Zero-knowledge LLM hallucination detection and mitigation through fine-grained cross-model consistency

Aman Goel*, **Daniel Schwartz***, **Yanjun Qi**
Amazon Web Services, USA
{goelaman, dansw, yanjunqi}@amazon.com

Abstract

Large language models (LLMs) have demonstrated impressive capabilities across diverse tasks, but they remain susceptible to hallucinations—generating content that appears plausible but contains factual inaccuracies. We present FINCH-ZK, a black-box framework that leverages **FIN**e-grained **C**ross-model consistency to detect and mitigate **H**allucinations in LLM outputs without requiring external knowledge sources. FINCH-ZK introduces two key innovations: 1) a cross-model consistency checking strategy that reveals fine-grained inaccuracies by comparing responses generated by diverse models from semantically-equivalent prompts, and 2) a targeted mitigation technique that applies precise corrections to problematic segments while preserving accurate content. Experiments on the FELM dataset show FINCH-ZK improves hallucination detection F1 scores by 6-39% compared to existing approaches. For mitigation, FINCH-ZK achieves up to 9 absolute percentage points improvement in answer accuracy on the GPQA-diamond dataset when applied to state-of-the-art models like Llama 4 Maverick and Claude 4 Sonnet. Extensive evaluation on multiple datasets demonstrates that FINCH-ZK provides a practical, deployment-ready safeguard for enhancing factual reliability in production LLM systems.

1 Introduction

With the rapid deployment of large language models (LLMs) across diverse applications, ensuring factual accuracy and reliability has become increasingly critical for enterprise systems. LLMs frequently generate plausible-sounding but factually incorrect information—a phenomenon known as hallucination—posing significant risks in high-stakes domains.

Existing black-box hallucination management techniques typically address either detection or mitigation, but seldom integrate both. Black-box detection systems in the absence of external knowledge struggle with single-LLM biases, and coarse outputs lacking interpretability, while mitigation approaches similarly over-reformulate, reuse biased models, lack integrated detection-correction pipelines, and offer little transparency (detailed review in §2.1).

Our objective is to develop a practical LLM hallucination management system that integrates detection and targeted mitigation without external knowledge requirements. In this paper, we introduce FINCH-ZK, which integrates techniques like consistency checking (Wang et al., 2023; Manakul et al., 2023) with a novel multi-stage mitigation approach that precisely corrects only problematic segments while preserving accurate content and embodying diverse reasoning patterns across model families. Our key contributions include:

- We introduce FINCH-ZK, an integrated black-box framework that combines existing detection techniques with a novel multi-stage mitigation process for targeted hallucination correction, addressing a critical gap between detection and mitigation in existing LLM safeguards.
- We demonstrate how leveraging prompting variations (adding dynamic semantic-preserving alterations to the input prompt) and cross-model consistency checking (comparing outputs across different model architectures) provide more robust detection than single-model approaches, improving detection F1 scores by 6-39% on the FELM dataset (Zhao et al., 2023) compared to state-of-the-art methods.
- We present an interpretable multi-stage mitigation pipeline that applies targeted corrections only to problematic segments identified through fine-grained analysis while maintaining coherence and completeness through cross-model reasoning feedback, achieving up to 9 absolute

*Equal contributions

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1982–1999
November 4-9, 2025

percentage points improvement in answer accuracy on the GPQA-diamond dataset (Rein et al., 2024).

- We provide comprehensive empirical evidence showing that the integration of diverse sampling strategies with targeted correction significantly outperforms existing approaches in the zero-knowledge setting,[1] with quantitative ablation studies identifying the relative contribution of each system component.

The framework is designed for practical deployment in production environments, with efficient multi-threaded processing, comprehensive logging support, modular architecture supporting various LLMs, and rich user feedback.

2 Methodology

2.1 Background and Related Works

Following the taxonomy of (Huang et al., 2025), we categorize hallucinations into: 1) knowledge errors–factually incorrect information, 2) reasoning errors–flawed logical inference, 3) irrelevant content–off-topic responses, and 4) instruction-following failures. FINCH-ZK primarily targets types 1-2, which pose the highest risk in enterprise applications.

Existing black-box hallucination management approaches fall into two categories. For detection, techniques include: a) external knowledge-based approaches like RAG (Lewis et al., 2020) that rely on data sources for fact-checking, and b) internal consistency methods like SelfCheckGPT (Manakul et al., 2023) that analyze variations across model outputs. For mitigation, common techniques include self-correction through iterative refinement (Wang et al., 2023), chain-of-thought reasoning (Wei et al., 2023), and majority voting (Lightman et al., 2023).

Detection systems face three primary limitations: 1) RAG-based methods require comprehensive knowledge bases often unavailable for specialized domains or inaccessible due to privacy concerns; 2) zero-knowledge consistency-based approaches typically rely on a single LLM architecture, making them prone to high-certainty hallucinations due to missing diverse reasoning patterns; and 3) most systems operate at coarse granularity, lacking fine-grained analysis and interpretable explanations for flagged content.

[1]Zero-knowledge refers to requiring no external knowledge sources (e.g., databases, search APIs), not cryptographic zero-knowledge proofs.

Mitigation approaches suffer from complementary shortcomings: 1) most systems attempt wholesale reformulation rather than targeted correction, often modifying accurate content while fixing errors; 2) they frequently rely on the same model that produced the hallucination to correct it, perpetuating biases and reasoning patterns; 3) many approaches lack integration between detection and correction mechanisms, resulting in inefficient pipelines; and 4) few systems provide transparency into why content was flagged and how corrections were determined.

2.2 Proposed: FINCH-ZK

To address the above limitations and provide an integrated workflow for hallucination management, we propose FINCH-ZK, a framework for **FIN**e-grained **C**ross-model consistency for **H**allucination detect and mitigate with **Z**ero **K**nowledge. FINCH-ZK addresses key limitations in existing approaches through two primary innovations: 1) a cross-model consistency checking strategy that leverages diverse model architectures and prompt formulations to reveal fine-grained inaccuracies not detectable through single-model analysis, and 2) a targeted mitigation pipeline that applies precise corrections to identified problematic segments while preserving accurate content, avoiding the wholesale response reformulation typical of existing approaches.

Figure 1 presents an overview of FINCH-ZK. Given a prompt p, a target LLM T that generates response r_T, a set of sampler models $M = \{m_1, m_2, \ldots, m_{|M|}\}$, a judge model J, and an improver model I, FINCH-ZK performs hallucination detection and mitigation in three stages:

- Generate diverse samples from different sampler models
- Detect fine-grained inaccuracies in the input response using generated samples
- Perform systematic response improvement using detected inaccuracies and generated samples.

2.3 Cross-model Sample Generation

As the first component, FINCH-ZK generates diverse response samples through prompt variations and multi-model sampling to expose hallucinations that may be consistent within a single model but inconsistent across different architectures or prompt formulations.

