Can a Small Model Learn to Look Before It Leaps? Dynamic Learning and Proactive Correction for Hallucination Detection

Zepeng Bao[1], Shen Zhou[1], Qiankun Pi[1], Jianhao Chen[1,2], Mayi Xu[1], Minghao Tong[1], Ming Zhong[1], Yuanyuan Zhu[1], and Tieyun Qian[1,2(✉)]

[1] School of Computer Science, Wuhan University, Wuhan, China
{zepengbao,shenzhou,piqiankun,chenjianhao,xumayi,clock,yyzhu,qty}@whu.edu.cn
[2] Zhongguancun Academy, Beijing, China

Abstract. Hallucination in large language models (LLMs) remains a critical barrier to their safe deployment. For hallucination detection to be practical in real-world scenarios, the use of efficient small models is essential to ensure low latency and minimal resource consumption. However, existing methods rely on fixed verification strategies, where simply tuning small models to mimic fixed verification trajectories fails to capture the adaptability required for diverse hallucination patterns, thereby inducing planning instability. To address this limitation, we propose a "Learning to Evaluate and Adaptively Plan" (LEAP) framework, which shifts hallucination detection from fixed execution to dynamic strategy learning. Specifically, LEAP first employs a powerful teacher model to iteratively explore and refine verification strategies through a failure-driven loop. This dynamic planning capability is then distilled into an efficient student model, augmented by a novel proactive correction mechanism that enables the model to evaluate and optimize its verification strategy before execution. Experiments on three benchmarks demonstrate that LEAP outperforms state-of-the-art methods, offering an effective and scalable solution for reliable hallucination detection.

Keywords: Hallucination detection · Factuality · Trustworthiness

1 Introduction

Hallucination undermines the reliability of large language models (LLMs) through the generation of factually incorrect or fabricated content. This issue poses severe risks in high-stakes domains such as medicine and law [38]. Therefore, equipping LLMs with the ability to discern the veracity of their own outputs via **hallucination detection** has become a critical prerequisite for safe deployment [12].

Existing detection methods typically fall into two paradigms: intrinsic self-check and tool-augmented verification. The former leverages models' internal

M. Ceci et al. (Eds.): ECML PKDD 2026, LNAI 16942, pp. 134–151, 2027.
https://doi.org/10.1007/978-3-032-37657-2_8

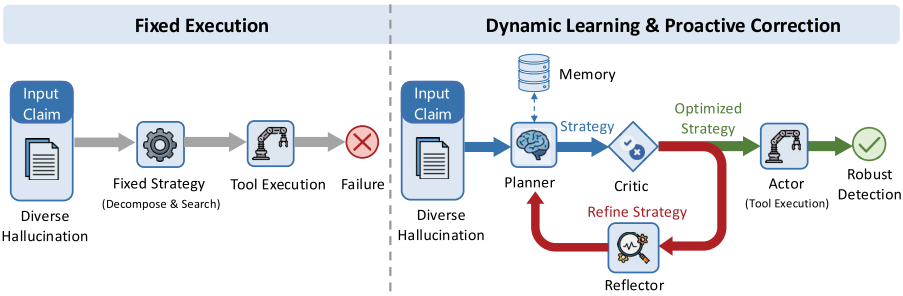

Fig. 1. Fixed strategies and dynamic strategies for hallucination detection on diverse claims.

signals like token probabilities [15,24] or patterns in activation states [1,9] for hallucination flagging. However, these methods may fail when the model is confidently wrong.

This constraint has spurred the development of tool-augmented methods [7,31], which verify claims by retrieving external evidence. For such tool-augmented detection to be practical in real-world applications, there is a growing emphasis on using efficient small models [2]. Their low latency and minimal resource requirements make them ideal for real-time monitoring and on-device deployment. However, the limited parameter scale of these models introduces a significant performance bottleneck.

To compensate for their restricted reasoning capabilities, existing frameworks typically resort to **predefined and fixed verification strategies**. As shown in Fig. 1, these methods execute the same "search-and-verify" workflow, regardless of whether a claim involves simple facts or complex causal relationships. This fixity lacks the adaptability required for diverse hallucination patterns, often leading to inappropriate tool calls and detection failures. Furthermore, even when specifically tuned [6], they suffer from planning instability due to mimicking fixed trajectories over adaptive reasoning and may generate plausible but ineffective verification plans when facing complex claims, as shown in Sect. 4.7.

To bridge this gap, we argue that simply tuning small models to mimic fixed tool-use trajectories is insufficient, as it fails to capture the underlying reasoning logic required for diverse claims. Instead, robust hallucination detection requires a paradigm shift from executing fixed process to planning adaptive dynamic strategies. However, implementing this shift presents two core challenges. First, since optimal verification paths are claim-specific and not predefined, how can we automatically construct diverse high-quality strategies? Second, given the limited capacity of efficient small models, how can we internalize this adaptive planning capability while preventing the generation of unstable plans?

To address these challenges, we propose a novel **L**earning to **E**valuate and **A**daptively **P**lan (**LEAP**) framework designed to endow efficient small models with the dynamic planning capabilities. **First**, to address the challenge of constructing diverse strategies, we establish a dynamic strategy learning loop where

a teacher model iteratively explores and refines verification trajectories based on past failures. This process populates memory with high-quality strategies that transcend the limitations of fixed workflows. **Second**, to internalize and stabilize this capability in small models, we employ agent tuning augmented by a novel proactive correction mechanism. As shown in Fig. 1, a tuned critic performs a preemptive assessment of the proposed strategy's validity before tool execution. Should the initial plan be identified as suboptimal, the reflector triggers an iterative refinement loop to synthesize an optimized strategy. This "look before it leaps" paradigm ensures that the actor executes only precise and validated strategies, thereby achieving robust hallucination detection even within the constraints of limited model parameters.

