Vol. 16 (2026): Proceedings of the 2026 International Conference on AI and Technological Innovation Applications (AITIA 2026)
E-ISSN: 3105-0522 P-ISSN: 3105-0514
DOI: https://doi.org/10.70267/aitia.2026370377

Mitigating Hallucinations in Natural Language Generation through Prompt Engineering: A Mechanism-Oriented Narrative Review

Jiani Lin*

Artificial Intelligence, Shanxi University, Shanxi, China

**Corresponding author: Jiani Lin.*

Abstract

Large language models (LLMs) can generate fluent and confident responses that are factually incorrect, unsupported by evidence, or inconsistent with the source material. These hallucinations reduce the reliability of LLM-based question answering, summarization, dialogue, and information retrieval systems, especially when users rely on generated content for consequential decisions. This narrative review examines prompt engineering as a practical inference-time approach for reducing hallucinations without modifying model parameters. Instead of listing individual prompting templates, the paper organizes existing methods according to five functional mechanisms: constraint and evidence grounding, decomposition and verification, multi-path consistency, iterative refinement and tool use, and retrieval-augmented prompting. For each mechanism, the review discusses the hallucination types it is most likely to address, the assumptions required for success, and the conditions under which it may fail. Particular attention is given to Chain-of-Verification (CoVE), which improves auditability by separating answer generation from targeted checking, but remains vulnerable when verification is performed by the same model without independent evidence. The review argues that prompt engineering should be understood as process-level risk reduction rather than a complete solution. Its strongest use is within evidence-centered system designs that combine retrieval, claim-level verification, calibrated abstention, provenance display, and human review in high-risk contexts.

Keywords

large language models, hallucination, prompt engineering, chain-of-verification, retrieval-augmented generation, information retrieval

1. Introduction

Large language models have made natural language generation more fluent, flexible, and accessible. However, fluency is not a reliable indicator of truth. An LLM may produce an answer that sounds coherent and authoritative while introducing claims that are false, unsupported, outdated, or inconsistent with the given source material. This problem is commonly referred to as hallucination [1–3]. The issue is not merely stylistic. In legal research, education, scientific assistance, medical communication, finance, and information retrieval, an unsupported answer may influence user decisions before it is checked.

Prompt engineering is widely used to reduce this risk because it can be applied at inference time without retraining the model or accessing its internal parameters. A prompt can instruct the model to stay within a supplied source, decompose a complex question, generate verification questions, compare alternative reasoning paths, use external tools, or retrieve supporting documents. These methods are relatively inexpensive and can be added to existing LLM applications [4,5]. Nevertheless, the term "prompt engineering" covers many different practices. A prompt that says "answer only from the passage" works through a different mechanism from self-consistency, self-critique, tool use, or retrieval-augmented generation (RAG). Therefore, it is not sufficient to state that better prompts reduce hallucination. The key question is which mechanism reduces which type of error, under what conditions, and with what limitations.

This review adopts a mechanism-oriented perspective. It first clarifies the main hallucination types that prompt-based interventions attempt to address: factual hallucination, faithfulness hallucination, intrinsic and extrinsic hallucination, and high-certainty hallucination. It then groups prompt-engineering strategies into five functional families: constraint and evidence grounding, decomposition and verification, multi-path consistency, iterative refinement and tool use, and retrieval-augmented prompting. The review pays special attention to CoVE because it illustrates both the value and the limits of model-generated verification [6]. Overall, the paper argues that prompt engineering is useful, but its effectiveness depends on evidence access, verification independence, and deployment context. It should be treated as one component of a broader reliability architecture rather than as a standalone cure for hallucination.

2. Hallucination as the Target of Prompt-Based Mitigation

A useful taxonomy of hallucination should support intervention design. At the broadest level, factual hallucination occurs when generated content conflicts with verifiable world knowledge. Faithfulness hallucination occurs when a response departs from the user input, the source document, the retrieved evidence, or the dialogue state [1,2]. This distinction is important because factuality often requires external evidence, while faithfulness can often be checked against material already provided to the model. In document-grounded tasks, faithfulness errors are commonly divided into intrinsic hallucinations and extrinsic hallucinations. Intrinsic hallucinations directly contradict the source, whereas extrinsic hallucinations add information that is not supported by the source [1].