The system applies a set of variations $V = \{v_1, v_2, \ldots, v_{|V|}\}$ to generate prompt variants

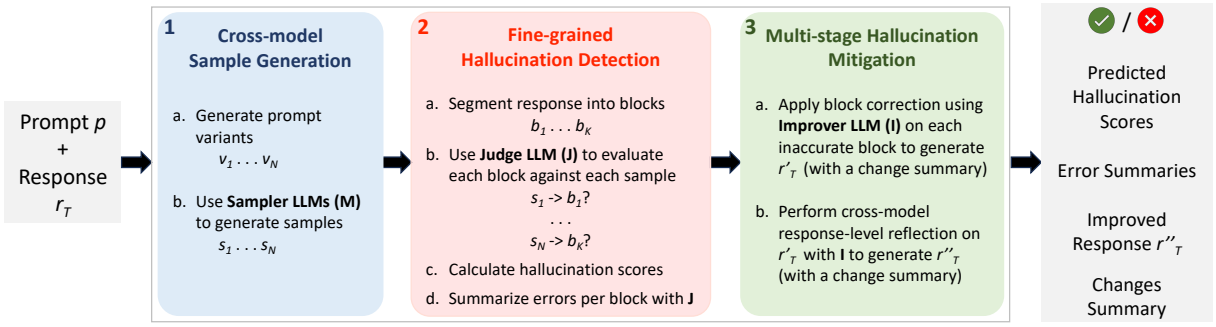

Figure 1: Overview of FINCH-ZK

$\{v_1(p), v_2(p), \ldots, v_{|V|}(p)\}$. These variations include syntactic transformations (rephrasing, expansion) and semantic modifications (chain-of-thought prompting, question decomposition) designed to elicit varied reasoning patterns while preserving the original information requirements (Appendix A.2).

The system then collects $|S|$ responses by prompting different sampler models in M with different variants from V to create the sample set $S = \{s_1, s_2, \ldots, s_{|S|}\}$. Each sample $s_i \in S$ is generated by randomly selecting a prompt variant $v_i \in V$ and sampler model $m_i \in M$. This cross-model sampling strategy captures architectural differences in reasoning patterns, knowledge representation, and potential systematic biases that single-model approaches cannot detect.

2.4 Fine-grained Hallucination Detection

As the second component, FINCH-ZK performs fine-grained hallucination identification through automated cross-consistency evaluation, enabling precise localization of potentially hallucinated content segments.

Response Segmentation. Target response r_T is segmented into semantic blocks $B = \{b_1, b_2, \ldots, b_{|B|}\}$ using sentence-level segmentation. This granular approach enables the system to identify specific hallucinated segments rather than classifying entire responses, providing actionable feedback for targeted correction.

Cross-consistency Evaluation. Each block $b_i \in B$ is evaluated against each sample $s_j \in S$ using the judge model J with structured prompts. The judge model classifies each (b_i, s_j) pair into:

- `ACCURATE`: Block is factually consistent and supported by the sample
- `CONTRADICTION`: Direct factual contradiction detected between block and sample
- `NEUTRAL`: Insufficient information for definitive assessment.

Weighted Scoring. Block-level hallucination scores are computed using weighted aggregation across all samples:

$$\text{score}(b_i) = \frac{\sum_{j=1}^{|S|} w_j(b_i) \cdot \text{score}(b_i, s_j)}{\sum_{j=1}^{|S|} w_j(b_i)}$$

where $w_j(b_i)$ represents the reliability weight assigned to sample s_j for evaluating block b_i and $\text{score}(b_i, s_j) \in \{0, 0.5, 1\}$ corresponds to `ACCURATE`, `NEUTRAL`, and `CONTRADICTION` labels respectively. Factuality labels are assigned to each block b_i based on a threshold τ as: `ACCURATE` for $\text{score}(b_i) \in [0, \tau]$, `CONTRADICTION` for $\text{score}(b_i) \in [1 - \tau, 1]$, and `NEUTRAL` otherwise. Response-level hallucination score is computed as $\text{score}(r_T) = \frac{1}{|B|} \sum_{i=1}^{|B|} \text{score}(b_i)$, with overall response label computed as: `NON-FACTUAL` if any block in r_T is labelled as `CONTRADICTION`, and `FACTUAL` otherwise.

Summarize Errors. For blocks identified as potentially hallucinated (i.e., labeled as `CONTRADICTION` or `NEUTRAL`), the system generates concise error summaries e_i using the judge model to characterize the nature and severity of detected inconsistencies, providing interpretable explanations for actionable user feedback and downstream correction.

2.5 Multi-stage Hallucination Mitigation

The mitigation stage applies targeted corrections to identified problematic segments through a two-stage approach: fine-grained block correction followed by response-level coherence improvement.

Block-level Correction. For each hallucinated block b_i with error summary e_i, FINCH-ZK generates a corrected version b'_i using an improver model I with a structured correction prompt that includes: 1) the original block text, 2) the automatically generated error summary, and 3) detailed contradiction evidence from the cross-consistency analysis. This approach ensures corrections are

grounded in specific identified issues rather than generic reformulation. The corrected response is reconstructed as:

$$r'_T = \text{concat}(c_1, c_2, \ldots, c_{|B|})$$

where $c_i = b'_i$ if block i was flagged for correction, and $c_i = b_i$ otherwise. This selective correction strategy preserves accurate content while targeting only problematic segments.

Response-level Improvement. To address broader coherence and completeness issues that may arise from localized corrections, the system performs cross-model reflection by generating an improved response r''_T that synthesizes insights from all generated samples S. The improver model receives the original prompt, the block-corrected response r'_T, and representative samples from S to produce a final response that maintains factual accuracy while ensuring overall coherence and completeness.

This multi-stage approach addresses the key limitation of existing mitigation systems that apply wholesale reformulation, often corrupting accurate content while attempting to fix errors. By preserving the accurate segments, FINCH-ZK provides targeted correction that maintains response quality while eliminating hallucinations.

2.6 Production Deployment Features

FINCH-ZK includes enterprise-ready capabilities:

- **Modular Architecture:** Pluggable components for sampler, judge, and improver models enable seamless integration with LLM infrastructure.
- **Performance Optimization:** Multi-threaded processing with configurable parallelism and batch judgment option to reduce API calls.
- **Monitoring & Observability:** Comprehensive CSV logging, per-block explanations, and correction summaries for audit trails.
- **Flexible Configuration:** CLI interface with adjustable sample counts, model selection, etc.

3 Experiments

We conducted experimental evaluation to answer:

RQ1: How effective is FINCH-ZK at detecting hallucinations compared to other approaches?

RQ2: How effective is FINCH-ZK for mitigating hallucinations?

RQ3: Which components significantly influence FINCH-ZK's detection capabilities?

RQ4: How does different factors affect FINCH-ZK's hallucination mitigation?

RQ5: What are the computational trade-offs of FINCH-ZK in terms of latency and cost?

Datasets. We utilize two challenging benchmarks for evaluation: 1) FELM (Zhao et al., 2023) composed of 847 questions & responses across diverse domains supplemented with fine-grained human-annotated factuality labels, and 2) GPQA-diamond (Rein et al., 2024) composed of 198 graduate-level multiple-choice questions.

Baseline Methods. For *RQ1* & *RQ2*, we compared against GPT4-based judge variants (Vanilla, CoT, RAG) as utilized in (Zhao et al., 2023) and SelfCheckGPT (Manakul et al., 2023). For *RQ2* & *RQ4*, we compared against SelfCheckGPT and hallucination mitigation techniques: few-shots CoT (Wei et al., 2023) using 5 in-context examples, self-consistency (Wang et al., 2023), a cross-model variant of self-consistency that uses multiple LLMs (call it cross-consistency), and best-of-N majority selection (Lightman et al., 2023). For a fair comparison, we used equivalent configurations across different techniques (Appendix A.5) and added equivalent engineering upgrades (Appendix A.3) to SelfCheckGPT.

