IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15574

AI-BASED VERIFICATION OF LLM RESPONSE

Mrs. Asha Sattigeri[1], Lavanya R[2], Pavan Gowda S[3]

Assistant Professor, Dept of CSE, KSIT, Karnataka, India[1]

Student, Dept of CSE, KSIT, Karnataka, India[2,3]

Abstract: Large Language Models(LLMs) now handle tasks like question answering, summarisation, code generation and dialogue with impressive results. Yet they still suffer from a key issue: hallucination happens when a model generates text that reads well but is factually wrong or not grounded in real evidence. The risk is higher in domains like healthcare, law, finance and research, where inaccurate outputs can lead to real damage. This survey focuses on how to detect hallucination verification in LLMs. We review 5 core detection approaches: retrieval-based, uncertainty-based, embedding-based, learning-based and self-consistency methods. We also cover current mitigation techniques, popular benchmarks such as Truthful QA and HaluEval, common evaluation metrics and verification tools such as xVerify and CompassVerifier. This paper closes by discussing open challenges and future directions for building more reliable, truthful LLMs

Keywords: Large Language Models, Hallucination Detection, Hallucination Mitigation, Factuality, Retrieval-Augmented Generation, TruthfulQA, HaluEval, Answer verification, Claim verification, Internal states

I.INTRODUCTION

Large language models (LLMs) have reshaped the field of AI. They are now capable of writing text, reasoning, generating code, and holding conversations at a level that was hard to imagine a few years ago. Models like GPT-4, Claude, Gemini and LLaMA show just how far language understanding and generations have come, and they have changed how we interact with technology day-to-day.

But there's still one major flaw: hallucination. This is when an LLM produces content that sounds smooth and confident but is actually false or completely made-up, unlike old rule-based systems, where errors were easy to trace. LLM isolation is baked into how the models work. They're probabilistic systems that don't know what they don't know, so they can present fiction with the same certainty as fact.

This isn't just a technical quirk. In healthcare, an isolated drug dose or diagnosis can be life-threatening. In law, fake case citations have already appeared in court filings, leading to penalties and judicial mistakes. In finance, incorrect statements in reports can push people towards bad investments. As LLMs get used in more autonomous roles, browsing the web, running code, and managing complex workflows, one undetected allocation can trigger real damage.

Tackling this problem means more than just flagging wrong answers; we need to understand what kinds of hallucinations exist, why they happen and how to measure them. This survey covers 5 areas to address that (1) a clear taxonomy of hallucination types, (2) the analysis of root causes across the LLM life cycle, (3) a review of five main detection methods, (4) current mitigation strategies, and (5) benchmarks, verification tools and open challenges.

II.LITERATURE REVIEW

1) AI Hallucination in LLMs: A Comprehensive Literature Review
One of the first major reviews laid out the big picture of LLM hallucination. It defined what they are, why they happen, the main types, and how people were trying to detect and fix them. The paper argued that hallucination verification comes from a gap between patterns the model learns during pre-training and the facts needed for specific tasks. It broke hallucination into intrinsic versus extrinsic and covered only fixes like retrieval-based checks and consistency samplings. The key takeaway was that hallucination isn't a bug you can fully patch by scaling up. It's baked into how likelihood-based training works on Internet-scale data, so we need dedicated detection and mitigation tools.[1]

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15574

2) A Comprehensive Survey: Causes, Detection, and Mitigation
Huang et al. Built an earlier work by creating a 2D taxonomy. They separated factuality hallucinations when the model contradicts facts or just makes things up from faithfulness hallucinations, which include instruction, context and logical inconsistency. For each type, they traced root causes across the LLM pipelines from base data and model architecture to RLHF sycophancy and decoding choices. Their breakdown of detection and mitigation methods became a go-to reference and helped shape later research.[2]

3) xVerify: Efficient Answer Verifier for Reasoning Model Evaluations
Chen et al. Noticed that regular string matching doesn't work when you are checking complex answers from reasoning models. So they created xVerify, and an answer verifier trained on VAR- a data set of QA pairs from different LLMs. They built models from 0.5B to 32B parameters. Even the smallest, xVerify-0.5 B -1, beat most baselines except GPT-4o, and the 3B version outperformed GPT-4o overall. This showed you can get strong verification without massive compute.[3]

