Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Conference on Emerging Data and Industry (EDI40), April 14-16, 2026, Istanbul, Türkiye

A Hallucination-Resistant RAG System for Software Update Queries

Farheen Shaikh*, Solomon Berhe

University of the Pacific, Stockton, 95204 California, USA

Abstract

Software update management is a high-risk domain in which incorrect information about patches, versions, and vulnerabilities can lead to misconfigurations, failed deployments, and security exposure. Recent approaches apply large language models (LLMs) with retrieval-augmented generation (RAG) to summarize software updates; however, these systems frequently hallucinate when evidence is incomplete, stale, or ambiguous. In such settings, hallucination is not a minor accuracy issue but an operational hazard. This paper presents a system prototype that adopts an abstention-first, evidence-gated design for software update and vulnerability queries. Instead of generating answers by default, the system evaluates query intent, vendor validity, evidence relevance, and temporal freshness before deciding whether a response should be produced. It retrieves live data from authoritative sources, including operating system release feeds, vendor-maintained repositories, and community platforms, and generates structured responses only when sufficient supporting evidence exists. When evidence is weak, missing, or inconsistent, the system explicitly abstains by returning an "I don't know" response. Evaluation across 20 queries covering version, patch, and CVE scenarios shows that the system prototype answered 16 queries and abstained on 4, with zero hallucinated outputs across all query types. These results demonstrate that treating answerability as a pre-generation decision significantly reduces hallucination while maintaining reliable coverage when evidence is available. The proposed approach reframes hallucination mitigation as a decision problem rather than a generation problem, offering a practical and reliable direction for software update management.

Keywords: Abstention; Hallucination; Evidence; Updates; Reliability

1. Introduction

Hallucination in large language models (LLMs) refers to the production of fluent and well-structured outputs that are factually incorrect, unsupported, or fabricated [1, 2]. Empirical studies show that hallucinations arise when models attempt to infer missing information in the absence of sufficient evidence, often presenting invented facts or misleading

* Corresponding author. Tel.: +1-209-641-4253.
E-mail address: f_shaikh1@u.pacific.edu

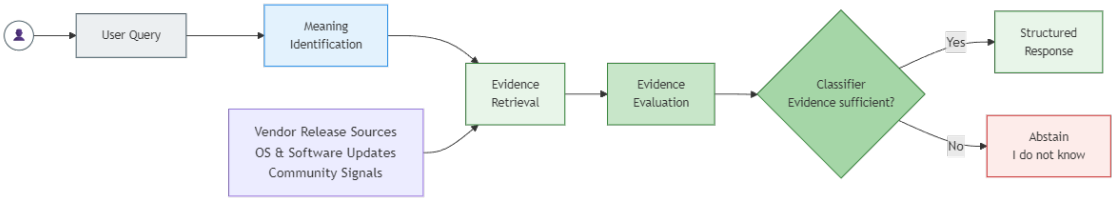

Fig. 1. System Architecture for Evidence-Gated Software Update Question Answering

claims with high confidence [3, 4]. From a systems perspective, hallucination is not merely a linguistic failure but a consequence of optimization objectives that reward answer completeness over epistemic uncertainty. When models are evaluated using accuracy-based metrics that do not reward abstention, they may prefer guessing over expressing epistemic uncertainty [5].

This incentive structure can be illustrated by a simple analogy. In a multiple-choice examination where leaving a question blank guarantees zero points while guessing offers a non-zero chance of success, guessing becomes a rational strategy. Similarly, when an LLM is expected to answer every query, it may generate a plausible but incorrect response instead of stating "I don't know" [6, 7]. Although individual guesses are unlikely to be correct, repeated guessing can inflate aggregate accuracy scores, making a guessing model appear more effective than one that abstains. For queries with a single verifiable answer, incorrect responses are often more harmful than abstentions, particularly in domains with operational or safety consequences [8].

Hallucination poses a heightened risk in software update systems, where accuracy directly affects system stability, security posture, and deployment reliability. Software ecosystems evolve rapidly: operating systems, libraries, and applications release updates on irregular schedules, version identifiers change frequently, and release information is dispersed across vendor feeds, changelogs, community forums, and security advisories. Prior work shows that LLMs queried without sufficient or recent evidence may fabricate answers, including incorrect version numbers, changelog entries, or patch information . Such errors can propagate into downstream workflows (e.g, automated CI/CD pipelines) and administrative decisions, leading to failed deployments, dependency conflicts, or prolonged exposure to known vulnerabilities [3, 9].