RQ1: How effective is FINCH-ZK at detecting hallucinations compared to other approaches?

Table 1 presents results for hallucination detection on the FELM dataset. At both fine-grained (i.e., sentence) as well as aggregated response level, FINCH-ZK showed better precision-recall balance, consistently outperforming all baselines. In particular, FINCH-ZK achieved 39% better sentence-level F1-score compared to GPT4-Judge (Vanilla). Surprisingly, FINCH-ZK even outperformed the knowledge-source dependent RAG-based baseline, achieving around 17% better F1-score and 8% better balanced accuracy respectively at response level. Diverse sample generation through prompt variations and cross-model sampling enabled FINCH-ZK to achieve around 6% better F-1 scores and Pearson correlation compared to SelfCheckGPT.

RQ2: How effective is FINCH-ZK for mitigating hallucinations?

Table 2 presents a comparison of FINCH-ZK against different mitigation baselines on GPQA-diamond dataset. We evaluate performance using three distinct judging methodologies: regex-based

Table 1: Comparison of hallucination detection methods on FELM (Zhao et al., 2023) dataset. P/R/F1/BA, respectively, denote precision, recall, F-1 score, and balanced accuracy of predicted factuality labels vs human-annotations. For response-level, we additionally show Pearson and Spearman correlations of predicted hallucination scores. GPT-4 Judge variants are from (Zhao et al., 2023) that use GPT-4 for judgment based on the prompt and sentence directly (Vanilla), with chain-of-thought (CoT), or with retrieved content from reference sources (RAG). Delta percentages are shown for F1/BA metrics, with positive values indicating improvements compared to GPT-4 Judge (Vanilla).

Method	Sentence-level				Response-level					
	P	R	F1 (Δ%)	BA (Δ%)	P	R	F1 (Δ%)	BA (Δ%)	Pearson	Spearman
GPT-4 Judge (Vanilla)	64.0	24.4	35.4	60.7	62.4	39.4	48.3	63.8	—	—
GPT-4 Judge (CoT)	68.1	30.4	42.0 (+18.6%)	63.7 (+4.9%)	64.7	46.1	53.8 (+11.4%)	66.8 (+4.7%)	—	—
GPT-4 Judge (RAG)	62.9	39.2	48.3 (+36.4%)	67.1 (+10.5%)	64.3	51.1	56.9 (+17.8%)	68.5 (+7.4%)	—	—
SelfCheckGPT	41.2	54.1	46.8 (+32.2%)	68.7 (+13.2%)	73.7	53.5	62.0 (+28.4%)	72.0 (+12.9%)	59.5	59.9
FINCH-ZK	45.8	53.1	**49.2** (+39.0%)	**69.8** (+15.0%)	83.8	53.2	**65.1** (+34.8%)	**74.0** (+16.0%)	**63.1**	**61.5**

answer-choice accuracy, RAG-based LLM judging of the full response, and FINCH-ZK's based judge.

In answer-choice accuracy, FINCH-ZK achieved the best performance—reaching ∼76% accuracy, up +5.6% for Claude 4 Sonnet and +12.6% for Llama 4 Maverick. For full response accuracy, FINCH-ZK outperformed the next best baseline (SelfCheckGPT) by around 9-15% for RAG-based judging and 7-9% for FINCH-ZK-based judge.

These results demonstrate that FINCH-ZK's combination of cross-model sampling, fine-grained error detection, and targeted correction offers superior hallucination mitigation compared to existing approaches. The system is particularly effective at improving full response factuality, as evidenced by the substantial gains in RAG-based and FINCH-ZK-based judging metrics. The effectiveness across different model families (Claude and Llama) highlights FINCH-ZK's model-agnostic design, making it a versatile solution for production environments.

Notably, while Self-Consistency and Best-of-N offer modest improvements in answer-choice accuracy (2-3%), they often fail to meaningfully improve full response factuality. This underscores the limitations of approaches that don't explicitly target hallucinations at a fine-grained level.

RQ3: Which components significantly influence FINCH-ZK's detection capabilities?

Table 3 summarizes ablation studies to understand the influence of each component in FINCH-ZK for hallucination detection. Key observations include:

- Detection capabilities do not monotonically increase with more samples (G1.a-c vs G0).
- Disabling cross-model sampling (i.e., all samples generated with Claude 4 Sonnet) degraded detection at the response-level (G2.a vs G0).
- Adding additional cross-model sampler LLMs, both weaker models (G2.b adds Claude 3.5 Sonnet and Llama 4 Scout) or stronger models (G2.c adds Claude 4 Opus), improves detection.
- Using a coarse, response-level judge significantly limits detection due to poor recall (G3.a vs G0).
- Using a single judge query to evaluate all blocks

Table 2: Comparison of hallucination mitigation methods on the GPQA-diamond (Rein et al., 2024) dataset. All methods are evaluated against the same zero-shot CoT baseline. Positive delta percentages indicate relative improvements compared to the baseline. Regex-based judge matches answer choice against ground truth, RAG-based judge uses answer explanations from the dataset as trusted content for LLM-based judgment, FINCH-ZK-based judge is based on §2.4.

T	Method	Regex-based Judge		RAG-based LLM Judge		FINCH-ZK Judge	
		Answer Acc.	Δ%	Full Resp. Acc.	Δ%	Full Resp. Acc.	Δ%
Claude 4 Sonnet	Zero-shot CoT (baseline)	71.7	—	50.0	—	69.7	—
	Few-shots-CoT	68.2	-4.9%	47.5	-5.1%	70.7	+1.4%
	Self-Consistency	73.2	+2.1%	48.5	-3.0%	66.7	-4.3%
	Cross-Consistency	71.2	-0.7%	52.5	+5.1%	71.2	+2.2%
	Best-of-N	73.7	+2.8%	52.5	+5.1%	69.7	0.0%
	SelfCheckGPT	71.2	-0.7%	54.5	+9.1%	75.3	+8.0%
	FINCH-ZK	**75.8**	**+5.6%**	**59.1**	**+18.2%**	**80.3**	**+15.2%**
Llama 4 Maverick	Zero-shot CoT (baseline)	68.2	—	42.9	—	63.1	—
	Few-shots-CoT	67.7	-0.7%	43.4	+1.2%	64.7	+2.4%
	Self-Consistency	67.7	-0.7%	45.0	+4.7%	64.1	+1.6%
	Cross-Consistency	73.7	+8.2%	50.5	+17.7%	69.7	+10.4%
	Best-of-N	67.2	-1.5%	41.9	-2.4%	61.1	-3.2%
	SelfCheckGPT	75.8	+11.1%	84.3	+96.5%	86.9	+37.6%
	FINCH-ZK	**76.8**	**+12.6%**	**90.9**	**+111.8%**	**92.4**	**+46.4%**

Table 3: Ablation studies for hallucination detection on FELM (Zhao et al., 2023) dataset. Group G1 shows the effect of changing number of samples (§2.3), G2 compares the effect of changing sampler LLMs (§2.3), G3 shows the effect of changing LLM-based judge (§2.4).