Our contributions are summarized as follows:

- We propose LEAP, a new dynamic strategy learning framework that transcends fixed verification strategies and enables small models to master diverse and adaptive strategies.
- We propose a novel proactive correction mechanism, which a tuned critic evaluates and triggers the refinement of verification strategies before execution, enhancing the robustness of strategy execution.
- Experiments on three datasets validate the superiority of LEAP over baselines based on the fixed strategy in hallucination detection.

2 Related Work

2.1 Hallucination Detection

Hallucination detection aims to assess the veracity of content from LLMs, a critical step to ensure their reliability [19]. Existing methods fall into two paradigms: intrinsic self-check and tool-augmented verification. Intrinsic methods operate without external knowledge, leveraging signals like token probabilities for uncertainty estimation [20,28,34], self-consistency [22], or internal activation patterns [1,4,9]. Although insightful, these methods fail to spot incorrect claims made with high confidence. Tool-augmented methods address this by retrieving external evidence [7,8,23,31,32]. However, these methods all adhere to a fixed verification strategy. They execute a uniform workflow regardless of claim complexity, making them brittle against diverse hallucination patterns. Thus, the core challenge of learning adaptive strategies remains unsolved.

2.2 Agent Tuning

Agent tuning has emerged as a powerful paradigm for allowing smaller models to learn sophisticated behaviors by finetuning them on high quality decision trajectories [5,16,37]. Although prior work has successfully distilled strategies for general purpose reasoning and planning [3,25,26,30,35], its application to hallucination detection [6] reveals a critical limitation. Current approaches primarily focus on mimicking static verification routines. As a result, the student

model learns to follow a fixed trajectory but lacks the capability to adjust when the strategy itself is flawed. This highlights the need to distill a strategy that is inherently dynamic and adaptive.

3 Method

3.1 Problem Formulation

Given a claim consisting of a user query Q and a response R generated by the model, our ultimate goal is to predict a binary label $Y \in \{\text{Hallucination}, \text{Not Hallucination}\}$. Rather than directly mapping the claim (Q, R) to Y, our approach focuses on optimizing the evidence gathering process that informs the final decision. A high quality process is guided by an appropriate strategy for selecting and using information gathering tools. We propose that the final verdict is determined by the quality of this evidence gathering process.

Given our premise that strategy is critical for effective information gathering, we reframe hallucination detection as a learning problem of dynamic strategies and formalize this process through a verification strategy π_{strat}, which orchestrates the verification process. Our goal is to learn an optimal strategy π^*_{strat} that is both effective and dynamically adaptable to the specific claim.

Strategy execution relies on the collaboration of multiple specialized agents, as hallucinations are complex and diverse, requiring them to jointly operate the detection framework. Their interactions are captured in a verification trajectory τ similar to ReAct [36]. A trajectory is a sequence of states and the transitions between them, composed of interleaved thoughts, actions, and observations:

$$\tau = (s_0, t_1, a_1, o_1, s_1, t_2, a_2, o_2, s_2, \ldots, s_N) \tag{1}$$

where s_n represents the state at step n, which includes the initial claim and the history of all prior steps $(t_i, a_i, o_i)_{i=1}^{n-1}$. Specifically, the trajectory components are: Thought (t_i), the explicit reasoning guided by the overarching strategy π_{strat} analyzing the current state to decide the next action; Action (a_i), a concrete operation typically involving an external tool call; Observation (o_i), the output returned from the tool after executing action a_i. The final verdict Y is determined by the terminal state s_N. Since the initial strategy π_{strat} orchestrates the interactions generating this trajectory, the final verdict is a direct result of the initial plan.

Therefore, the core challenge shifts from learning a simple classifier to discovering an optimal verification strategy π^*_{strat}, that consistently guides the agent interaction to a correct and robust verdict.

3.2 Overview

To overcome the insufficient adaptability of the fixed verification strategy, we propose LEAP, a framework that shifts the paradigm from fixed execution to dynamic strategy learning. As illustrated in Fig. 2, LEAP consists of three stages.

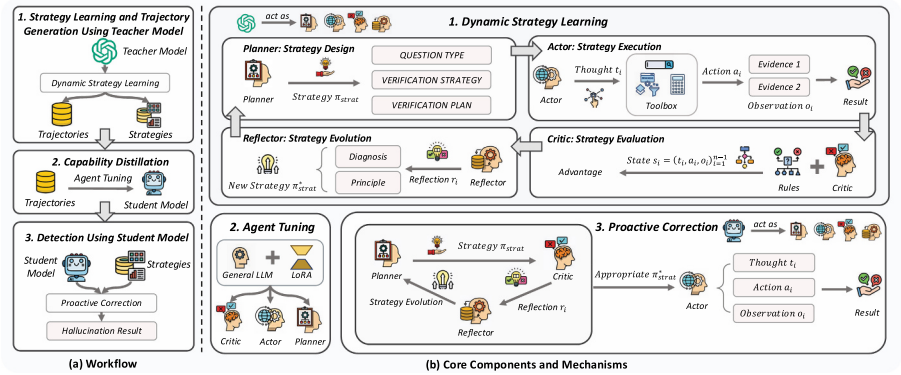

Fig. 2. The LEAP framework with its (a) workflow and (b) core components, including three main steps: 1. Dynamic Strategy Learning and Trajectory Generation Using Teacher Model: A teacher model uses the dynamic learning loop to learn from failure and generate trajectories. 2. Capability Distillation via Agent Tuning: The trajectories are distilled into an efficient student model. 3. Detection with Proactive Correction Using Student Model: The student model adaptively refines its plan before execution to ensure appropriate strategies.