High-certainty hallucination is a more difficult reliability problem. Simhi et al. describe cases in which a model can produce the correct answer under some prompting conditions but gives an incorrect answer with strong apparent confidence under others [7]. In these cases, the problem is not simply a lack of knowledge. It may involve unstable knowledge access, poor calibration, or a prompt-conditioned preference for a plausible but wrong answer. High-certainty hallucinations are particularly dangerous because repeated generations may appear consistent, and the model may not express uncertainty.

Different hallucination types require different prompt objectives. Factual hallucinations require evidence acquisition and claim verification. Faithfulness hallucinations require strict source tracking and limits on unsupported inference. Intrinsic errors benefit from sentence- or claim-level comparison with the source. Extrinsic errors require explicit evidence boundaries and abstention rules. High-certainty hallucinations require independent evidence or model-external checks, because model confidence alone is not trustworthy. Table 1 summarizes these relationships.

The causes of hallucination are distributed across training data, learning objectives, alignment procedures, and inference conditions [2,3]. Training corpora may contain outdated, contradictory, or sparse information. Next-token prediction rewards plausible continuation rather than factual adjudication. Instruction tuning and reinforcement learning from human feedback may encourage the model to be helpful even when evidence is insufficient. At inference time, ambiguity, long contexts, stochastic decoding, and multi-fact questions create further opportunities for error. Prompt engineering mainly acts at this final stage. It can guide how the model uses evidence and structures reasoning, but it cannot fully correct weaknesses in the model's internal knowledge. This boundary is important: prompt engineering is a form of process control, not a guarantee of truth.

Table 1: Hallucination types and corresponding prompt objectives

Type	Operational definition	Prompt/system implication
Factual	Conflicts with verifiable world knowledge	Retrieve authoritative evidence and verify atomic claims
Faithfulness	Conflicts with the input, source, or dialogue context	Constrain answers to supplied material and require claim–evidence alignment
Intrinsic	Directly contradicts a source statement	Use sentence- or claim-level comparison with the source
Extrinsic	Adds information not supported by the source	Set evidence boundaries and require abstention when support is absent
High-certainty	Produces a stable, confident error despite latent access to the correct answer	Use independent evidence, conflict detection, and escalation

3. Prompt-Engineering Strategies for Hallucination Mitigation

3.1 Constraint and Evidence-Grounded Prompting

Constraint-based prompts define the evidence boundary and specify what the model should do when support is absent. Typical instructions include "answer only from the provided documents," "cite the passage supporting each claim," and "state that the answer is unknown if the evidence is insufficient." The value of these prompts is not that they give the model more factual knowledge. Rather, they narrow the acceptable answer space and make unsupported claims easier to identify. This strategy is therefore most relevant to faithfulness errors and extrinsic hallucinations in summarization, document question answering, and retrieval interfaces.

Effective constraints should be operational rather than vague. An instruction such as "be accurate" provides no testable rule. By contrast, "do not assert a claim unless it is supported by one of the supplied passages" gives the model and the evaluator a clearer standard. Structured outputs can further improve transparency by separating the final answer, supporting evidence, and uncertainty labels. However, constraint prompting is not sufficient on its own. A model may ignore the instruction, over-interpret weakly related evidence, or fabricate citations. Constraint prompts should therefore be treated as a baseline control that improves auditability, while higher-risk applications still require downstream validation.

3.2 Decomposition and Verification

Decomposition reduces the number of facts that must be generated and checked at the same time. A complex question can be divided into smaller subquestions, and a long answer can be converted into a list of verifiable claims. CoVE follows this logic by separating an initial response from targeted verification questions and a revised answer [6]. Its main value lies in changing the unit of inspection. Instead of judging a fluent paragraph as a whole, the system can check specific dates, entities, quantities, and relations.