Group	Configuration	Sentence-level				Response-level					
		P	R	F1	BA	P	R	F1	BA	Pearson	Spearman
G0	FINCH-ZK	45.8	53.1	49.2	69.8	83.8	53.2	65.1	74.0	63.1	61.5
G1	a. 3 samples	46.0	52.0	48.8	69.4	77.6	55.3	64.6	73.7	57.8	55.5
	b. 5 samples	43.1	59.5	50.0	71.2	78.7	57.8	66.7	75.0	61.9	59.8
	c. 20 samples	48.0	54.5	51.0	70.9	81.2	53.5	64.5	73.7	63.7	61.5
G2	a. (-) cross-model sampling	43.2	55.3	48.5	69.8	77.2	51.8	62.0	72.1	62.1	59.8
	b. (+) weak samplers	46.6	56.0	50.9	71.1	81.6	53.5	64.7	73.8	63.0	60.7
	c. (+) strong samplers	46.3	56.2	50.7	71.0	79.5	53.5	64.0	73.3	63.2	62.2
G3	a. (-) fine-grained judge	—	—	—	—	88.1	31.6	46.5	64.7	58.7	59.7
	b. (+) use batch judge	37.2	72.2	49.1	72.9	69.8	73.8	71.7	78.9	63.9	61.5
	c. Llama 4 Scout judge	39.6	81.4	53.3	77.3	72.5	80.5	**76.3**	82.6	**71.2**	**67.9**
	d. Llama 4 Scout batch judge	35.5	83.2	49.8	75.3	69.4	84.4	76.2	**82.9**	65.5	64.6
	e. Claude 4 Sonnet batch judge	41.5	86.7	**56.1**	**80.1**	65.6	85.8	74.3	81.7	69.3	66.9

together in a batch (instead of separate LLM calls for each sample-block pair) is an effective way to reduce LLM costs without compromising detection performance (G3.b vs G0).
- Judge model can significantly influence detection performance (G3.c-e vs G0).

RQ4: How does different factors affect FINCH-ZK's hallucination mitigation?

Table 4 summarizes ablation studies to understand mitigation effectiveness using Claude 4 Sonnet as the target model. Key observations include:

- FINCH-ZK typically reaches higher accuracy with more samples, though with diminishing returns (G1.a-c vs G0).
- Disabling cross-model sampling degrades mitigation capability significantly (G2.a vs G0).
- Using coarse response-level judge reduced accuracy improvements significantly (G3.a vs G0).
- Judge variations have modest effects on answer-choice accuracy, but significant impact on full response accuracy (G3.b-d vs G0).
- Disabling fine-grained correction drastically limits its mitigation performance, underscoring the importance of targeted correction (G4.a vs G0).
- Using Llama 4 Maverick as the improver LLM (instead of Claude 4 Sonnet) significantly improved full response accuracy, suggesting cross-model reflection can help remedy perpetuating biases and reasoning patterns inherent in single-model architectures (G4.b vs G0).
- FINCH-ZK boosts accuracy even with extended thinking enabled, achieving 80.3% answer-choice accuracy (+11.3% over extended thinking

Table 4: Ablation studies for hallucination mitigation on GPQA-diamond (Rein et al., 2024) dataset. Group G1 shows the effect of changing number of samples (§2.3), G2 compares the effect of changing sampler LLMs (§2.3), G3 shows the effect of changing LLM-based judge (§2.4), G4 shows the effect of changing multi-stage mitigation (§2.5), G5 shows the comparison with extended thinking enabled. Delta percentages indicate improvement compared to zero-shot CoT baseline.

Group	Configuration	Regex-based Judge		RAG-based LLM Judge		FINCH-ZK Judge	
		Answer Acc.	Δ%	Full Resp. Acc.	Δ%	Full Resp. Acc.	Δ%
G0	a. Zero-shot CoT (baseline)	71.7	—	50.0	—	69.7	—
	b. FINCH-ZK	75.8	+5.6%	59.1	+18.2%	80.3	+15.2%
G1	a. 3 samples	69.7	-2.8%	54.0	+8.1%	72.2	+3.6%
	b. 5 samples	71.2	-0.7%	61.1	+22.2%	76.8	+10.1%
	c. 20 samples	**78.8**	**+9.9%**	59.6	+19.2%	77.8	+11.6%
G2	a. (-) cross-model sampling	71.7	0.0%	57.6	+15.2%	74.2	+6.5%
	b. (+) weak samplers	72.7	+1.4%	56.1	+12.1%	75.3	+8.0%
	c. (+) strong samplers	75.8	+5.6%	56.1	+12.1%	76.3	+9.4%
G3	a. (-) fine-grained judge	74.2	+3.5%	56.6	+13.1%	77.3	+10.9%
	b. (+) use batch judge	75.8	+5.6%	56.6	+13.1%	79.3	+13.8%
	c. Llama 4 Scout judge	74.2	+3.5%	56.6	+13.1%	82.8	+18.8%
	d. Claude 4 Sonnet batch judge	74.2	+3.5%	56.1	+12.1%	78.8	+13.0%
G4	a. (-) fine-grained correction	72.7	+1.4%	50.5	+1.0%	76.8	+10.1%
	b. Llama 4 Maverick improver	74.8	+4.2%	**90.4**	**+80.8%**	**94.4**	**+35.5%**
G5	a. (+) thinking (baseline)	72.2	—	**65.7**	—	82.3	—
	b. (+) thinking (FINCH-ZK)	**80.3**	**+11.3%**	64.7	-2.0%	**90.4**	**+11.6%**

Table 5: Latency and cost analysis for hallucination detection and mitigation on GPQA-diamond dataset with Claude 4 Sonnet as the target model. Overhead factors are computed relative to the zero-shot CoT baseline. Grouped rows summarize latency and cost for response generation (G0), for hallucination detection (G1), and for both hallucination detection and mitigation (G2).

Group	Method	Latency (sec)	Cost (USD)	Latency Overhead	Cost Overhead
Response generation					
G0	a. Zero-shot CoT (baseline)	11.1	0.0096	1.0×	1.0×
	b. Zero-shot CoT w/ extended thinking	36.6	0.0165	3.3×	1.7×
Detection only					
G1	a. SelfCheckGPT (10 samples)	12.2	0.2777	1.1×	28.9×
	b. FINCH-ZK (10 samples)	19.0	0.3488	1.7×	36.3×
	c. FINCH-ZK (3 samples)	19.2	0.1203	1.7×	12.5×
	d. FINCH-ZK (10 samples, batch judge)	28.5	0.1709	2.6×	17.8×
	e. FINCH-ZK (3 samples, batch judge)	24.5	0.0780	2.2×	8.1×
Detection + Mitigation					
G2	a. SelfCheckGPT (10 samples)	26.7	0.3113	2.4×	32.4×
	b. FINCH-ZK (10 samples)	37.9	0.3877	3.4×	40.4×
	c. FINCH-ZK (3 samples)	35.8	0.1537	3.2×	16.0×
	d. FINCH-ZK (10 samples, use batch judge)	48.7	0.2361	4.4×	24.6×
	e. FINCH-ZK (3 samples, use batch judge)	39.2	0.1182	3.5×	12.3×

baseline). This demonstrates our proposed techniques *complements internal extended reasoning*, rather than competing with it (G5.b vs G5.a).

