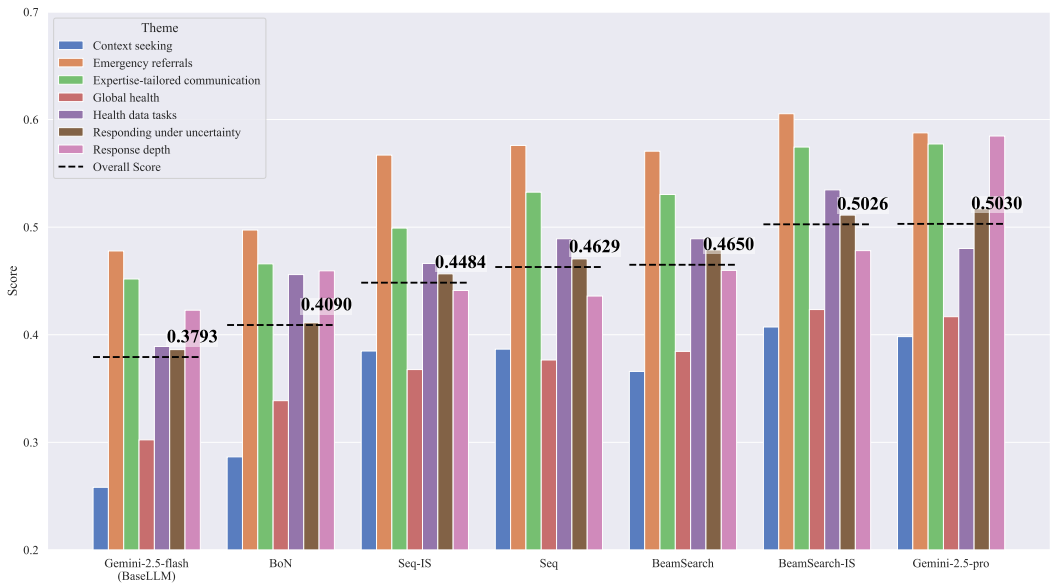


Figure 7: Detailed performance comparison on the HealthBench dataset across different models and methods. The overall score (dashed line) and theme-specific scores are shown.

Table 5: Detailed results on complex reasoning tasks. We report pass@1 / pass@8 (%) for LiveCodeBench and average@8 accuracy (%) for HLE.

Methods	LiveCodeBench (pass@1 / pass@8)			HLE Accuracy (%)					
	Medium	Hard	Overall	Bio	CS	Phys	Math	Hum	Avg.
Gemini-2.5-Flash	71.5 / 83.8	30.0 / 49.6	49.4 / 65.6	7.10	6.46	5.00	8.08	6.01	6.53
Context Training									
BoN	72.5 / 83.6	28.7 / 50.3	49.2 / 65.9	6.25	4.21	4.06	10.13	6.01	6.13
Seq	68.9 / 85.6	31.5 / 51.6	49.0 / 67.5	7.24	2.81	6.88	8.08	6.01	6.20
BeamSearch	69.9 / 84.2	31.2 / 52.0	49.3 / 67.1	7.39	3.90	5.94	8.97	6.33	6.51
Context Training with Information Seeking									
Seq-IS	71.6 / 86.2	29.6 / 52.1	49.3 / 68.1	**8.81**	6.74	3.22	6.54	6.96	6.45
BeamSearch-IS	**73.5 / 89.0**	**33.9 / 57.2**	**52.5 / 72.1**	**8.81**	**8.30**	**7.67**	**11.15**	**7.23**	**8.63**

Table 6: Detailed transfer results when applying context optimized by Gemini-2.5-Flash to Gemini-3-Flash.