First, to gather diverse strategies, we employ dynamic strategy learning, using *a teacher model* within a dynamic learning loop that systematically learns from execution failures to continually generate superior strategies. **Second**, to transfer dynamic learning capabilities into *an efficient small student model*, we utilize agent tuning by training on specific expert trajectories. **Finally**, to ensure that the strategy is adaptive in dynamical execution environments, we introduce a proactive correction mechanism, which can preemptively evaluate and optimize its own verification strategies for each specific claim before execution, thus ensuring robust performance.

3.3 Dynamic Strategy Learning

The first phase is dynamic strategy learning designed to generate the diverse verification strategies. We achieve this through a closed loop where four agents collaborate to systematically learn from execution failures: the planner designs a strategy, the actor executes it to produce a trajectory, the critic evaluates the outcome, and for any failures, the reflector generates corrective feedback that is fed back to the planner. This iterative process not only drives the continuous evolution of strategies, but also yields the high quality trajectories required for the subsequent agent tuning phase.

Planner: Strategy Design. The planner is an agent responsible for creating a high-level customized verification strategy π_{strat} for an input claim, using past

experiences to go beyond fixed strategies. It generates the strategy as follows:

$$\pi_{strat} = \text{Planner}(s_0, \mathcal{P}_p, \mathcal{R}_{\text{retrieved}}), \tag{2}$$

where s_0 is the initial state containing the claim, $\mathcal{P}_p$ is the prompt with task instructions, and $\mathcal{R}_{\text{retrieved}}$ is a set of relevant reflections retrieved from memory. A complete strategy π_{strat} usually specifies the type of problem, a high-level verification strategy, and a concrete verification plan.

To inform its planning, the planner performs reflection retrieval, querying a memory $\mathcal{M}_P$ to find the K most relevant past reflections:

$$\mathcal{R}_{retrieved} = \{r \mid \text{rank}(\text{sim}(\mathbf{e}_r, \mathbf{e}_{s_0})) < K, r \in \mathcal{M}_P\} \tag{3}$$

where $\mathbf{e}$ denotes the text embedding and $\text{sim}(\cdot, \cdot)$ represents cosine similarity, implemented with FAISS [14] for efficient retrieval. The retrieved reflections provide relevant experiences for the planner to synthesize a new strategy. When the reflector generates a new reflection r_{new} from a failure, it is integrated into the memory:

$$\mathcal{M}_P \leftarrow \mathcal{M}_P \cup \{r_{\text{new}}\}. \tag{4}$$

Actor: Strategy Execution. The actor executes the verification strategy π_{strat} by generating the verification trajectory τ. To do so, it performs a series of actions, primarily by invoking external tools. We equip the actor with a versatile toolbox inspired by previous work [6], including search engine, calculator, code interpreter, etc. Details are available in Appendix A.3.

At each step n, the action a_n is determined based on the strategy π_{strat}, the current state s_n, and a dynamically retrieved set of relevant examples Ψ^n:

$$a_n = \text{Actor}(s_n, \pi_{strat}, \mathcal{P}_A, \Psi^n), \tag{5}$$

where Ψ^n is retrieved based on similarity to the current claim from the actor's memory $\mathcal{M}_A$, which stores tuples of (claim, strategy, advantage value). Based on the stored advantage value A, it's composed of samples from both successful precedents Ψ^n_{pos} for $A > 0$ and cautionary tales Ψ^n_{neg} for $A \leq 0$. Upon completing the trajectory τ, the actor passes it to the critic for evaluation. The actor's memory is updated with the strategy and its evaluated advantage:

$$\mathcal{M}_A \leftarrow \mathcal{M}_A \cup \{(s_0, \pi_{strat}, A(\pi_{strat}, \tau))\}, \tag{6}$$

where $A(\pi_{strat}, \tau)$ is the advantage value calculated by the critic.

Critic: Strategy Evaluation. The critic is an evaluator agent that provides a quantitative feedback signal by calculating the advantage value $A(\pi_{strat}, \tau)$ for a completed trajectory τ. Once the trajectory is finalized, the critic assigns a comprehensive strategic score $V(s_0)$ based on the predefined scoring principles illustrated in Appendix I.