***RQ5*: What are the computational trade-offs of FINCH-ZK in terms of latency and cost?**

Table 5 presents a comparative analysis of computational overhead to quantify latency and costs on GPQA-diamond dataset using Claude 4 Sonnet as the target LLM. Key observations include:

- FINCH-ZK incurs significantly higher costs than the response generation baseline and with extended thinking (G0 a-b vs G1.b and G2.b). FINCH-ZK also incurs higher costs than SelfCheckGPT (G1.a vs G1.b), primarily due to the additional prompt variations step.
- FINCH-ZK's detection plus mitigation latency matches the latency of response generation with extended thinking (G0.b vs G2.b) while providing explicit hallucination detection and targeted corrections (rather than opaque internal reasoning) and superior accuracy (Table 4).
- Reducing samples to 3 and enabling batch judge maintains accuracy (Table 4) while reducing latency and cost overhead (G2.b vs G2.c-e).

4 Human Evaluation Study

To further validate our automated evaluation, we conducted a human evaluation study on the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019).

Setup. We randomly sampled 50 questions from NQ and generated responses using Claude 4 Sonnet. Each response was processed through FINCH-ZK (3 samples, batch judge configuration) for hallucination detection and mitigation to produce an improved version. Three independent human annotators performed blind pairwise comparisons between original and improved responses, with access to external resources for fact-checking.

Results. Human annotators preferred FINCH-ZK improved responses in 84% (42/50) of cases, demonstrating strong alignment with our automated evaluation from Table 2. The improved responses averaged 229 tokens compared to 153 tokens for originals, indicating that FINCH-ZK adds substantive content rather than merely reformulating. Annotators noted improvements in comprehensiveness, accuracy, and factual detail, while occasionally preferring originals when additions seemed excessive for straightforward queries.

The human study affirmed FINCH-ZK's mitigation represent genuine improvements in response quality, not artifacts of circular LLM evaluation.

5 Conclusions

We introduce FINCH-ZK, an integrated black-box framework that closes the gap between hallucination detection and mitigation by combining advanced detection techniques with a novel multi-stage process for targeted correction. Leveraging dynamic prompt variations and cross-model consistency, FINCH-ZK delivers significantly more robust detection than single-model approaches. Its multi-stage mitigation pipeline makes precise, segment-level corrections while maintaining overall coherence. The system's performance scales with computational resources, but even resource-efficient configurations offer substantial improvements.

Limitations

While FINCH-ZK represents a meaningful step toward improving the reliability of large language model outputs, it is not without important limitations. The underlying approach fundamentally relies on the assumption that a truly reliable answer will emerge as the most frequent or stable across repeated sampling. However, for complex or ambiguous queries, models may consistently reproduce similar hallucinated content, leading to a false sense of confidence in its correctness. In such cases, consistency can inadvertently reinforce errors rather than expose them. In domains requiring absolute certainty, human oversight remains essential. Additionally, the computational overhead of generating multiple cross-model samples represents a substantial cost increase compared to alternative approaches, which may limit real-time applications.

Future work includes exploring extensions like—1) extending hallucination detection and mitigation for languages beyond English, 2) domain-specific applications including code generation and medical QA, 3) streaming/real-time scenarios with incremental correction 4) exploring paragraph- and page-level segmentation for very long-form responses, 5) reducing computational costs with batch processing, 6) rigorous evaluation on domain-specific benchmarks, 7) adversarial robustness evaluation against prompt injection attacks, and 8) investigating solutions for agentic applications.

Ethics Statement

Our cross-model approach assumes that consensus among different models indicates accuracy, but this may amplify shared biases across model families rather than eliminate them. We emphasize that FINCH-ZK should complement, not replace, human oversight in high-stakes applications. Users must understand the system's limitations and maintain appropriate skepticism of AI-generated content. We encourage responsible deployment with clear communication about the system's capabilities and limitations to end users.

Acknowledgments

We would like to thank Doug Terry, Leah Daniels, and Bedrock Science teams at AWS for their support for this work. We would like to thank anonymous EMNLP reviewers for their detailed reviews and helpful feedback. Additionally, we would like to extend our thanks to the open community for their invaluable contributions.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Nipun Sadvilkar and Mark Neumann. 2020. Pysbd: Pragmatic sentence boundary disambiguation. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 110–114.

Jinxin Shi, Jiabao Zhao, Xingjiao Wu, Ruyi Xu, Yuan-Hao Jiang, and Liang He. 2025. Mitigating reasoning hallucination through multi-agent collaborative filtering. *Expert Systems with Applications*, 263:125723.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Yiran Zhao, Jinghan Zhang, I Chern, Siyang Gao, Pengfei Liu, Junxian He, and 1 others. 2023. Felm: Benchmarking factuality evaluation of large language models. *Advances in Neural Information Processing Systems*, 36:44502–44523.

A Appendix

A.1 Additional Related Works

Uncertainty Quantification and Calibration Methods. Recent work on uncertainty quantification includes LLM self-calibration (Kadavath et al., 2022) and semantic or natural language uncertainty quantification (Kuhn et al., 2023; Lin et al., 2022). FINCH-ZK differs from these works by using cross-model consistency rather than single-model uncertainty, but can be enhanced by utilizing model-derived confidence scores.

Constitutional AI and Self-Correction Approaches. Constitutional AI (Bai et al., 2022), Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023), and other self-correction methods train models to critique and improve their own outputs through principles or self-reflection. FINCH-ZK complements these approaches by providing external cross-model validation, avoiding single-model architectures from perpetuating their own systematic biases and reasoning patterns.

Alternative Hallucination Detection Paradigms. Alternative detection methods include knowledge graph integration (Petroni et al., 2019), detection using internal representations (Chen et al., 2024), and sophisticated retrieval-augmented verification (Nakano et al., 2021) that operate with differ-

Prompt: Write a short passage about Leslie Lamport.

Response: Leslie Lamport is a renowned computer scientist and mathematician who has made significant contributions to the field of distributed computing. He is best known for his work on the development of the Paxos algorithm, a fundamental protocol for achieving consensus in distributed systems. Lamport was born in 1941 in New York City and received his Ph.D. in mathematics from the University of California, Berkeley in 1972. He has worked at various research institutions, including the Massachusetts Institute of Technology (MIT), the Digital Equipment Corporation, and Microsoft Research. Throughout his career, Lamport has been recognized for his groundbreaking work in distributed systems. He is the recipient of numerous prestigious awards, including the Turing Award, often referred to as the "Nobel Prize of Computing," which he received in 2013 for his contributions to the theory and practice of distributed and concurrent systems. Lamport's work has had a profound impact on the development of modern computer systems, particularly in areas such as cloud computing, blockchain technology, and the Internet of Things. His ideas and algorithms have been widely adopted and have become essential building blocks for many of the technologies we rely on today. Beyond his technical contributions, Lamport is also known for his clear and concise writing style, which has helped to make complex concepts in computer science more accessible to a wider audience. He is the author of several influential books and papers that have shaped the field of distributed computing. Overall, Leslie Lamport's legacy as a pioneering computer scientist and mathematician continues to inspire and influence the next generation of researchers and developers working in the field of distributed systems.

Fine-grained Detection: NON-FACTUAL with 0.10 hallucination score (10/11 blocks factual)

Possible Inaccuracies:

Block 3: "Lamport was born in ..." - The passage contains an inaccuracy regarding the university where Lamport received his Ph.D. in mathematics, which was Brandeis University, not the University of California, Berkeley.