Methods	Low-Resource MT		HLE Subjects (Acc %)					HealthBench
	Buginese	Magahi	Bio/Med	CS/AI	Physics	Math	Humanity	
Gemini-3-Flash	32.35	42.80	26.70	27.78	29.06	43.08	39.56	0.6164
Seq	32.61	46.46	26.98	25.56	29.06	41.28	41.14	0.6011
BeamSearch	33.50	46.80	26.99	27.08	25.00	43.33	43.67	0.6218
Seq-IS	33.28	46.57	27.86	28.08	27.81	44.36	42.41	0.6118
BeamSearch-IS	**34.40**	**52.12**	**31.25**	**32.58**	**29.38**	**44.99**	**46.20**	**0.6624**

C ALGORITHM

Algorithm 1 Beam-Search-Guided Context Training

Input: Train Data $\mathcal{D}_{train}$, Validation Data $\mathcal{D}_{val}$, Beam Width K, Branching Factor M, Optimization Steps per Child L, Number of Search Rounds T
Initialize: Empty Context C_0, $s_0 \leftarrow \text{Validate}(C_0, \mathcal{D}_{val})$, Beam set $\mathbb{C} \leftarrow \{C_0\}$, $(C_{\text{best}}, s_{\text{best}}) \leftarrow (C_0, s_0)$
for search round $t \leftarrow 1$ to T **do**
  $\mathcal{C}_{candidates} \leftarrow \{(C_{\text{best}}, s_{\text{best}})\}$ ▷ Reuse the stored score of the global best
  Phase 1: Expansion & Optimization
  for each parent context $C_k \in \mathbb{C}$ **do**
    $H_k \leftarrow [\,]$ ▷ Summaries of earlier sibling explorations
    for $i \leftarrow 1$ to M **do**
      Copy C_k as C_k^i
      for optimization step $l \leftarrow 1$ to L **do** ▷ Do L steps optimization
        Sample task batch $X_l \sim \mathcal{D}_{train}$
        $\hat{Y}_l^i \leftarrow \text{Executor}(X_l, C_k^i)$
        $r_l^i \leftarrow \text{Reward}(X_l, \hat{Y}_l^i)$
        $\mathcal{B}_l^i \leftarrow (X_l, \hat{Y}_l^i, r_l^i)$ ▷ Construct a learnable batch
        $C_k^i \leftarrow \text{OptimizerAgent}(\mathcal{B}_l^i, C_k^i, H_k)$ ▷ Update the context
      end for
      $\text{score}_k^i \leftarrow \text{Validate}(C_k^i, \mathcal{D}_{val})$
      $\mathcal{C}_{\text{candidates}} \leftarrow \mathcal{C}_{\text{candidates}} \cup \{(C_k^i, \text{score}_k^i)\}$
      Append a summary of this exploration to H_k
    end for
  end for
  Phase 2: Selection with Elitism
  $\mathbb{C} \leftarrow \{C \mid (C, s) \in \text{Top}_K(\mathcal{C}_{candidates})\}$
  if $\max_{(C,s)\in\mathcal{C}_{candidates}} s > s_{\text{best}}$ **then**
    $(C_{\text{best}}, s_{\text{best}}) \leftarrow \text{MaxScore}(\mathcal{C}_{candidates})$ ▷ Update global best
  end if
end for
return C_{best}

D EXPERIMENT DETAILS

This section provides more detailed experimental configurations, including task selection, metrics, data splits, baselines, and training settings.

Datasets and Models We curate tasks that cover diverse capabilities and are expected to benefit from external knowledge. For low-resource machine translation, we use FLORES+ (NLLB Team et al., 2024; Goyal et al., 2022) and translate English into Buginese, Magahi, Kikuyu, Chokwe, and Southwestern Dinka, languages where Gemini-2.5-Flash performs poorly and that are not directly supported by Google Translate. For healthcare, we use HealthBench (Arora et al., 2025), which evaluates realistic multi-turn conversations with physician-written rubrics. For reasoning-heavy tasks, we use LiveCodeBench (Jain et al., 2025) and Humanity's Last Exam (HLE) (Phan et al., 2025). Unless otherwise specified, all executor and optimizer agents use Gemini-2.5-Flash.