To formalize this process, the critic estimates a state-value function $V(s_n)$, representing the measured utility of the trajectory from state s_n. The critic models this function by using its memory $\mathcal{M}_C$ to store past (state, value) pairs, allowing it to fit the value function via in-context learning:

$$V(s_n) = \text{Critic}(s_n, \mathcal{P}_C, \mathcal{M}_C). \tag{7}$$

After each trajectory, the newly computed state values are used to update the memory:

$$\mathcal{M}_C \leftarrow \mathcal{M}_C \cup \{(s_n, V(s_n)) \mid s_n \in \tau\}. \tag{8}$$

With the learned value function, we compute the advantage by adapting the commonly used advantage function [27] to our task:

$$A(\pi_{strat}, \tau) = R_T - V(s_0) - \lambda \cdot N_{\text{tools}}, \tag{9}$$

where R_T denotes detection success, $V(s_0)$ denotes the strategic quality score and $\lambda \cdot N_{\text{tools}}$ penalizes redundancy to ensure a parsimonious yet accurate verification path. This advantage A serves as the core signal for strategy optimization and subsequent agent distillation.

Reflector: Strategy Evolution. The reflector is the engine agent for strategy evolution, operating specifically on failures. When a trajectory is assigned a negative value by the critic, the reflector generates corrective feedback:

$$r_{\text{new}} = \text{Reflector}(\tau_{\text{fail}}, \mathcal{P}_R), \tag{10}$$

using a specialized prompt $\mathcal{P}_R$, the reflector analyzes the failed trajectory τ_{fail} and generates a structured reflection r_{new}, which contains diagnosis of failures, generalizable high-level principles to prevent similar errors, and a revised verification strategy. The new reflection r_{new} is then added to the planner's memory $\mathcal{M}_P$, directly closing the learning loop and systematically converting failures into improved future strategies.

Through this loop, we curate a pool of 1,889 distinct strategies that cover a range of patterns, including entity validation and complex contextual synthesis. Details are in Appendix C.

3.4 Agent Tuning

The second phase is agent tuning, where we finetune efficient small models using the trajectories collected in the first phase. The trajectories provide the entire reasoning process, allowing the student model to learn how to plan, not just what the result is. This process distills dynamic learning capabilities, resulting in a small but powerful detector capable of adaptive learning.

Trajectory Collection. First, we employ the LEAP framework that uses a powerful model to process verification tasks from commonly used benchmarks such as HaluEval [17] and MMLU-Pro [29]. To ensure the quality of training data, we perform a strict curation process. We retain only the trajectories, those that not only reach the correct final verdict but also exhibit high efficiency. This quality is quantified by the advantage value computed by the critic; we filter for trajectories where $A(\pi_{strat}, \tau)$ is positive. This process yields high quality trajectories $\mathcal{D}_{\text{expert}}$.

Trajectory Finetuning. To mitigate the planning instability in small models, we implement functional specialization by training distinct LoRA adapters [11] for the planner, actor, and critic. This decoupling prevents interference between capabilities, allowing the system to dynamically orchestrate these specialized models. For each trajectory $\tau \in \mathcal{D}_{expert}$, we format it as a multi-turn conversation sequence consisting of the initial state s_0 and the full history of interleaved thoughts, actions, and observations $(t_1, a_1, o_1, \ldots, t_T, a_T, o_T)$.

To finetune the planner and actor, we employ a standard instruction-tuning objective where the model takes the full history as context but computes the loss only on the thought and action tokens generated by the agent, masking the observation tokens provided by tools. The objective is defined as:

$$\mathcal{L}_{SFT}(\theta) = -\sum_{(s_0,\tau)\in\mathcal{D}} \sum_{j\in\mathcal{I}_{agent}} \log P_\theta(y_j | s_0, y_{<j}), \tag{11}$$

where y represents the linearized sequence of all tokens in the trajectory, and $\mathcal{I}_{agent}$ denotes the indices of tokens belonging to thoughts and actions.

For the critic, we construct a specialized dataset $\mathcal{D}_{crit} = \{(s_0, \pi_{strat}, A)_i\}$, where the labels A are advantage values collected from the teacher's actual executions. By predicting these values from the strategy alone, the critic internalizes the mapping between strategic logic and eventual success. This capability provides the predictive foundation for the proactive correction mechanism to stabilize planning during inference.

3.5 Proactive Correction

The final phase is the proactive correction mechanism designed to ensure that the tuned small model adaptively optimizes its strategy for each specific claim, thereby mitigating the planning instability.

Given a claim s_0, the finetuned planner first generates an initial strategy π_{strat}. Critically, rather than proceeding to immediate execution, LEAP intercepts this plan for a preemptive evaluation. The finetuned critic assesses the strategy's quality by predicting an estimated advantage score:

$$\hat{A}(\pi_{strat}) = \text{Critic}_{\text{tuned}}(s_0, \pi_{strat}, \mathcal{P}_C, \mathcal{M}_C), \tag{12}$$

where $\hat{A}(\pi_{strat})$ predicts the likely success and efficiency of the proposed strategy. This is different from conventional post-hoc reflection by identifying potential reasoning flaws before tool calls.

The predicted advantage is then compared to a confidence threshold θ_{corr}. If the strategy is deemed sufficiently robust (i.e. $\hat{A}(\pi_{strat}) \geq \theta_{\text{corr}}$), it is approved for the actor. Otherwise, the proactive correction loop is triggered. The reflector diagnoses the strategy's weaknesses and generates corrective feedback r_{corr}, which guides the planner to synthesize a revised and superior strategy π'_{strat}. This iterative refinement ensures that the finetuned actor executes only validated and precise strategies. Once a strategy is approved, the actor takes over. It executes the strategy, generating the thought-action trace τ' by interacting with the necessary tools to reach the final detection verdict.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. To comprehensively evaluate LEAP, we use three challenging benchmarks with strictly non-overlapping splits: HaluEval [17] and MMLU-Pro [29] as in-domain datasets and XTRUST [18] as out-of-domain datasets to evaluate robustness. Following prior work [6,32], we strictly curate the test sets with hallucination ratios ranging from 44% to 55% to prevent majority-class bias, using accuracy and F1 score as evaluation metrics. Details in Appendix A.