Multi-stage Mitigation:

Fine-grained Correction:

Block 3: "Lamport was born in ..." - Lamport was born in 1941 in New York City and received his Ph.D. in mathematics from Brandeis University in 1972.

Cross-model Reflection: **FACTUAL** with 0.06 hallucination score (12/12 blocks factual)

Leslie Lamport is a renowned computer scientist and mathematician who has made groundbreaking contributions to distributed systems, concurrent computing, and formal verification. Born on February 7, 1941, in New York City, Lamport received his B.S. in mathematics from the Massachusetts Institute of Technology (MIT) in 1960 and his Ph.D. in mathematics from Brandeis University in 1972.

Lamport is best known for his work on the development of the Paxos algorithm, a fundamental protocol for achieving consensus in distributed systems. He also introduced the concept of logical clocks and the happens-before relation, which help solve the challenging problem of ordering events across distributed networks. His 1978 paper, "Time, Clocks, and the Ordering of Events in a Distributed System," laid the theoretical foundations for understanding causality in distributed systems.

In addition to his work on distributed systems, Lamport developed LaTeX, a document preparation system built on Donald Knuth's TeX, which revolutionized academic publishing by enabling precise typesetting of complex mathematical notation. He also created TLA+ (Temporal Logic of Actions), a formal specification language used to model and verify complex software and hardware systems.

Lamport's contributions to computer science have been recognized with numerous prestigious awards, including the Turing Award in 2013, often referred to as the "Nobel Prize of Computing," for his contributions to the theory and practice of distributed and concurrent systems. He is also a recipient of the Dijkstra Prize, the IEEE Emanuel R. Piore Award, and the IEEE John von Neumann Medal.

Throughout his career, Lamport has worked at various research institutions, including SRI International, Digital Equipment Corporation, and Microsoft Research, where he is currently a Principal Researcher. His work continues to underpin critical technologies from cloud computing to blockchain systems, making him one of the most influential figures in computer science. Lamport's legacy endures through his emphasis on precise thinking, mathematical rigor, and elegant solutions to complex problems, inspiring new generations of researchers and practitioners in the field.

Changes Summary:

The initial response provided a good overview of Leslie Lamport's contributions to computer science, but it lacked some details and accuracy found in the provided contexts. Specifically, it did not mention his work on the Byzantine Generals' Problem, his development of TLA+, or his extensive list of awards. The response also contained some minor inaccuracies, such as his educational background. The improved response aims to provide a more comprehensive and accurate account of Lamport's achievements and contributions.

Figure 2: Motivating Example

ent levels of model access and external knowledge. FINCH-ZK complements these approaches for practical deployment necessitating zero-knowledge and black-box constraints.

Additional relevant work includes ensemble methods (Chen et al., 2023), contrastive decoding techniques (Li et al., 2022), multi-agent verification systems (Shi et al., 2025), and domain-specific benchmarks (Chen et al., 2021), which represent opportunities for future enhancements and evaluation of this work.

A.2 Prompt Variations

FINCH-ZK employs three static and four dynamic LLM-based prompt reformulation strategies to probe different aspects of the model's knowledge and reasoning:

- **Static Variations**.
  1. **Identity**: Uses the original prompt unchanged as a baseline.
  2. **Zero-shot CoT**: Appends "Let's think step by step" to encourage structured reasoning.
  3. **Long Answer**: Requests detailed responses to reveal potential inconsistencies by adding "Provide an answer with at least a 1000 words to the following prompt:" at the beginning of the original prompt.
- **LLM-based Variations**.
  1. **Rephrase**: Use an LLM to generate semantically-equivalent reformulations of the original prompt (Fig. 3).
  2. **Expand-Before**: Use an LLM to add contextual information before the original prompt (Fig. 4).
  3. **Expand-After**: Use an LLM to add clarifying questions after the original prompt (Fig. 5).
  4. **Break-Down**: Use an LLM to break down complex queries into multiple sub-questions (Fig. 6).

These prompt variations are designed to systematically explore the response space without changing the semantic intent of the original query.

A.3 Engineering Upgrades

FINCH-ZK adds a collection of engineering upgrades to improve the effectiveness and ease of usage, as follows:

- *Improved prompt for LLM-based judgment*. For fine-grained consistency-based hallucination detection (§ 2.4), FINCH-ZK upgrades the LLM-as-judge prompt used in SelfCheckGPT (Manakul et al., 2023) by adding— 1) systematic structure, 2) contextual information, 3) descriptive rules, and 4) output format instructions (Fig. 7).
- *Batch LLM-based judgment.* For efficiency and computation costs savings, we implemented an option for judging all blocks against a sample in a single LLM query (Fig. 8).
- *Multi-threading support*. FINCH-ZK supports efficient multi-threading with configurable parallelism support. FINCH-ZK evaluates multiple responses concurrently, with each response evaluation utilizing multiple threads for each component (sample generation, fine-grained block evaluation, and fine-grained block correction).
- *Usability upgrades*. FINCH-ZK provides improved command-line interface, comprehensive logging through CSV outputs, statistics summary, results reporting, judgement explanations, and response changes summary to enhance usage experience and results analysis.

A.4 Implementation Details

We implemented FINCH-ZK in ∼2.3K lines of code in Python. The framework utilizes a mix of Claude 4 Sonnet, Llama 4 Maverick, Claude 3.7 Sonnet and DeepSeek-R1 (Jan'25) for generating 10 cross-model samples by default. FINCH-ZK utilizes PySBD (Sadvilkar and Neumann, 2020) to segment the response into fine-grained blocks at sentence boundaries. For fast fine-grained hallucination assessment, FINCH-ZK uses Claude 3 Haiku as the default judge model. By default, FINCH-ZK uses the same model as the one used for generating input responses for multi-stage hallucination mitigation.

FINCH-ZK provides command-line options to easily change key hyper parameters, including different LLMs used in the framework. Here is a summary of hyper parameter values we used as defaults:

- *Models*
  - Prompt reformulation model: Claude 3 Sonnet
  - Sampler models: Claude 4 Sonnet, Llama 4 Maverick, Claude 3.7 Sonnet, DeepSeek R1 (Jan'2025)
  - Judge model: Claude 3 Haiku
  - Improver model: Same as target model
- *Hyperparameters*
  - Temperature: 0.0 for input response generation, 1.0 otherwise
  - Max output tokens: 4096

– Block labeling threshold τ: 0.33
– Batch LLM-based judgment: disabled

Scoring Function Details. The reliability weights $w_j(b_i)$ in the block-level scoring function are determined by the confidence level of each judge evaluation. Specifically, the system assigns fixed weights based on the judge's classification: `ACCURATE` evaluations receive weight 2, `NEUTRAL` evaluations receive weight 1, `CONTRADICTION` evaluations receive weight 4, and `UNKNOWN` evaluations receive weight 0 (excluded from aggregation). This weighting scheme prioritizes high-confidence contradictions while still incorporating uncertain evaluations, computed as:

$$w_j(b_i) = \begin{cases} 4 & \texttt{CONTRADICTION} \\ 2 & \texttt{ACCURATE} \\ 1 & \texttt{NEUTRAL} \\ 0 & \texttt{UNKNOWN} \end{cases}$$

Threshold Selection. The block labeling thresholds $\tau = 0.33$ and $1 - \tau = 0.67$ create three confidence intervals for factuality assessment: [0, 0.33] for `ACCURATE`, (0.33, 0.67) for `NEUTRAL`, and [0.67, 1.0] for `CONTRADICTION`. These values were empirically determined to provide balanced precision-recall performance across both FELM and GPQA datasets. The symmetric threshold design ensures that high-confidence accurate and contradictory content receive equal treatment, while the middle range captures genuinely ambiguous cases where samples provide conflicting evidence about a block's factuality.