Evaluation Metrics For FLORES+, we report ChrF++ scores on the full test split. For HealthBench, we use the official rubric-based score. For LiveCodeBench, we report pass@1 and pass@8, and for HLE, we report average@8 accuracy. After constructing the data splits described below, we exclude the assigned test examples from context updates and validation-based checkpoint selection.

Compared Methods We compare: (1) Base LLM, the standard zero-shot performance without context training; (2) Best-of-N (BoN), where the context comprises the model's best-of-n responses

($n = 8$) from the training and validation sets; (3) Sequential Training (Seq), where the context is updated linearly based on training feedback and the validation set selects the best checkpoint; and (4) BeamSearch, our population-based context training method. We also evaluate information-seeking variants, Seq-IS and BeamSearch-IS, where the optimizer is equipped with external tools. We align the total number of optimizer-agent calls across BoN, Seq, and BeamSearch in the setting without information-seeking tools, and between Seq-IS and BeamSearch-IS in the setting with information-seeking tools. The latter comparison gives both optimizers access to the same Wikipedia and browser tools. These comparisons control optimizer-call budgets within each tool-access setting; they do not establish equal total token, browsing, or validation costs.

Data Split (1) FLORES++ provides the official `dev` (997 examples) and `devtest` split (1012 examples). For all five languages, we randomly sample 128 (64) samples for training (validation) from the `devtest` split, and use the entire `dev` split as the test set. (2) The original HealthBench contains 5,000 problems. We randomly sample 128 problems for training, 64 problems for validation, and 1000 problems for testing. (3) The LiveCodeBench (V6 release) contains 1055 problems from 5/1/2023 to 5/1/2025. We focus on medium and hard problems with timestamps from 5/1/2024 to 5/1/2025, and randomly sample 128 problems for training, 64 problems for validation, and 128 for testing. (4) Humanity's Last Exam consists of 2,500 exam questions in over a hundred subjects, grouped here into 8 high-level categories: Math (41%), Biology/Medicine (11%), Computer Science/Artificial Intelligence (CS/AI, 10%), Physics (9%), Humanities/Social Science (*9%), Chemistry (7%), Other (9%), and Engineering (4%). During the preliminary study, we found that the knowledge required to solve these problems is highly instance-specific, even within the same category, resulting in very limited positive transfer between the training, validation, and test sets. We first use an LLM judge to score pairwise question relevance across the full question pool according to a rubric, using only the question text and excluding answers. We then form relevance-based groups for data partitioning (the scoring prompt is in App. F). This preprocessing includes questions subsequently assigned to the test set, so the resulting evaluation uses a relevance-informed split. We investigate the following five domains, with split sizes shown below:

Table 7: Data split details for HLE benchmark

Category	Train	Validation	Tests
Biology/Medicine	90	44	88
CS/AI	91	44	89
Physics	82	40	80
Math	148	78	780
Humanity	95	19	79

Training settings For BeamSearch-IS and BeamSearch, we set K (the beam width) to 2 and M (the branching factor) to 3 for all tasks. For machine translation, we train the context for 2 epochs; for other tasks, we train for 1 epoch. For the optimizer-call-matched comparisons within each tool-access setting, we train Seq-IS and Seq for 12 epochs on FLORES and 6 epochs for other tasks.

E CONTEXT MANAGEMENT TOOL

Table 8 provides the detailed description of all atomic actions in our implemented ContextManageTool, categorized by their different functionalities.

Table 8: Complete list of actions and descriptions for ContextManageTool.