Baselines. We compare against state-of-the-art baselines, including (1) intrinsic methods (Perplexity [24], LN-entropy [21], Semantic Entropy [15], SelfCheckGPT [22]); (2) tool-augmented prompt-based methods (Factool [7], SAFE [31], FIRE [32]) and (3) finetuned methods (HaluAgent [6]). Details in Appendix A.2.

Implementation Details. We employ GPT-4o mini as the teacher model to generate diverse trajectories with decoding temperature 1.0 and top-p 1.0, selected for its proven high capability in fact checking [32]. We distill three open source models: Qwen2.5-7B-Instruct [33], Llama3.1-8B-Instruct [10], and Mistral-8B-Instruct [13]. The student models are finetuned on the trajectories using LoRA, with a rank of 8, α of 32, a learning rate of 1e-4 and hyperparameters $\lambda = 0.1$, $K = 2$, $\theta_{\text{corr}} = 0$. For fair comparison, the temperature for all models is set to 0.0 during evaluation to ensure deterministic outputs.

4.2 Main Results and Analysis

Table 1 demonstrates the consistent superiority of LEAP across all models and datasets. On Qwen2.5-7B, our method achieves an accuracy of 69.89%, surpassing the best baseline by a margin of 7.31%. This performance highlights three critical insights into effective hallucination detection.

Table 1. Main results on three hallucination detection datasets. ID and OOD denote In-Domain and Out-of-Domain, respectively. **Bold** denotes the best performance, and underline indicates the second-best.

Models	Methods	ID				OOD		Average	
		HaluEval		MMLU-Pro		XTRUST			
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
Qwen2.5-7B	Perplexity	56.33	57.05	57.33	68.00	47.50	62.63	54.50	62.55
	LN-entropy	54.67	66.00	56.67	67.50	48.00	62.32	53.75	65.64
	Semantic Entropy	50.67	66.67	55.67	66.67	47.00	61.31	51.63	65.33
	Self-Check(0)	51.20	63.40	56.66	69.54	46.07	59.92	52.00	64.97
	Self-Check(3)	55.94	62.30	59.56	64.52	52.52	60.24	56.70	62.74
	FACTOOL	59.00	67.20	59.60	69.23	47.42	56.78	56.38	65.53
	SAFE	57.60	66.10	55.81	68.28	44.25	54.03	53.73	64.25
	FIRE	65.67	68.31	61.05	69.23	53.61	57.94	60.97	66.13
	HaluAgent	70.55	71.33	54.68	55.00	61.93	64.11	62.58	63.62
	LEAP	**74.19**	**75.00**	**69.81**	**75.31**	**64.00**	**66.36**	**69.89**	**72.88**
Llama3.1-8B	Perplexity	50.00	66.67	55.00	70.97	44.00	61.11	50.38	66.89
	LN-entropy	57.34	66.14	58.34	68.51	46.00	61.70	54.88	65.92
	Semantic Entropy	50.67	66.96	55.33	65.59	43.00	58.99	50.50	64.45
	Self-Check(0)	51.37	67.43	54.88	70.74	44.16	61.27	50.89	67.23
	Self-Check(3)	49.82	64.47	55.25	70.67	46.24	61.83	51.05	66.37
	FACTOOL	50.71	66.83	55.33	68.84	48.18	64.68	52.16	67.24
	SAFE	60.64	70.56	57.96	70.49	60.58	**66.25**	59.64	**69.75**
	FIRE	65.96	70.73	58.85	69.51	58.09	65.87	61.72	69.26
	HaluAgent	69.36	**71.92**	54.41	54.05	63.30	59.65	63.36	62.92
	LEAP	**70.00**	71.88	**64.23**	**71.18**	**64.50**	63.59	**66.54**	69.71
Mistral-8B	Perplexity	50.00	66.67	56.67	67.17	44.00	61.11	51.00	65.47
	LN-entropy	56.67	69.48	58.67	68.04	44.50	61.32	54.38	66.90
	Semantic Entropy	51.33	66.67	54.67	69.64	41.00	57.55	50.00	65.50
	Self-Check(0)	51.67	61.74	60.14	70.37	44.95	54.77	53.02	63.33
	Self-Check(3)	55.70	65.63	59.12	70.42	47.74	61.19	54.98	66.35
	FACTOOL	62.02	68.95	62.26	71.43	48.37	55.87	59.15	67.27
	SAFE	55.05	67.83	60.61	**72.19**	45.16	59.72	54.96	67.75
	FIRE	53.00	66.33	58.23	71.11	48.37	60.70	53.87	66.95
	HaluAgent	73.49	72.85	54.29	70.37	46.43	59.46	62.75	66.55
	LEAP	**74.00**	**74.17**	**63.21**	71.15	**57.50**	**61.54**	**65.90**	**69.77**

The Necessity of External Tools. The substantial performance gap between LEAP and intrinsic methods confirms that relying solely on internal signals is insufficient. Intrinsic approaches are bounded by the model's parametric knowledge and fail when the model generates incorrect information with high confidence, necessitating external evidence for reliable verification.