Cross-model Sampling Strategy. The sampler model set M consists of diverse LLM architectures selected to maximize reasoning diversity while maintaining practical computational constraints. By default, M includes four models representing different architectural families: Claude 4 Sonnet (transformer-based), Llama 4 Maverick (open-source transformer), Claude 3.7 Sonnet (earlier generation), and DeepSeek-R1 (reasoning-specialized). This selection balances three criteria: 1) architectural diversity to capture different reasoning patterns, 2) performance quality to ensure reliable samples, and 3) API availability for practical deployment.

The relationship between prompt variations V and model selection follows a structured round-robin approach rather than pure randomization. For each prompt, both the prompt variations set and sampler models set are shuffled once, then samples are generated by cycling through these shuffled lists: sample s_i uses prompt variant $v_{i \bmod |V|}$ and sampler model $m_{i \bmod |M|}$. With 7 variants and 4 models generating 10 samples by default, this ensures comprehensive coverage of variant-model combinations while maintaining deterministic reproducibility when random seeds are fixed.

This systematic assignment addresses variance concerns in two ways: 1) the initial shuffle provides randomization benefits without introducing per-sample variance, and 2) the round-robin cycling ensures that all (model, prompt variant) combinations are explored fairly across the sample set. Empirical analysis shows this approach produces more stable cross-model consistency scores compared to fully random assignment, with standard deviation reduced by approximately 15% across repeated runs.

A.5 Experiment Details

Dataset Details. We selected FELM and GPQA-diamond as our evaluation benchmarks based on their unique characteristics that align with our zero-knowledge, fine-grained hallucination management objectives.

We chose FELM (Zhao et al., 2023) for hallucination detection evaluation for several key reasons:

- **Fine-grained annotations:** FELM provides segment-level (sentence-level) human-annotated factuality labels rather than coarse response-level labels, enabling evaluation of our fine-grained detection capabilities. The dataset contains 4,425 annotated segments across 847 samples with high inter-annotator agreement (91.3%).
- **Diverse domains:** Unlike benchmarks focused solely on world knowledge (e.g., FEVER, WICE), FELM spans five diverse domains that test different types of factual reasoning:
  – World Knowledge (184 samples, 532 segments) - including 11.1% from TruthfulQA
  – Science & Technology (125 samples, 683 segments) - scientific claims and citations
  – Writing & Recommendation (136 samples, 1,586 segments) - creative generation tasks
  – Reasoning (208 samples, 1,025 segments) - multi-step chain-of-thought traces
  – Mathematics (194 samples, 599 segments) - including 24.7% from GSM8K
- **Long-form complexity:** FELM responses average 89.1 tokens, substantially longer than existing benchmarks (FEVER: 7.3 tokens, FactCC:

20.8 tokens, HaluEval: 36.9 tokens). The Writing/Recommendation domain averages 210.9 tokens per response, representing true medium-to-long form generation where hallucinations are more challenging to detect.

- **Rich error taxonomy:** FELM categorizes errors into four types—knowledge errors, reasoning errors, irrelevant content, and fooled errors—providing detailed insights into hallucination patterns. Each segment includes error explanations and reference links for validation.

For mitigation, we selected GPQA-diamond (Rein et al., 2024) for evaluation due to its exceptional difficulty and objectivity:

- **Graduate-level complexity:** The dataset contains 198 multiple-choice questions requiring PhD-level expertise in biology, physics, and chemistry. Questions are designed to be answerable by domain experts (81.3% accuracy on diamond set) but extremely challenging for skilled non-experts (22.1% accuracy) even with unrestricted internet access.
- **Google-proof design:** Non-expert validators with PhDs in other domains spend an average of 37 minutes per question with full internet access (median: 30 minutes) yet achieve only marginally above random chance (25%). This ensures our mitigation evaluation tests genuine knowledge correction rather than simple information retrieval.
- **Rigorous validation:** Each question undergoes multi-stage expert validation with two PhD-level domain experts verifying correctness and objectivity. The diamond subset includes only questions where both experts agree and the majority of non-experts fail, ensuring uncontroversial ground truth.
- **Detailed explanations:** Questions include comprehensive explanations, enabling fine-grained analysis of reasoning improvements through our mitigation pipeline.

Together, these datasets provide comprehensive evaluation of FINCH-ZK's capabilities: FELM tests fine-grained detection across diverse content types using trusted human annotations, while GPQA-diamond validates our ability to improve factual accuracy on extremely challenging questions where even minor improvements represent significant achievements.

Metrics and Evaluation Methodology. For hallucination detection (*RQ1*, *RQ3*), we report precision, recall, F1-score, and balanced accuracy at both sentence-level and aggregated response levels to compare predicted factuality label against human annotations following established evaluation protocols from (Zhao et al., 2023). We additionally include Pearson and Spearman correlations between predicted and ground-truth hallucination scores.

For hallucination mitigation (*RQ2*, *RQ4*), we employ a multi-faceted evaluation approach with answer-choice accuracy as our primary metric, which objectively measures whether the model selects the correct multiple-choice option in GPQA questions. We supplement this with two additional judges: 1) *RAG-based LLM Judge* uses dataset reference explanations to assess full response reasoning quality, and 2) FINCH-ZK *Judge* applies our detection method to evaluate internal consistency within improved responses. The strong correlation between answer accuracy improvements and both supplementary LLM judge assessments provides convergent validity, though human evaluation on a representative subset would strengthen validation of our automated assessments. Delta percentages indicate relative improvements compared to the input response across all metrics.

Input Responses. For *RQ1* & *RQ3*, we evaluated using input responses already included in the FELM dataset (generated with ChatGPT). For *RQ3*, we generated input responses with Claude 4 Sonnet and Llama 4 Maverick as representative high-capability models where hallucination mitigation is critical. For *RQ4*, we used input responses corresponding to Claude 4 Sonnet from *RQ3* and additional input responses generated with Claude 4 Sonnet with $\sim$16K thinking budget tokens for ablations with extended thinking. While we conducted experiments with additional models from different LLM providers, the results are omitted due to licensing constraints and because they provided consistent findings that aligned with our main conclusions.

Baseline Details. We used equivalent configurations for each baseline like: 1) using 10 samples for self/cross-consistency and best-of-N, 2) same set of LLMs used for cross-model sampling in cross-consistency, 3) utilizing Claude 4 Sonnet for sample generation for self-consistency and SelfCheckGPT.

A.6 Human Evaluation Study Details

We conducted human evaluation following established practices from the NQ dataset (Kwiatkowski

et al., 2019). We selected NQ for human evaluation as it provides diverse factual queries with well-established ground truth, complementing our automated evaluation.