Prior studies further demonstrate that retrieval-augmented generation (RAG) systems continue to hallucinate when retrieved evidence is weak, incomplete, contradictory, or stale [3, 4, 10]. Even approaches designed to improve grounding, such as REALM, ATLAS, and RARR, assume that retrieved documents provide sufficient support for generation, an assumption that frequently fails in dynamic domains [11, 12, 13]. While abstention and honesty-aligned training have been explored, most existing systems lack domain-specific mechanisms for evaluating evidence freshness, cross-source consistency, or vendor validity. As a result, response generation remains the default behavior even when reliable evidence is unavailable [6, 7, 14].

This work addresses a foundational question: how can a software update assistant determine not only how to answer a query, but also whether it should answer at all? We argue that software update support should be reframed as an answerability assessment problem, in which abstention is a principled, pre-generation decision rather than a reactive safeguard. This perspective motivates a system prototype that evaluates query intent, vendor validity, and the relevance and freshness of retrieved evidence before allowing any response to be produced, enabling more reliable and verifiable reasoning about software updates.

2. Related Work

Retrieval-augmented generation (RAG) systems have been widely explored as a means of grounding large language model (LLM) outputs in external evidence. Early approaches such as REALM, ATLAS, and RETRO assume that retrieved documents provide sufficient contextual support for generation. However, multiple studies demonstrate that this assumption often fails in practice. When retrieval results are incomplete, noisy, or temporally outdated, RAG systems continue to hallucinate despite access to external text. Subsequent work has attempted to address these limitations: RARR reframes RAG as a rewriting process to reduce unsupported content, while FreshPrompt shows that

Table 1. Comparison of the System Prototype with Prior Work

Dimension	Prior Work	System Prototype
Representative Work	REALM, ATLAS, RARR, FreshPrompt, AbstainQA, IDK, Constitutional AI, SCAlign [12, 13, 11, 10, 6, 7, 15]	This work
Primary Goal	Improve grounding or enforce refusal policies [12, 6, 15]	Answer only if sufficient and recent evidence
Core Assumption	Retrieved context or model confidence is adequate [11, 10]	Evidence must be reliable and up to date
Outdated Evidence	Often ignored or weakly handled [10]	Explicit freshness and consistency checks
Response Strategy	Generate by default, refuse post hoc [6]	Pre-generation abstention decision
Domain Awareness	Model- or prompt-centric signals only [15]	Vendor identity, timestamps, source agreement
Failure Mode	Hallucination under weak or stale retrieval [3]	Unsupported answers blocked before generation
Data Fusion	Limited live or historical integration [13]	Hybrid retrieval over curated and live sources
Safety Stage	During or after generation [16]	Pre-generation answerability assessment

LLMs systematically over-trust stale documents, underscoring the importance of recency-aware retrieval in dynamic domains such as software updates.

A complementary line of research focuses on post-generation verification. Techniques such as LLM Verification and Program-of-Thought Verification assess model outputs after they are generated, filtering responses that exhibit inconsistencies or unsupported reasoning. While these methods can improve correctness, they remain inherently reactive, as a candidate answer must be produced before its reliability can be evaluated. In contrast, the proposed system prototype treats answerability as a pre-generation decision, aiming to prevent unsupported responses from being generated in the first place [16].

Safety-aligned models offer another perspective on refusal and uncertainty. Approaches such as Constitutional AI and SCAlign enforce restricted behaviors through refusal policies driven by ethical or safety constraints. However, these techniques do not incorporate domain-specific signals such as evidence freshness, cross-source consistency, or update-timeline volatility. As a result, they remain vulnerable to hallucinations in domains where factual accuracy depends on rapidly evolving information rather than normative constraints. The proposed system prototype complements this line of work by applying refusal based on evidence insufficiency rather than safety alignment alone [15, 8].

Related ideas also appear in automated program repair (APR), where abstention has been introduced as a mechanism for predicting whether a repair attempt is likely to succeed. By abstaining when confidence is low, APR systems reduce wasted developer effort and improve trust. Although developed for a different application domain, this work illustrates the broader value of abstention-based decision pipelines for improving the reliability of AI-assisted systems. The proposed system prototype adopts a similar philosophy for software-update queries, determining whether a query is answerable before initiating generation [14].