This strategy is well suited to long factual answers, multi-hop questions, and source-based synthesis. It can reveal local errors that remain hidden in fluent prose and can create an audit trail for reviewers. However, verification only works when it is both sufficiently comprehensive and sufficiently independent. The verifier must identify the claims most likely to be wrong, ask questions that can falsify them, and avoid reproducing the same unsupported answer. If a critical claim is not checked, or if the same model repeats the same mistaken association during verification, the error may remain unchanged. Multi-step verification also increases latency and cost, so selective verification is often more practical than checking every sentence.

3.3 Multi-Path Consistency

Chain-of-thought prompting asks the model to generate intermediate reasoning steps [8]. Self-consistency samples several reasoning paths and selects the most frequent answer [9]. Contrastive chain-of-thought provides examples of both correct and incorrect reasoning to sharpen discrimination [10]. These methods are useful when the error comes from stochastic decoding, fragile reasoning, or an unlucky single path, as in mathematical reasoning, commonsense inference, and multi-step question answering.

However, consistency should not be confused with correctness. If the model shares the same misconception across several samples, majority voting will reinforce the error. Similarly, a fluent chain of thought may rationalize a false premise instead of exposing it. SelfCheckGPT uses disagreement across samples as a black-box signal for hallucination detection [11], but low disagreement does not prove factuality. Multi-path methods are therefore best understood as uncertainty and robustness heuristics. For verifiable factual tasks, they should be combined with external evidence rather than used as substitutes for evidence.

3.4 Iterative Refinement and Tool Use

Iterative methods separate drafting, critique, and revision. Self-Refine formalizes a generate–feedback–improve loop in which the model critiques its own output and then revises it [12]. ReAct interleaves reasoning with actions such as search or tool calls, allowing external observations to influence the final answer [13]. These approaches are valuable for writing, explanation, coding, and interactive problem solving because they create opportunities to correct omissions, logical gaps, or execution errors after the first draft.

The key question is whether the feedback introduces new evidence. Self-critique by the same model may improve clarity while preserving the original falsehood. Repeated revision can even make an incorrect answer more polished and persuasive. Tool-mediated refinement is stronger when the tool provides information that is independent of the initial response, such as search results, database records, code execution, or calculation. Even then, tools create new failure points. The system must check whether the tool was appropriate, whether the query was well formed, and whether the observation actually supports the conclusion.

3.5 Retrieval-Augmented Prompting

RAG reduces reliance on parameterized memory by retrieving external documents and conditioning generation on them [14]. It is especially important for time-sensitive, specialized, or organization-specific questions. When retrieval is accurate, the model can ground its claims in current evidence and provide provenance. This directly addresses many factual hallucinations and can also reduce extrinsic additions in document-based answers.

RAG, however, relocates uncertainty rather than eliminating it. Retrieved documents may be irrelevant, outdated, contradictory, or adversarial. The model may also ignore strong evidence, overuse weak evidence, or combine passages incorrectly. Chain-of-Note asks the model to create reading notes for retrieved documents and distinguish useful evidence from noise, improving robustness when retrieval results are imperfect [15]. CoV-RAG further combines retrieval with verification and revision [16]. These extensions show that retrieval should be paired with evidence selection, claim–source alignment, and abstention when the retrieved material does not support a reliable answer.

3.6 Evaluating Prompt-Based Mitigation

Evaluation should distinguish a genuinely safer system from one that merely sounds more cautious. Aggregate answer accuracy is not enough, because a method may reduce unsupported claims while increasing refusals, latency, or omission of useful information. At minimum, evaluation should report factual correctness, faithfulness to supplied evidence, citation precision at the claim level, evidence coverage, calibration or selective accuracy, and appropriate abstention. For multi-step methods, computational cost and response time should be treated as part of the reliability trade-off rather than secondary engineering details.

The evaluation protocol should match the mechanism being tested. Constraint prompts require source-grounded tasks in which unsupported additions can be identified. Verification methods require claim-level annotations, including whether the generated verification questions cover important possible errors. RAG systems should be evaluated separately for retrieval quality and generation quality, so that retrieval failures are not mistakenly attributed to the prompt. Robustness should also be tested across prompt paraphrases, model versions, domains, and misleading contexts. A method that works only under one carefully tuned template provides limited assurance in deployment.