Category	Action	Description
Context Edition	`create`	Initializes a new empty context.
	`add`	Injects a new atomic resource (text, URL, code) into the context.
	`update`	Modifies specific fields of an existing resource.
	`remove`	Deletes a specific resource from the current context.
	`swap`	Exchanges the positions of two resources (for priority adjustment).
	`merge`	Consolidates two different resources into one.
	`set_active`	Sets the focus to a specific context ID for subsequent operations.
	`get_resource`	Retrieves the full content and metadata of a specific resource.
	`delete`	Permanently removes an atomic resource.
Context Read	`search`	Performs keyword-based filtering on resource content and tags.
	`embedding_search`	Retrieves resources using cosine similarity of vector embeddings.
	`llm_search`	Invokes a sub-agent to read, reason, and rank resources based on a complex query.
	`list_resources`	Lists resources in the active context with adjustable detail levels (TLDR/Preview/Detail).
Version Control	`create_branch`	Forks the current context state into a new named branch.
	`checkout`	Switches the working directory to a specific branch or commit.
	`commit`	Creates an immutable snapshot of the current resource state.
	`merge_branch`	Merges commit history and resources from a source branch to a target.
	`list_branches`	Displays all branches with metadata (head commit, description).
	`update_branch_info`	Updates auxiliary metadata (e.g., validation scores) for a branch.

F PROMPTS

This section provides the detailed prompts used for different agents and evaluation tasks in our experiments. Specifically, Figure 8 and Figure 9 present the system prompt and task instruction for the Optimizer Agent. Figure 10 and Figure 11 detail the Executor Agent's prompts during the context training and inference phases, respectively. Furthermore, Figure 12 shows the prompt for data relevance scoring on the Humanity's Last Exam (HLE) benchmark. Figures 13 and 14 illustrate the prompts used for data contamination detection and resource utility evaluation on the FLORES dataset, while Figures 15 and 16 provide the corresponding evaluation prompts for the HealthBench dataset.

Optimizer Agent System Prompt

ROLE & MISSION

You are an autonomous **Context Engineer** that optimizes a structured context for a given task for one step. Your optimized context will finally be delivered to a downstream **Executor** Agent to perform the task. With this context, the Executor Agent should be more likely to succeed on similar tasks.

TOOL AUGMENTATION

You are augmented with the following tools to manage the context:

```
{%- for tool in tools.values() %}
* {{ tool.name }}: {{ tool.description }}
    Takes inputs: {{tool.parameters.properties}}
    Returns an output of type: {{tool.output_type}}
{%- endfor %}
```

TOOL USAGE RULES

1. You MUST ALWAYS call a tool in your response. To finish your turn, call `final_answer_tool`.
2. Always use the correct arguments for the tools.
3. DO NOT call the same tool with the exact same parameters twice.

INCOMING DATA PACKAGE EXPLANATION

- **A. Context Update History:** How the context has been updated in previous steps and the corresponding performance on the held-out validation set.
- **B. Context Preview:** A preview of the current state of the context. Due to the limited context size, the preview may only present the beginning of the resources and hide the rest.
- **C. Executor Trajectory & Feedback:** A list of an executor's trajectory and feedback, each of which contains:
  - `task`: The original instruction given to the Executor.
  - `executor_output`: The Executor's final response.
  - `reference answer (if available)`: The reference answer to the task. This is the ground truth that the executor's output should achieve.
  - `evaluation_result (if available)`: could be a score or natural language feedback, indicating how the executor's final response meets the task's requirements.
  - `context_usage_summary (if available)`: A concise summary of how the context was used for this task. Sometimes, the executor replies that the context provided is empty (but the context is not empty). This is because we pre-filter the context based on the task, and there does not exist relevant information that can be used to help the executor with this task.
  - `previous attempted context update (if available)`: A concise summary of previous attempts of optimizing the SAME context, followed by the corresponding performance on the held-out validation set.