A separate research direction suggests that hallucination can, in some controlled settings, improve reasoning performance. For example, null-shot prompting shows that deliberately inserting hallucinated information can increase arithmetic reasoning accuracy. While these findings reveal complex dynamics in LLM behavior, such strategies are incompatible with safety-critical applications. In software-update contexts, fabricated version numbers or nonexistent patches can lead to deployment failures, misconfigurations, or extended vulnerability exposure, making hallucination a liability rather than a benefit [1, 2].

Across work on RAG grounding, post-generation verification, safety alignment, and abstention, a common limitation remains: generation is typically the default action, with refusal occurring only after a potential failure is detected. Existing systems rarely incorporate domain-aware signals such as evidence recency, multi-source agreement,

or identifiable vendor validation. The proposed system prototype addresses this gap by introducing a pre-generation abstention mechanism that evaluates evidence strength, reliability, and freshness before allowing an LLM to respond. This abstention-first design reframes hallucination mitigation as an answerability assessment problem, making it particularly suited to fast-evolving and operationally sensitive software-update environments [3, 4, 9, 11, 10].

3. Method

This section describes the design of an AI-assisted software-update question answering system that emphasizes evidence sufficiency over response completeness. The system is implemented as an evidence-gated pipeline in which natural language understanding, retrieval, evaluation, and decision-making are explicitly separated and coordinated. Rather than assuming that every query should be answered, the system introduces a pre-generation determination of answerability based on the availability, reliability, and freshness of supporting evidence.

The system is organized around a single guiding principle: responses are generated only when sufficient, reliable, and up-to-date evidence exists. In contrast to conventional question-answering systems that default to producing an answer, this design introduces an explicit decision point that determines whether answering a query is appropriate at all. Figure 1 illustrates the overall system architecture. A user query is processed through a sequence of stages consisting of meaning identification, evidence retrieval, evidence evaluation, and a final answerability decision. Depending on the outcome of this decision, the system either constructs a structured, evidence-backed response or explicitly abstains by returning an "I don't know" output. The pipeline is designed to be modular and interpretable, favoring reliability and safety over speculative completeness.

Processing begins when a user submits a natural-language query related to software updates, patches, versions, or vulnerabilities. Rather than treating the input as a keyword-based search request, the system first performs meaning identification, aiming to infer the conceptual intent of the query. At this stage, the system determines the intended query type, such as version status, patch availability, or vulnerability information, as well as the contextual scope of the query, including the referenced software artifact, operating system, or vendor. This interpretation step constrains downstream processing and ensures that subsequent retrieval focuses on relevant evidence. Queries that cannot be confidently mapped to a supported intent or scope are conservatively handled, reducing the likelihood of unsupported responses.

Once query meaning has been identified, the system retrieves candidate evidence from a curated set of authoritative and heterogeneous sources. These sources reflect the types of artifacts typically consulted in operational software-update decision making and include official vendor release feeds and release notes, operating system and software update repositories, security advisories and vulnerability disclosures, and structured community or ecosystem announcements. Evidence is retrieved from a controlled datalake rather than the open web, enabling consistent access to structured metadata and provenance information. Retrieved artifacts are ranked based on their relevance to the inferred intent and contextual scope, while preserving temporal and source-related attributes required for validation.

Retrieved evidence is not used directly. Instead, each candidate artifact is evaluated along multiple dimensions to assess its suitability for answering the query. This evaluation examines relevance, indicating whether the evidence directly addresses the user's question; consistency, indicating whether multiple sources support compatible conclusions; and freshness, indicating whether the information reflects the current state of the software or system. Evidence that is outdated, weakly related, or inconsistent across sources is filtered out, ensuring that only high-quality and context-appropriate information can influence subsequent decision-making.

At the core of the system design is an explicit answerability decision. Based on the evaluated evidence, a classifier determines whether the available information is sufficient to support a reliable response. This determination is made prior to any response generation. If the evidence satisfies the system's sufficiency criteria, processing proceeds to response construction. If the evidence is missing, weak, inconsistent, or stale, the system explicitly abstains and returns an "I don't know" response. This abstention-first mechanism prioritizes correctness and safety, particularly in domains where inaccurate information may lead to operational or security risks.