Table 2: Functional comparison of prompt-engineering strategies

Strategy family	Primary mechanism	Best-suited target	Main limitation
Constraint and evidence grounding	Narrows the admissible answer space	Faithfulness and extrinsic errors	Instructions may be ignored; fabricated evidence remains possible
Decomposition and verification	Checks atomic claims separately	Long factual answers and multi-hop synthesis	Coverage gaps and non-independent verification
Multi-path consistency	Aggregates several reasoning paths	Random or path-dependent reasoning errors	Consistent misconceptions may be amplified
Iterative refinement and tool use	Uses critique, revision, or external actions	Writing, coding, and interactive tasks	Self-feedback may polish rather than correct errors
Retrieval-augmented prompting	Conditions generation on external documents	Knowledge-intensive and time-sensitive tasks	Retrieval noise and evidence misuse

4. CoVE, High-Certainty Hallucinations, and the Limits of Self-Verification

CoVE is useful because it turns verification from a vague instruction into a structured procedure [6]. Its main strength is procedural separation. A baseline response encourages the model to produce a complete answer, while targeted verification redirects attention to individual claims. Verification questions can be answered against retrieved evidence rather than against the original response, which reduces anchoring. For developers and reviewers, the resulting chain of claims, questions, evidence, and revisions is more auditable than a single opaque answer.

CoVE is most likely to succeed under three conditions. First, the answer must contain identifiable claims that can be checked independently. Second, the verification stage should use evidence that is at least partly independent from the mechanism that produced the first answer. Third, the revision stage must be allowed to delete, qualify, or withhold unsupported content instead of simply rephrasing it. Under these conditions, CoVE is useful for biographical facts, dates, numerical claims, multi-entity comparisons, and evidence-based synthesis.

Its failure modes show why self-verification should not be treated as external validation. A model may generate superficial questions that confirm the first answer rather than challenge it. It may fail to ask about the most consequential claim. It may also answer the verification question from the same erroneous internal association that caused the original hallucination. In these cases, the procedure adds tokens but not epistemic independence. Verification quality therefore depends on adversarial question design, access to reliable evidence, and a decision rule that permits abstention.

High-certainty hallucinations make this problem more serious. If a model repeatedly retrieves the same wrong association with strong apparent confidence, generic caution, self-consistency, and model-only CoVE may converge on the same incorrect answer [7]. Agreement across several reasoning paths can then create false reassurance. The better response is not simply to ask for more reasoning. The system should seek evidence independent of the initial completion, compare each claim against that evidence, and downgrade or withhold the answer when conflict remains unresolved.

For this reason, CoVE is best understood as a claim-level verification framework rather than a universal remedy. Self-consistency is stronger against random reasoning variation. Self-Refine is better suited to improving open-ended outputs. ReAct is useful when corrective actions or tools are available. RAG is essential when the answer depends on external or updated knowledge. CoVE becomes most valuable when it coordinates these components: extracting claims from a RAG answer, generating falsifiable verification questions, checking them against independent passages, and revising only after evidence comparison.

5. Design Implications for Reliable Information Retrieval Systems

For information retrieval systems, the design goal should shift from fluent response generation to evidence-centered answer production. A reliable pipeline begins with retrieval but must also assess document relevance, freshness, source authority, and contradiction before generation. Prompts should require claim-level citations and should explicitly forbid unsupported synthesis. When a question cannot be answered from the retrieved

set, the preferred behavior is calibrated abstention or a request for clarification rather than completion from memory.

Selective verification can make this architecture more efficient. Instead of applying CoVE to every sentence, the system can prioritize claims involving names, dates, quantities, causal relations, legal or medical recommendations, and other high-impact content. Verification questions should be designed to falsify the draft, not merely restate it. When possible, they should be answered using different passages, tools, or models from those used in the first generation. A contradiction policy should specify whether the answer is revised, qualified, escalated for human review, or withheld.