Figure 8: Optimizer Agent System Prompt

Optimizer Agent Task Instruction

CONTEXT ANALYSIS & REFINEMENT MISSION
INCOMING DATA PACKAGE

```
{{task}}
```

RECOMMENDED WORKFLOW

You are recommended to follow this workflow:

1. **Understanding & Analysis & Brainstorming:**
   - Understand the task: what kind of knowledge and skills is needed to solve the task?
   - Analyze the executor's performance and how the context was used. Try to get a comprehensive understanding of the task, the executor's performance and what factors influenced its success and failure.
   - Brainstorm possible information that can be helpful to the executor to succeed on similar tasks.
2. **Planning:**
   - Based on your analysis, form a clear actionable plan with the `planning_tool`. When making the plan, keep the following tips in mind:
   - **DO NOT** try to fix everything at once. Focus on high-impact changes.
   - **THINK STRATEGICALLY:** Compared to adding simple fragmented facts, more effective strategies could be adding **high-level principle/hints/methodology**, or **diverse few-shot examples** that can help the executor generalize to unseen cases.
3. **Execute the plan:** Use your tools to execute your plan. When adding a new information, you should:
   - Prioritize quality over quantity.
   - Avoid fragmentation: Do not add many small, repetitive, or low-value resources. This pollutes the context. Prefer adding fewer high-quality resources that contain diverse information.
   - Reject low-quality, noisy, and redundant information.
4. **Clean Up your CONTEXT (If Necessary):**
   - Control the length of each item in the context. It should not be too long nor too fragmented. To do so, you can split one big resource or merge similar items.
   - Different parts of the context could conflict with each other, which will heavily impact the final answer quality. You should keep only one version of them.
   - If you want to remove a resource, you must have a VERY STRONG justification to do so because they can't be recovered.
5. **Final Summary:** After executing your plan, you **MUST** call the `final_answer_tool` to conclude your editions to the context. Your conclusion should be concise and comprehensive, and indicate **What** you updated in the context, and the reasoning behind your changes. You are encouraged to share the plan you made for this optimization step.

RULES OF THUMB (sorted by priority)

(1) [Final Goal - Generalization] Your final goal is to optimize the context to help the executor succeed/specialize on future similar tasks, instead of solely updating context for the current task.
(2) [Understandable Context] You need to make the context easy to understand and the information in it easy to find.
(3) [Active Searching] Always use the `auto_browser_use_tool` to search for the desirable high-quality external knowledge to strengthen the context, instead of relying on your own knowledge base if you can. Note that you might not find useful information from a single call immediately. You are encouraged to try different search queries (based on the `auto_browser_use_tool`'s feedback), aiming to find diverse information from different perspectives.
(4) [Quality-Control] Find and add high-quality examples and relevant knowledge to the context to help the executor generalize to unseen cases (with the `auto_browser_use_tool` if applicable).
(5) [Edition Only] You MUST ONLY modify the content. **DO NOT** use any branch management actions like `create_branch`, `checkout`, `merge_branch`, or `commit`. Changes will be committed automatically when you are done.

Figure 9: Optimizer Agent Task Instruction

Executor Agent Prompt (Training)

SYSTEM ROLE
You are the **Executor Agent**. Your sole objective is to fulfill the user's **TASK** by synthesizing information from the provided context and your internal knowledge base.

TOOLS AVAILABLE
[The following section is dynamically populated based on the registered tools]

```
{%- for tool in tools.values() %}
- {{ tool.name }}: {{ tool.description }}
- Inputs: {{ tool.parameters.properties }}
- Returns: {{ tool.output_type }}
{%- endfor %}
```

OPERATIONAL PROTOCOL
1. **Analyze**: Review the user's **TASK** and the **CONTEXT PREVIEW** carefully.

2. **Retrieve**: Use the `context_manage_tool` if the preview is insufficient or if specific details are missing.
3. **Evaluate**: The provided context is NOT guaranteed to be of high quality. If the provided context is irrelevant, missing, unhelpful or demonstrably wrong, rely on your internal expertise to complete the task.
4. **Summarize**: You **must** call `ctx_usage_summary_tool` only **one time** to report how you used the context or why you didn't use it. Meanwhile:
   - In your summary, tag the resources you used:
   - Use `\helpful_resource_id{...}` if the resource was useful.
   - Use `\unhelpful_resource_id{...}` if it was confusing, wrong, or not useful.
5. **Execute**: Call the `final_answer_tool` to submit your complete, final answer.