When evidence is deemed sufficient, the system generates a structured response designed to improve interpretability and traceability. Generated responses separate information derived from different evidence sources, highlight key findings such as version changes, available patches, or vulnerability status, and remain grounded in retrieved and

Table 2. Case Studies Demonstrating Answered and Abstained Queries

Case Study	Scenario	System Behavior	Outcome	Significance
Recent Version Query (NVIDIA)	User asks about the most recent version of a vendor model	The system identifies a version intent, validates the vendor, and retrieves curated release artifacts filtered by relevance and freshness	Evidence-backed version information is returned	Demonstrates reliable response generation when authoritative evidence exists
CVE Query (Windows)	User asks whether a fix exists for a specific CVE	The system evaluates security advisories and vendor bulletins for consistency and recency	The system answers if confirmed, otherwise abstains	Illustrates abstention as a principled safety mechanism for vulnerability queries

validated artifacts. This structured format supports informed decision making by making both conclusions and their evidentiary basis explicit, while remaining compatible with both automated workflows and human oversight.

4. Results

This section reports the observed behavior of the evidence-gated question answering system when applied to software update queries. The evaluation focuses on two aspects: (i) how the system responds when sufficient, reliable evidence is available, and (ii) how it behaves when evidence is missing, weak, or stale, emphasizing principled abstention over speculative generation.

4.1. Experimental Setup and Configuration

The evaluation dataset consists of software update artifacts collected from operating system release feeds, vendor-maintained repositories, security advisories, and structured community announcements. Queries were constructed to reflect real-world information needs, including version identification, patch availability, and vulnerability status. The system operates with a small set of fixed and interpretable parameters governing intent confidence, vendor validity, evidence sufficiency, and temporal freshness. Supported query intents include version, patch, and vulnerability queries, while vendor validation relies on a curated index derived from authoritative sources. Evidence is retrieved from a controlled datalake, ranked by relevance and recency, and filtered using a predefined freshness window to reduce the influence of outdated information. Conservative thresholds are applied consistently across all experiments. Queries are answered only when intent recognition, vendor validation, and evidence sufficiency exceed minimum confidence requirements. When these conditions are not met, the system explicitly abstains to avoid unsupported or potentially misleading outputs, ensuring reproducible and safety-oriented behavior.

4.2. Qualitative Evaluation: Answered and Abstained Queries

The system exhibits two primary behaviors depending on evidence availability. When a query references a recognized vendor and sufficient recent evidence exists, the system produces a structured, evidence-backed response grounded in authoritative artifacts. For example, in a version-related query concerning a well-known vendor, relevant release artifacts are retrieved, validated for relevance and freshness, and summarized into a clear response.

In contrast, when evidence is missing, inconsistent, or unverifiable such as in certain vulnerability-related queries the system abstains and returns an explicit “I don’t know” response. This behavior is particularly important for safety-critical queries, where premature or incorrect claims may lead to operational or security risks.

4.3. Abstention Logic and Decision Pathways

To better understand abstained outcomes, we analyze the internal decision pathways that lead the system to decline answering. These pathways are intentionally simple, interpretable, and conservative, ensuring that abstention is a deliberate and explainable result rather than a failure case.