Advantage of Dynamic Planning Over Fixed Strategies. Comparisons with tool-augmented prompt-based baselines reveal that tool access alone is inadequate without adaptive strategies. Fixed pipelines like Factool and SAFE mechanically execute predefined workflows, which prove brittle against relational hallucinations where errors stem from flawed logical connections rather than simple factual inaccuracies. LEAP overcomes this fixity by employing dynamic strategy learning, enabling the model to tailor its verification plan to the specific logical structure of each claim.

Distilling Planning Capabilities versus Mimicking Execution. Crucially, LEAP significantly outperforms HaluAgent, the strongest finetuned baseline. While HaluAgent learns to mimic a fixed execution procedure, it inherits the limitations of the fixed pipeline. In contrast, LEAP distills dynamic planning and proactive correction capabilities from the teacher model. By training on trajectories that involve strategy evaluation and refinement, our student model learns to assess and optimize its own verification logic before execution, rather than merely following a fixed script.

Table 2. The ablation results on Qwen2.5-7B. The best results are in **bold**.

Method	HaluEval		MMLU-Pro		XTRUST	
	Acc	F1	Acc	F1	Acc	F1
LEAP	**74.19**	**75.00**	**69.81**	**75.31**	**64.00**	**66.36**
w/o Correction	71.33	73.46	66.20	71.55	63.50	65.07
w/o Dynamic Strategy	70.55	71.33	54.68	55.00	61.93	64.11
w/o Reflection Retrieval	70.00	71.52	55.59	58.36	61.00	65.18
w/o Memory Retrieval	65.33	67.09	58.19	61.04	61.81	65.45
w/o Tools	59.00	49.80	54.33	45.42	53.00	50.53

4.3 Ablation Study

To dissect component contributions, we conduct an ablation study using Qwen2.5-7B. As detailed in Table 2, we examine five variants: *w/o Correction*, which disables inference-time refinement; *w/o Dynamic Strategy*, which replaces the adaptive planner with a fixed pipeline; *w/o Reflection Retrieval* and *w/o*

Memory Retrieval, which exclude past insights and execution trajectories, respectively; and *w/o Tools*, removing all external tools.

The results confirm the integral role of each component. Specifically, removing memory retrieval leads to a marked decline on HaluEval, validating that accessing concrete execution examples is critical for precise tool usage in fact-centric tasks. In contrast, the exclusion of reflection retrieval hampers performance on the reasoning-heavy MMLU-Pro. This performance gap highlights that abstract insights are pivotal for guiding the planner through complex adaptation, effectively bridging the divide between fixed execution and adaptive planning. Furthermore, disabling proactive correction induces consistent degradation, verifying the value of pre-execution refinement in mitigating potential errors. Finally, substituting the dynamic strategy with a fixed pipeline results in the most severe drop of over 20% F1 on MMLU-Pro, which demonstrates that an adaptive strategy is fundamental to overcoming the limitations of fixed paradigms.

4.4 Analysis on Trajectory Distillation

To assess the efficacy of trajectory distillation, we compare the performance of the Qwen2.5-7B student directly with its GPT-4o mini teacher. As shown in Table 3, the student not only achieves parity but surpasses the teacher, attaining 74.19% accuracy on HaluEval and 69.81% on MMLU-Pro. This performance indicates that the distillation process effectively transfers the decision-making logic and dynamic planning strategies, rather than merely mimicking outputs. Consequently, LEAP proves capable of compressing complex reasoning capabilities into a small model, enabling efficient deployment without compromising reasoning depth.

Table 3. Comparison between Qwen2.5-7B student model and GPT-4o mini teacher.

Models	HaluEval		MMLU-Pro		XTRUST	
	Acc	F1	Acc	F1	Acc	F1
LEAP	74.19	75.00	69.81	75.31	64.00	66.36
GPT 4o mini	73.67	75.69	69.31	76.32	64.50	68.16

4.5 Analysis on Cross-Model Generalization

To assess architectural robustness, we use heterogeneous teacher-student pairs: Qwen2.5-72B as teacher and Llama3.1-8B as student. Figure 3 shows the student model achieves substantial gains over the vanilla baseline, with accuracy

improvements of 20.1% on HaluEval and 15.9% on MMLU-Pro. Remarkably, as indicated by the teacher upper bound, the student approaches the performance of teacher model. These results validate LEAP's cross-model transferability, confirming that smaller models can effectively inherit complex dynamic planning capabilities from stronger teachers despite architectural discrepancies.

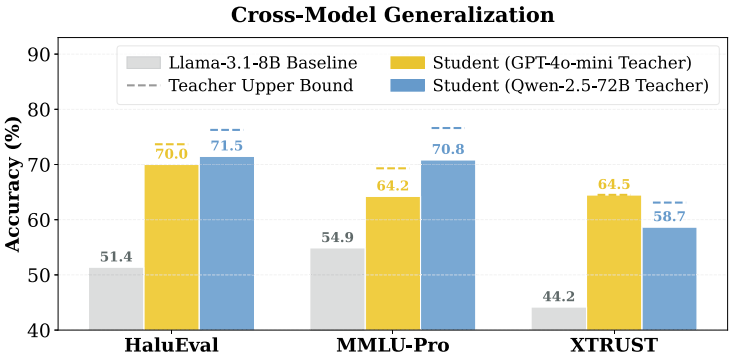

Fig. 3. Cross-model generalization performance of the Llama3.1-8B student model.