Interface design also affects reliability. Users should be able to distinguish retrieved evidence from model inference, open the underlying sources, and see which claims have passed or failed verification. Confidence should not be displayed as a single model-generated percentage. It should reflect observable signals such as evidence coverage, source agreement, retrieval quality, and verification outcomes. In high-risk domains, the final layer should include domain-specific rules and human oversight. Prompt engineering then becomes one part of a broader assurance architecture rather than a substitute for governance.

A risk-tiered routing policy can make the system practical. Low-impact requests may use a single grounded response, while questions involving consequential decisions can trigger deeper retrieval, claim extraction, independent verification, or expert review. The routing criterion should depend on both the domain and the answer characteristics, including numerical recommendations, legal obligations, diagnoses, or claims supported by only one weak source. This approach reserves expensive verification for cases in which an error would matter most while preserving responsiveness for routine use.

6. Discussion and Conclusion

Prompt engineering can reduce hallucination by controlling evidence boundaries, decomposing complex generation, comparing alternative reasoning paths, enabling revision, and incorporating external knowledge. Its advantages are accessibility, flexibility, and the ability to expose intermediate decisions. These features make it useful for rapid deployment and for systems in which developers cannot modify model parameters.

At the same time, its limitations must be stated clearly. Prompt effects are sensitive to wording, model choice, and task context. Model-generated critique may repeat the original misconception. Retrieval introduces errors of its own. Multi-step workflows increase latency and cost. Most importantly, high-certainty hallucinations show that confidence and consistency are not reliable indicators of truth. A model may be consistently wrong, and additional self-reflection may stabilize rather than correct the error.

The most defensible conclusion is therefore conditional. Prompt engineering is effective as process-level risk reduction, especially for faithfulness errors and claim-level factual checking, but it should not be presented as a complete solution. Reliable natural language generation requires a layered design that combines high-quality retrieval, explicit evidence constraints, targeted verification, calibrated abstention, provenance display, and risk-sensitive human review. Future research should focus on automated selection of claims for verification, stronger measures of verifier independence, multilingual evaluation, efficient routing between small and large models, and standardized benchmarks that measure not only answer accuracy but also evidence alignment and appropriate refusal.

References

[1] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*(12), 1–38. https://doi.org/10.1145/3571730

[2] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems, 43*(2), 1–55. https://doi.org/10.1145/3703155

[3] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., & Shi, S. (2025). Siren's song in the AI ocean: A survey on hallucination in large language models. *Computational Linguistics, 51*(4), 1373–1418. https://doi.org/10.1162/coli.a.16

[4] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys, 55*(9), 1–35. https://doi.org/10.1145/3560815

[5] Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv*. https://arxiv.org/abs/2402.07927

[6] Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., & Weston, J. (2023). Chain-of-verification reduces hallucination in large language models. *arXiv*. https://arxiv.org/abs/2309.11495

[7] Simhi, A., Itzhak, I., Barez, F., Stanovsky, G., & Belinkov, Y. (2025). Trust me, I'm wrong: High-certainty hallucinations in LLMs. *arXiv*. https://arxiv.org/abs/2502.12964

[8] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 24824–24837).

[9] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E. H., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

[10] Chia, Y. K., Chen, G., Tuan, L. A., Poria, S., & Bing, L. (2023). Contrastive chain-of-thought prompting. *arXiv*. https://arxiv.org/abs/2311.09277

[11] Manakul, P., Liusie, A., & Gales, M. J. F. (2023). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 9004–9017). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.557

[12] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., & Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. *arXiv*. https://arxiv.org/abs/2303.17651

[13] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

[14] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 9459–9474).

[15] Yu, W., Zhang, H., Pan, X., Ma, K., Wang, H., & Yu, D. (2024). Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 14672–14685). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.emnlp-main.813

[16] He, B., Chen, N., He, X., Yan, L., Wei, Z., Luo, J., & Ling, Z.-H. (2024). Retrieving, rethinking and revising: The chain-of-verification can improve retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024* (pp. 10371–10393). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-emnlp.607

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.