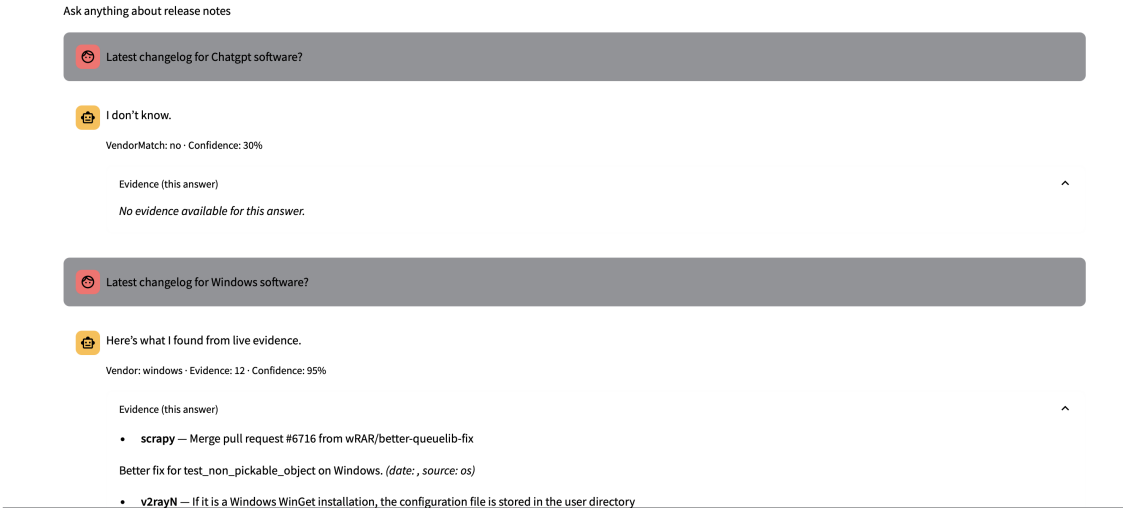

Fig. 2. Example evidence-backed response for a version query

4.3.1. *Vendor Detection and Early Abstention*

```
vendor ← detect_vendor_from_query(query, allowed_vendors)

IF vendor is not found THEN
    fallback_vendor ← last token of query OR "your vendor"
    RETURN abstain with confidence = 0.30
END IF
```

The first abstention pathway occurs when the system cannot resolve a valid vendor from the query. In this case, the system abstains immediately, assigning a low confidence score to reflect the absence of grounding. This prevents speculative reasoning when the query does not map to a known vendor in the curated index.

4.3.2. *Evidence Sufficiency Check*

```
vendor_items ← FILTER items WHERE vendor appears in evidence

IF vendor_items is empty THEN
    RETURN abstain with confidence = 0.80
END IF
```

Even when a vendor is successfully identified, the system requires corroborating evidence from trusted sources. If no relevant and recent evidence artifacts are found, the system abstains despite higher internal confidence. This distinction reflects the system's design principle that vendor recognition alone is insufficient for answering.

4.3.3. *Intent-Aware Abstention Responses*

```
IF vendor exists AND intent exists THEN
    RETURN "I don't know about <vendor> for <intent>"
ELSE
    RETURN "I don't know"
END IF
```

When abstaining, the system generates responses that remain context-aware. If both vendor and intent are known, the abstention message explicitly references them, preserving conversational relevance while avoiding unsupported claims.

Table 3. Quantitative Evaluation of Answerability and Abstention

Query Type	Total Queries	Answered	Abstained	Hallucinated
Version	10	6	1	0
Patch	5	6	1	0
CVE	5	4	2	0
Overall	**20**	**16**	**4**	**0**

4.3.4. Normalization of Abstention Outputs

```
FUNCTION normalize_idk(payload, query):
    IF not abstained THEN
        RETURN answer
    END IF
    vendor ← resolved vendor OR last token of query
    intent ← resolved intent
    RETURN normalized "I don't know" response
END FUNCTION
```

A normalization step ensures that all abstention responses follow a consistent linguistic structure regardless of which internal rule triggered the abstention. This guarantees predictable, user-facing behavior and prevents leakage of internal decision states. Table 3 summarizes system behavior across all evaluated queries. The results report the number of answered and abstained queries by type, as well as the absence of hallucinated outputs. Across all 20 queries, no hallucinated responses were observed, indicating that unsupported answers were successfully blocked prior to generation.

To support reproducibility, the system prototype used in this evaluation is publicly available.[1]

5. Discussion

This section discusses the implications of the experimental results and case studies, focusing on the broader lessons learned from adopting an answerability-first design for software update systems and its impact on reliability and safety.

Answerability over generation. The findings show that modeling software update queries as an answerability decision, rather than a purely generative task, significantly reduces hallucinated outputs. Introducing a pre-generation decision point ensures that responses are produced only when evidence is sufficient, reliable, and current, which is essential in safety-sensitive contexts such as vulnerability and patch management.

Abstention as a correct outcome. A central lesson is that abstention should be treated as a successful system behavior rather than a failure. Instead of maximizing response coverage, the system prioritizes evidence accuracy. Quantitative results indicate that most queries are answered when authoritative evidence exists, while abstention is triggered when reliable conclusions cannot be drawn, reflecting a deliberate trade-off between informativeness and safety.

Transparency and trust. The qualitative case studies illustrate that structured, evidence-backed responses are generated when conditions are met, and uncertainty is communicated explicitly otherwise. This transparency helps users distinguish verified information from unresolved cases, reducing the risk of misinformed operational decisions and improving user trust.