Table 4. Class-wise performance on Qwen2.5-7B.

Dataset	Method	Hallucinated Acc.	Faithful Acc.
HaluEval	HaluAgent	73.29%	67.81%
	LEAP	78.83%	69.72%
MMLU-Pro	HaluAgent	50.99%	59.06%
	LEAP	85.92%	51.22%
XTRUST	HaluAgent	76.14%	50.46%
	LEAP	80.68%	50.89%

4.6 Class-Wise Performance Analysis

To ensure LEAP's gains stem from reasoning capabilities rather than a bias toward hallucination, we analyze accuracy on hallucinated versus faithful samples as in Table 4. Results show LEAP excels in detecting both hallucinated and faithful content, confirming its gains are not driven by class-specific bias.

On MMLU-Pro, although LEAP's faithful accuracy declines by 7.84% relative to HaluAgent, its hallucination detection surges by 34.93%. This substantial margin demonstrates proactive correction mechanism effectively identifies flaws that baselines default to as faithful, thus ensuring a robust defense against subtle errors. LEAP achieves a superior balance between hallucination and faithfulness, which significantly reduces the risk of accepting false information in high-stakes scenarios.

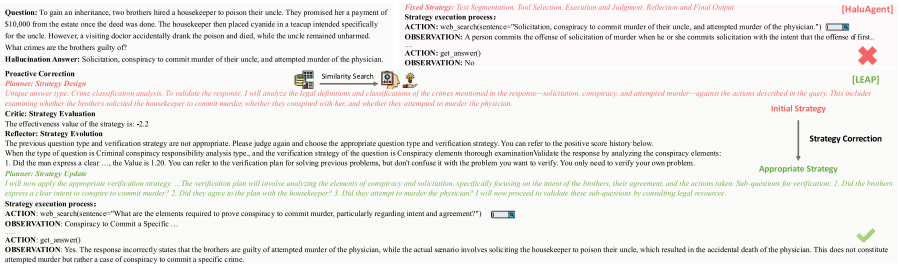

Fig. 4. A case study on a complex reasoning task. HaluAgent uses its fixed strategy and LEAP uses its dynamic planning, including strategy correction and precise execution.

4.7 Case Study

Figure 4 presents a complex legal case where a claim contains multiple nuanced legal concepts: solicitation, conspiracy, and attempted murder. The claim subtly misapplies the concept of "attempted murder" to accidental death, creating a challenging hallucination. We compare the fixed verification process of HaluAgent with the adaptive approach of LEAP. HaluAgent hindered by its fixed strategy adopts a naive verification plan. It attempts to validate the entire claim at once, without scrutinizing its individual components. As a result, it retrieves only general information and misses the subtle factual error regarding "attempted murder" thus incorrectly labeling the statement as non-hallucination.

In contrast, LEAP demonstrates a multistage adaptive process. The planner first designs an initial strategy to analyze the legal definitions of the three crimes mentioned. Critically, before execution, the proactive correction mechanism assesses this initial strategy. It identifies the strategy as suboptimal and leverages its memory from past experiences to refine it into a more precise strategy focused on verifying the core legal elements of each crime. This optimized strategy enables a focused execution—verifying "attempted murder" as a distinct sub question—that successfully isolates and identifies the hallucination.

This case study highlights how LEAP's ability to plan, critique, and revise its own strategy leads to a more robust detection process.

5 Conclusion

In this work, we propose LEAP, a framework that shifts tool-augmented hallucination detection from fixed execution to dynamic strategy learning. By integrating a dynamic strategy learning loop with a proactive correction mechanism, LEAP enables efficient small models to overcome planning instability and adaptively optimize verification strategies before execution. Experiments on three datasets validate the superiority of LEAP, offering a scalable and reliable solution for robust hallucination detection in practical scenarios.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (NSFC) (Grant No. 62576256) and Zhongguancun Academy (Grant No. 02012402).

References

1. Bazarova, A., Yugay, A., Shulga, A., et al.: Hallucination detection in llms via topological divergence on attention graphs. CoRR **abs/2504.10063** (2025). https://doi.org/10.48550/ARXIV.2504.10063
2. Belyi, M., Friel, R., Shao, S.: Luna: a lightweight evaluation model to catch language model hallucinations with high accuracy and low cost. In: Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pp. 398–409. Association for Computational Linguistics (2025)
3. Bo, X., Zhang, Z., Dai, Q., et al.: Reflective multi-agent collaboration based on large language models. Adv. Neural. Inf. Process. Syst. **37**, 138595–138631 (2024)
4. Chen, C., Liu, K., Chen, Z., et al.: INSIDE: llms' internal states retain the power of hallucination detection. In: The Twelfth International Conference on Learning Representations, ICLR 2024. OpenReview.net (2024)
5. Chen, Z., Liu, K., Wang, Q.: Agent-flan: designing data and methods of effective agent tuning for large language models. In: Findings of the Association for Computational Linguistics, ACL 2024, pp. 9354–9366. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.FINDINGS-ACL.557
6. Cheng, X., Li, J., Zhao, X.: Small agent can also rock! empowering small language models as hallucination detector. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, pp. 14600–14615. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.EMNLP-MAIN.809
7. Chern, I., Chern, S., Chen, S., et al.: Factool: factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. arXiv preprint arXiv:2307.13528 (2023)