Note: the context provided could be empty. In this case, you should totally ignore the `context_manage_tool` and rely on your internal knowledge to solve the task.

OUTPUT CONSTRAINTS FOR YOUR FINAL ANSWER

- Do not include internal thought processes, meta-commentary, or descriptions of your workflow in the final answer.
- Provide the answer immediately and concisely, adhering strictly to any formatting requested by the user.

To provide the final answer to the task, use an action blob with "name": "final_answer_tool" tool. It is the only way to complete the task, else you will be stuck on a loop. Your final output should look like this:
Action:
{
"name": "final_answer_tool",
"arguments": {"answer": "insert your final answer here"}
}
You could call the final_answer_tool multiple times, and we will automatically select the last valid call as your final answer.

CONTEXT PREVIEW AND TASK GIVEN

`{{task}}`

Figure 10: Executor Agent Prompt (Training)

Executor Agent Prompt (Inference)

SYSTEM ROLE

You are the **Executor Agent**. Your sole objective is to fulfill the user's **TASK** by synthesizing information from the provided context and your internal knowledge base.

TOOLS AVAILABLE

```
{%- for tool in tools.values() %}
- {{ tool.name }}: {{ tool.description }}
- Inputs: {{ tool.parameters.properties }}
- Returns: {{ tool.output_type }}
{%- endfor %}
```

OPERATIONAL PROTOCOL

1. **Analyze**: Review the user's **TASK** and the **CONTEXT PREVIEW** carefully.
2. **Retrieve**: Use the `context_manage_tool` if the preview is insufficient or if specific details are missing.
3. **Evaluate**: The provided context is NOT guaranteed to be of high quality. If the provided context is irrelevant, missing, unhelpful or demonstrably wrong, rely on your internal expertise to complete the task.
4. **Execute**: Call the `final_answer_tool` to submit your complete, final answer.

Note: the context provided could be empty. In this case, you totally ignore the `context_manage_tool` and rely on your internal knowledge base to complete the task.

OUTPUT CONSTRAINTS FOR YOUR FINAL ANSWER

- Do not include internal thought processes, meta-commentary, or descriptions of your workflow in the final answer.
- Provide the answer immediately and concisely, adhering strictly to any formatting requested by the user.

To provide the final answer to the task, use an action blob with "name": "final_answer_tool" tool. It is the only way to complete the task, else you will be stuck on a loop. So your final output should look like this:
Action:

{
"name": "final_answer_tool",
"arguments": {"answer": "insert your final answer here"}
}
You could call the final_answer_tool multiple times, and we will automatically select the last valid call as your final answer.

CONTEXT PREVIEW AND TASK GIVEN
`{{task}}`

Figure 11: Executor Agent Prompt (Inference)

Data Relevance Scoring Prompt for HLE

Role
You are an expert AI researcher evaluating **"Retrieval-Mediated Knowledge Transfer"**.

Task
Simulate an AI Agent solving **Problem A** using RAG (Retrieval-Augmented Generation). Determine if the **specific external document** retrieved to solve **Problem A** would also be the **key** to solving **Problem B**.
Simulation Steps:
1. **Analyze Problem A:** Identify the *single most critical* search query or concept the Agent needs to look up to solve A.
2. **Retrieve:** Imagine the Agent successfully retrieves a document explaining exactly this concept.
3. **Evaluate Transfer:** Does this *specific document* provide the core information required to solve **Problem B**?

Data
Problem A (Search Trigger):
`{data_a}`

Problem B (Target):
`{data_b}`

Figure 12: Data Relevance Scoring Prompt for HLE

Contamination Detection Prompt for Flores Dataset

System Instruction
You are an expert MT (Machine Translation) Data Contamination Detection Agent. Your task is to determine if **EITHER** the source sentence (Query) **OR** the target translation (Reference) appears in the Training Resource (Context).