Limitations and outlook. The current evaluation is limited in scale and scope and relies largely on qualitative comparisons. Future work will expand datasets, incorporate automated baselines, and study user interactions with abstaining systems. Currently, the response time is, on average, 3 seconds. For improvement, adaptive thresholding

[1] For reproducibility and transparency, an implementation of the system prototype used in this study is available at https://github.com/farheen-shaikh530/RAG-Pipeline-for-ReleaseHub.

and domain-specific calibration are promising directions. In summary, the Pre-generation abstention and evidence gating provide a practical and effective foundation for mitigating hallucinations in dynamic, high-risk software update environments.

6. Conclusion

This work presented an abstention-first, evidence-gated system for software update and vulnerability queries. In contrast to conventional retrieval-augmented approaches that generate responses by default, the system evaluates query intent, vendor validity, evidence relevance, and temporal freshness before deciding whether an answer should be produced. When these conditions are not met, the system explicitly abstains, avoiding unsupported or speculative outputs. Across qualitative case studies and quantitative evaluation, the system consistently generated structured, evidence-backed responses when reliable information was available and abstained in the presence of uncertainty, incomplete data, or unverifiable sources. These findings demonstrate that reframing software update assistance as an answerability assessment problem leads to safer and more predictable system behavior, particularly in operationally sensitive contexts. Future work will focus on expanding the evaluation to larger and more diverse datasets, introducing systematic baseline comparisons, and exploring adaptive thresholding and confidence calibration strategies. Overall, this study suggests that abstention-first decision pipelines provide a practical and effective foundation for building reliable AI systems in dynamic, high-risk information domains.

Acknowledgements

We thank the open data and research communities for continuously publishing software-related content across public platforms and for providing APIs that enable programmatic access and reproducible analysis. Their openness and data-sharing efforts make it possible to study large-scale discussions of software updates across diverse ecosystems.

References

[1] Ji, Z., et al., “Survey of Hallucination in Natural Language Generation,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
[2] Varshney, A., et al. “Iin nvestigating and Addressing Hallucinations of LLMs in Tasks Involving Negation,” in *arXiv preprint arXiv:2406.05494*, 2024.
[3] Li, S., et al., “Detecting Hallucinations in Graph Retrieval-Augmented Generation via Attention Patterns and Semantic Alignment,” in *arXiv preprint arXiv:2512.09148v1*, 2025.
[4] Alansari, A., et al., “Large Language Models Hallucination: A Comprehensive Survey,” in *arXiv preprint arXiv:2510.06265v2*, 2025.
[5] Kadavath, S., et al., “Language Models (Mostly) Know What They Don’t Know,” in *arXiv preprint arXiv:2207.05221*, 2022.
[6] Anh, H., et al., “Teaching LLMs to Abstain via Fine-Grained Semantic Confidence Reward” in *arXiv preprint arXiv:2510.24020*, 2025.
[7] Zhang, H., et al., “R-Tuning: Instructing Large Language Models to Say ‘I Don’t Know’,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
[8] Ganguli, D., et al., “Red Teaming Language Models to Understand Safety Risks,” in *arXiv preprint arXiv:2301.08650*, 2023.
[9] Niu, C., et al., “RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models,” in *arXiv preprint arXiv:2401.00396*, 2023.
[10] Pesaranghader, A., et al., “Hallucination Detection and Mitigation in Large Language Models,” in *arXiv preprint*, 2026.
[11] Krishna, K., et al., “RARR: Retrieval-Augmented Rewriting for Reliable Large Language Model Generation,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.
[12] Guu, K., et al., “REALM: Retrieval-Augmented Language Model Pre-Training,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.
[13] Izacard, G., et al., “ATLAS: Few-Shot Learning with Retrieval,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
[14] Madhusudhan, A., et al., “Do LLMs Know When to Not Answer? Investigating Abstention Abilities of Large Language Models,” in *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 7224–7244, 2025.
[15] Bai, Y., et al., “Constitutional AI: Harmlessness from AI Feedback,” in *arXiv preprint*, 2022.
[16] Zhou, Y. et al., “Variation in Verification: Understanding Verification Dynamics in Large Language Models,” in *arXiv preprint*, 2025.