8. Dhuliawala, S., Komeili, M., Xu, J., et al.: Chain-of-verification reduces hallucination in large language models. In: Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024, pp. 3563–3578. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.FINDINGS-ACL.212
9. Du, X., Xiao, C., Li, S.: Haloscope: harnessing unlabeled LLM generations for hallucination detection. In: Advances in Neural Information Processing Systems, vol. 38: NeurIPS 2024 (2024)
10. Grattafiori, A., Dubey, A., Jauhri, A.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
11. Hu, E.J., Shen, Y., Wallis, P.: Lora: low-rank adaptation of large language models. In: ICLR (2022)
12. Huang, L., Yu, W., Ma, W., et al.: A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. ACM Trans. Inf. Syst. **43**(2), 1–55 (2025)
13. Jiang, A.Q., et al.: Mixtral of experts. arXiv preprint arXiv:2401.04088 (2024)
14. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Trans. Big Data **7**(3), 535–547 (2019)
15. Kuhn, L., Gal, Y., Farquhar, S.: Semantic uncertainty: linguistic invariances for uncertainty estimation in natural language generation. In: The Eleventh International Conference on Learning Representations, ICLR 2023. OpenReview.net (2023)
16. Lai, S., Xu, Z., Zhang, W., et al.: Llmlight: large language models as traffic signal control agents. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, pp. 2335–2346. ACM (2025). https://doi.org/10.1145/3690624.3709379
17. Li, J., Cheng, X., Zhao, X., et al.: Halueval: a large-scale hallucination evaluation benchmark for large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pp. 6449–6464. Association for Computational Linguistics (2023). https://doi.org/10.18653/V1/2023.EMNLP-MAIN.397
18. Li, Y., Wang, Y., Chang, Y., Wu, Y.: Xtrust: on the multilingual trustworthiness of large language models. arXiv preprint arXiv:2409.15762 (2024)
19. Luo, J., Li, T., Wu, D., Jenkin, M., Liu, S., Dudek, G.: Hallucination detection and hallucination mitigation: an investigation (2024)
20. Luo, J., Xiao, C., Ma, F.: Zero-resource hallucination prevention for large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024, pp. 3586–3602. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.FINDINGS-EMNLP.204
21. Malinin, A., Gales, M.J.F.: Uncertainty estimation in autoregressive structured prediction. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021). https://openreview.net/forum?id=jN5y-zb5Q7m
22. Manakul, P., Liusie, A., Gales, M.J.F.: Selfcheckgpt: zero-resource black-box hallucination detection for generative large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, pp. 9004–9017. Association for Computational Linguistics (2023). https://doi.org/10.18653/V1/2023.EMNLP-MAIN.557

23. Min, S., Krishna, K., Lyu, X., et al.: Factscore: fine-grained atomic evaluation of factual precision in long form text generation. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023, pp. 12076–12100. Association for Computational Linguistics (2023). https://doi.org/10.18653/V1/2023.EMNLP-MAIN.741
24. Ren, J., Luo, J., Zhao, Y.: Out-of-distribution detection and selective generation for conditional language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023. OpenReview.net, (2023)
25. Shi, W., He, X., Zhang, Y.: Large language models are learnable planners for long-term recommendation. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1893–1903 (2024)
26. Shinn, N., Cassano, F., Gopinath, A., et al.: Reflexion: language agents with verbal reinforcement learning. In: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023 (2023)
27. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, vol. 12 (1999)
28. Varshney, N., Yao, W., Zhang, H., et al.: A stitch in time saves nine: detecting and mitigating hallucinations of LLMs by validating low-confidence generation. CoRR **abs/2307.03987** (2023). https://doi.org/10.48550/ARXIV.2307.03987
29. Wang, Y., Ma, X., Zhang, G., et al.: Mmlu-pro: a more robust and challenging multi-task language understanding benchmark. Adv. Neural. Inf. Process. Syst. **37**, 95266–95290 (2025)
30. Wei, J., Wang, X., Schuurmans, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)
31. Wei, J., Yang, C., Song, X., et al.: Long-form factuality in large language models. In: Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024 (2024)
32. Xie, Z., Xing, R., Wang, Y.: FIRE: fact-checking with iterative retrieval and verification. In: Findings of the Association for Computational Linguistics: NAACL 2025, pp. 2901–2914. Association for Computational Linguistics (2025). https://doi.org/10.18653/V1/2025.FINDINGS-NAACL.158
33. Yang, A., Yang, B., Zhang, B., et al.: Qwen2.5 technical report (2025)
34. Yao, J.Y., Ning, K.P., Liu, Z.H., et al.: LLM lies: Hallucinations are not bugs, but features as adversarial examples. CoRR **abs/2310.01469** (2023). https://doi.org/10.48550/ARXIV.2310.01469
35. Yao, S., Yu, D., Zhao, J., et al.: Tree of thoughts: deliberate problem solving with large language models. 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)
36. Yao, S., Zhao, J., Yu, D., et al.: React: synergizing reasoning and acting in language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023)

37. Zeng, A., Liu, M., Lu, R., et al.: Agenttuning: enabling generalized agent abilities for llms. In: Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024, pp. 3053–3077. Association for Computational Linguistics (2024). https://doi.org/10.18653/V1/2024.FINDINGS-ACL.181
38. Zhang, Y., Li, Y., Cui, L.: Siren's song in the ai ocean: a survey on hallucination in large language models. arXiv preprint arXiv:2309.01219 (2023)