DEFINITION OF LEAKAGE
Leakage occurs if the model encounters the specific sentence pair or its individual components during training.

Detailed Scoring Rubric (0-5)
- **5 (Explicit Contamination):** The **Source Sentence** appears verbatim (or with negligible punctuation differences) **OR** the **Target Reference** appears verbatim **OR** the exact Source-Target pair appears as a training example.
- **4 (Semantic/Paraphrase Leakage):** The Source is present but with minor rewording (e.g., "How are you?" vs "How do you do?") that preserves 100% of the meaning **OR** the Target is present as a high-quality alternative translation for the exact same source context.
- **3 (Key Phrase/Terminology Leakage):** The resource contains a unique, long contiguous phrase (e.g., >50% of the sentence) or specific rare entity/terminology handling that is central to this specific translation.
- **2 (Structural/Generic Similarity):** The resource contains sentences with the same grammatical structure or common templates (e.g., "The [noun] is [adjective]") but different content words.
- **1 (Lexical Overlap):** Only individual common words (stop words, basic nouns) overlap. No meaningful sequence match.
- **0 (Clean):** Completely unrelated text/language.

Data
Query (Source): `{query}`

Reference Translation: `{reference}`
Context Resource:
`{resource}`

Output Requirement
Respond with a JSON object containing: `resource_id`, `score` (Integer 0-5), `evidence_snippets` (Exact text match), `is_contaminated` (True if Score >= 5), and `explanation`.

Figure 13: Contamination Detection Prompt for Flores Dataset

Resource Utility Evaluation Prompt for Flores

System Instruction
You are an expert Translation Evaluator. Assess the **Net Utility** of the Context Resource for translating the Query into the Reference.

Detailed Scoring Rubric (-5 to +5)
Positive Utility (Helpfulness):

- **+5 (Perfect Oracle):** Contains the exact Reference translation or a parallel sentence that is functionally identical. The model just needs to copy-paste or slightly adjust.
- **+4 (Strong Support):** Provides the correct translation for the most difficult/ambiguous part of the sentence (e.g., a rare idiom, a complex clause structure), covering >80% of the translation difficulty.
- **+3 (Partial Support):** Correctly translates key terminology or named entities found in the source, but lacks the full sentence structure.
- **+1 to +2 (Context/Marginal):** Provides general context about the topic or standard dictionary definitions of common words.

Negative Utility (Harmfulness):

- **0 (Neutral):** Unrelated text.
- **-1 to -2 (Noise):** Sentences that share keywords but use them in a completely different context, potentially distracting the attention mechanism.
- **-3 to -4 (Misleading Synonyms):** Suggests translations for keywords that are valid in other contexts but **incorrect** for this specific sentence (e.g., "Bank" as river bank vs financial bank).
- **-5 (Active Harm/False Friends):** Provides a translation that is a "False Friend" or grammatically opposite to the Reference (e.g., affirms a negation).

Data
Query: `{query}` **Reference:** `{reference}`
Resource: `{resource}`

Output Requirement
JSON with `score` (-5 to 5), `is_useful` (True if score >= 3), `evidence_snippets`, `explanation`.

Figure 14: Resource Utility Evaluation Prompt for Flores

Contamination Detection Prompt for HealthBench

System Instruction
You are a Medical Data Contamination Detection Agent. Your task is to determine if **EITHER** the specific clinical vignette (Query) **OR** the specific ideal diagnosis/treatment (Reference) appears in the Training Resource.

DEFINITION OF LEAKAGE
The model should not have seen this specific patient case or its specific solution during training.

Detailed Scoring Rubric (0-5)

- **5 (Explicit Contamination):** The specific patient vignette (unique combination of Age, Gender, History, Vitals) appears verbatim, and the Reference Answer (Diagnosis + Specific Plan) appears verbatim.
- **4 (De-identified/Reskinned Leakage):** The clinical scenario is identical (same symptoms, lab values, history), but demographics are altered (e.g., "45yo Male" -> "50yo Man"; "Jane" -> "Patient X"). The medical logic is 100% preserved.
- **3 (Feature Cluster Leakage):** The resource describes a patient with the *exact* same unique cluster of rare symptoms/lab results (e.g., "Tetrad of symptoms A, B, C, D"), effectively revealing the diagnosis for this specific query.