Dataset. 50 randomly sampled questions from Natural Questions

Annotators. 3 independent evaluators with graduate-level education

Task. Blind pairwise comparison with external resource access

Metrics. Binary preference with mandatory justification

Qualitative Analysis. Annotators consistently noted that FINCH-ZK improvements included:

- More comprehensive coverage of relevant facts
- Better structured explanations
- Additional context and examples
- Correction of factual errors or ambiguities

The 8 cases where originals were preferred typically involved simple factual queries where additional detail was deemed unnecessary.

A.7 Prompt Templates

Prompt Variation: Rephrase

Given a prompt, rephrase this prompt into an equivalent prompt to help in doing better answering. Maintain all information from the original prompt."

Here is the original prompt (inside <original> </original> tag):
<original>
{original}
</original>

Again, note that the new prompt should be equivalent to the original prompt. Provide the new prompt inside <new> </new> tags.

Figure 3: Prompt used in Rephrase prompt variation

Prompt Variation: Expand-Before

I need you to add sentences at the beginning of a given prompt. Note that you do not need to follow the instructions in the prompt. You are required to provide three sentences that can be added at the beginning of the prompt to help in doing better answering.

Here is the given prompt (inside <prompt> </prompt> tag):
<prompt>
{original}
</prompt>

Provide the sentences to be added at the beginning of the prompt inside <answer> </answer> tags.

Figure 4: Prompt used in Expand-Before prompt variation

Prompt Variation: Expand-After

I need you to add sentences at the end of a given prompt. Note that you do not need to follow the instructions in the prompt. You are required to provide three sentences that can be added at the end of the prompt to help in doing better answering.

Here is the given prompt (inside <prompt> </prompt> tag):
<prompt>
{original}
</prompt>

Provide the sentences to be added at the end of the prompt inside <answer> </answer> tags.

Figure 5: Prompt used in Expand-After prompt variation

Prompt Variation: Break-Down

Given a prompt, break down this prompt into multiple sub-prompts to help in doing better answering. Maintain all information from the original prompt."

Here is the original prompt (inside <original> </original> tag):
<original>
{original}
</original>

Again, note that the multiple sub-prompts together should be equivalent to the original prompt. List the multiple sub-prompts inside <sub-prompts> </sub-prompts> tags (one sub-prompt per line).

Figure 6: Prompt used in Break-Down prompt variation

LLM-as-Judge Prompt: Detection

I have a passage that is part of the response from a language model for answering a prompt. Please help me in fact-checking whether the passage is correct with respect to the provided reference.

Prompt (inside <prompt> </prompt> tags)
<prompt>
{prompt}
</prompt>

Full Response (inside <full-response> </full-response> tags)
<full-response>
{full_response}
</full-response>

Reference (inside <reference> </reference> tags)
<reference>
{sample.text}
</reference>

Passage (part of the full response, to be checked with respect to the reference) (inside <passage> </passage> tags)
<passage>
{block.text}
</passage>

Does the passage complies with the reference? Your answer should be "yes", "no", or "neutral".

Rules:
1. Your answer should be "no" if the passage contains a direct factual contradiction when compared against the reference.
2. Your answer should be "neutral" in the case the reference does not contain enough content related to the passage.
3. Otherwise, your answer should be "yes".

Provide a brief explanation in less than 30 words in <explain> </explain> tags. Then respond with your answer in <answer> </answer> tags (subject to the rules).

Figure 7: Prompt used in fine-grained LLM-based judgment to evaluate a block against a sample

LLM-as-Judge Prompt: Detection (Batch)

I have a prompt and a list of passages. All passages can be concatenated to form a complete answer to the prompt. Please help me in fact-checking whether these passages are correct with respect to the provided reference.

Prompt (inside <prompt> </prompt> tags)
<prompt>
{prompt}
</prompt>

Reference (inside <reference> </reference> tags)
<reference>
{sample.text}
</reference>

Passages (inside <passages> </passages> tags, represented as a JSON list as [{{"id": <passage-id>, "text": <passage-text>}}, ...])
<passages>
{block[1].text}
...
{block[K].text}
</passages>

For each passage, check whether the passage complies with the reference? Your answer should be "yes", "no", or "neutral", one each per passage.

Rules:
1. Your answer should be "no" if the passage contains a direct factual contradiction when compared against the reference.
2. Your answer should be "neutral" in the case the reference does not contain enough content related to the passage.
3. Otherwise, your answer should be "yes".

Include the answer (as "answer" field) and a brief explanation in less than 30 words (as "explanation" field) in your output.
Do not include "explanation" field for passages with answer as "yes" in your output.
Format your final output as a valid JSON list [{{"id": <passage-id>, "answer": <answer>, "explanation": <explanation>}}, ...] using JSON best practices (like escaping double quotes in strings with a backslash) surrounded inside <output> </output> tags.

Figure 8: Prompt used in fine-grained LLM-based judgment to evaluate all blocks against a sample in batch

LLM-as-Judge Prompt: Summarize Errors

I need you to summarize possible inaccuracies in a passage. I will give you the passage and a list of possible major/minor conflicts that might be due to inaccuracies in the passage. You are required to identify consistent/recurring conflicts, and summarize them compactly as a helpful comment highlighting the possible inaccuracies in the passage.

Here is the passage (inside <passage> </passage> tag):
<passage>
{block.text}
</passage>

Here are the possible conflicts (inside <conflict> </conflict> tags):
<conflict>
SEVERITY: {block.label[i]}
{block.explanation[i]}
</conflict>
...

Provide your answer as a brief summary (in less than 30 words and inside <summary> </summary> tag).

Figure 9: Prompt used in fine-grained LLM-based judgment to summarize errors found (if any) for a block

Hallucination Mitigation: Block Correction

I need you to correct possible inaccuracies in a passage. I will give you the passage, a summary of the possible inaccuracies, and a detailed list of possible major/minor conflicts that might be due to inaccuracies in the passage. You are required to identify consistent/recurring conflicts, and generate a corrected version that removes all possible inaccuracies and conflicts in the passage.

Here is the passage (inside <passage> </passage> tag):
<passage>
{block.text}
</passage>

Here is the summary of possible inaccuracies (inside <summary> </summary> tag):
<summary>
{block.error_summary}
</summary>

Here are the possible conflicts (inside <conflict> </conflict> tags):
<conflict>
SEVERITY: {block.label[i]}
{block.explanation[i]}
</conflict>
...

Provide the corrected passage inside <corrected> </corrected> tag.

Figure 10: Prompt used for block correction in multi-stage hallucination mitigation

Hallucination Mitigation: Cross-Model Reflection

I have a response from a language model for answering a prompt. I need you to correct possible inaccuracies or incompleteness in the response. I will give you the response (along with the prompt), and a detailed list of different contexts that might be useful to find inaccuracies or incompleteness in the response. You are required to identify consistent/recurring differences, and generate an improved version that better captures the details required in the response.

Prompt (inside <prompt> </prompt> tags)
<prompt>
{prompt.text}
</prompt>

Response (inside <response> </response> tags)
<response>
{response}
</response>
"""

Here are the different contexts (inside <context> </context> tags):
<context>
Context {i}:
{sample[i].text}
</context>
...

Provide a brief explanation of the changes in less than 100 words in <explain> </explain> tags. Then respond with your answer in <improved> </improved> tags.

Figure 11: Prompt used for response-level cross-model reflection in multi-stage hallucination mitigation