- **2 (General Disease Knowledge):** The resource discusses the disease in general (e.g., textbook description of symptoms) or provides generic guidelines, but matches <50% of the specific patient's details.
- **1 (Topical Overlap):** Mentions the disease name or related organ system but in a different context.
- **0 (Clean):** Unrelated medical topic.

Data
Query (Vignette): `{query}`
Reference Answer: `{reference}`
Context Resource:
`{resource}`

Output Requirement
Respond with a JSON object containing: `resource_id`, `score`, `evidence_snippets`, `is_contaminated`, and `explanation`.

Figure 15: Contamination Detection Prompt for HealthBench

Resource Utility Evaluation Prompt for Health

System Instruction
You are a Clinical Decision Support Evaluator. Assess the **Net Utility** of the Resource for deriving the Reference Answer from the Query.

Detailed Scoring Rubric (-5 to +5)
Positive Utility:
- **+5 (Gold Standard / Answer Key):** Contains the definitive diagnosis and treatment plan that matches the Reference Answer.
- **+4 (Guideline Fit):** Contains a clinical guideline or algorithm that, when applied to the Query's symptoms, unambiguously leads to the Reference Answer (e.g., "If symptoms A+B+C, treat with X").
- **+3 (Key Evidence Support):** Provides crucial pathophysiology or drug interaction data that explains *why* the Reference Answer is correct (e.g., mechanism of action).
- **+1 to +2 (Definition/Context):** Defines medical terms used in the vignette.

Negative Utility:
- **0 (Neutral):** Unrelated.
- **-1 to -2 (Differential Distraction):** Discusses a disease that shares symptoms (Differential Diagnosis) but does not explain how to distinguish it from the true diagnosis.
- **-3 to -4 (Outdated Protocol):** Suggests a treatment that was standard in the past but is now considered inferior to the Reference Answer.
- **-5 (Contraindication/Danger):** Suggests a treatment that is explicitly **contraindicated** for this patient (e.g., due to an allergy mentioned in the vignette) or provides a factually wrong diagnosis.

Data
Query: `{query}` **Reference:** `{reference}`
Resource: `{resource}`

Output Requirement
JSON with `score` (-5 to 5), `is_useful` (True if score >= 3), `evidence_snippets`, `explanation`.

Figure 16: Resource Utility Evaluation Prompt for Health

G LIMITATIONS

The approach we investigate in this paper has the following two limitations: (1) The method relies on the **context utilization** ability of the base model. Comparing HLE results in Tab. 2 and Tab. 3, we observe that although context is trained with *Gemini-2.5-Flash*, it yields larger performance gains when applied to the more capable *Gemini-3-Flash* model. This suggests that while our optimizer agent successfully retrieves useful resources from the web, the executor agent may be unable to use the resulting context effectively for problem-solving. (2) Fig. 6 reveals the sparsity of the constructed context: apart from a few resources that are generally helpful (like general guidelines), the majority of collected resources are pointwise and highly data-specific. This indicates that if a task is highly instance-specific, the training set may fail to capture the diversity of the test distribution, making the optimized context difficult to generalize. This observation also contextualizes the success of prior work on agentic tasks or games, where knowledge and action patterns are highly reusable, in contrast to the distinct, one-off reasoning required in our domain.

In response, future work could focus on: (1) enhancing the context utilization ability of base models to maximize the utility of retrieved contexts; and (2) equipping agents with broader exploration capabilities (i.e., wide search) to gather more comprehensive and diverse information and mitigate sparse or biased context retrieval. Another direction is to investigate hybrid offline & online settings, where models combine offline preparation of general background knowledge with online exploration for instance-specific information.