4) A Framework for Hallucination Detection in LLMs
Deshpande and Li introduced a full framework focused on continuous improvement and finding root causes. They grouped isolation sources into three types: model-related, like outdated knowledge data, related, like bias training sets and contains, like value prompts or forgetting details in long search. They used detection methods like uncertainty estimation and consistency checks with targeted fixes like grounding the model in external knowledge or calibrating confidence. Their big finding was that a tailored solution rag for stale knowledge, recalibration for sycophancy beat generic patches.[4]

5) The Internal State of an LLM Knows When it's Lying
Azaria and Mitchell showed that an LLM's hidden layer actually leaks whether it's being truthful. By training a classifier on those internal activations, they got 71% to 83% accuracy in telling true statements from false across multiple models. This worked better than just looking at output probabilities, because token probabilities get skewed by sentence length and word frequency; their work opened up internal state analysis as a serious way to detect hallucinations.[5]

6) Large Language Models Hallucination: A Comprehensive Survey
Alajmi and Lytras put together one of the most recent, thorough services. They organised detection methods into five groups: retrieval-based, uncertainty-based, embedding-based, learning-based, learning based and self-consistency. They covered cause across the old LLM life cycle and reviewed fixes from the prompt tricks and drag to model level changes. They also looked at benchmarks and metrics. Their conclusion was clear: no single method solves every type of alternation; hybrid approaches work best. They also highlighted gaps in multilingual support, low resource settings and health nation in agentic, multi-step tasks.[6]

7) LLM- based Corroborating and Refuting Evidence (CIBER)
Wang et a. extended the Rag with CIBER, a design for checking scientific claims. Instead of digging into model weight, CIBER probes the same claim with many paraphrases and checks if the answers stay consistent. This works for both open and closed models and doesn't need label data. In testing across different scientific areas, CIBER outperformed standard rag baselines on LLMS with varying domain knowledge.[7]

8) TruthfulQA: Measuring How Models Mimic Human Falsehoods
Lin et al. created TruthfulQA, a benchmark with 817 questions across 38 topics like health, law, finance, and politics. The goal was to test whether LLM hallucination gives truthful answers instead of repeating common human misconceptions. The questions were designed to trick models into repeating false beliefs they might have learned from training data. The result was surprising; the best model was truthful only 58% of the time, while humans scored 94%. Even worse, bigger models tended to be less truthful, suggesting that scale alone makes models better at copying popular myths rather than correcting them. Truthful quickly became the go-to benchmark for truthfulness and pushed a lot of new work on detecting and reducing hallucination.[8]

9) Compass Verifier: A Unified and Robust Verifier for LLMs
Compass verifier tackles a key problem in LLM evaluation. Basic string machine misses answer that means the same thing but are worded differently, and general LLM judges aren't reliable for strict true or false grading. Compass verifier works as a unified tool for both model evaluation and RL reward calculation. It can tell if two answers are semantically equivalent and pulls out clear responses from long, message chains of thought text. It performs well across reasoning tasks and model sizes. This links hallucination checking to the bigger challenge of automatically evaluating reasoning models.[9]

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15574

10) A Comprehensive Survey of Hallucination Mitigation Techniques
Tonmoy et al. reviewed over 32 mitigation methods and grouped them into four buckets: engineering, retrieval augmented generation, self-refinement, and decoding strategies. They looked at techniques like RAG, chain of verification, and inference time intervention. Each has trade-offs. Drag is great for grounding facts, but or dice by retrieval quality. Self-refinement doesn't need external data but burns a lot of compute. Decoding tweaks can improve factuality without extra tools, but might make outputs less diverse. The big takeaway was that no single fix works everywhere. The best results come from combining methods based on the type of hallucination and where the model is deployed.[10]

III.OBJECTIVES

1. Sort out different types of hallucination verification in LLMs: We will build a clear taxonomy that separates factuality errors, where the model gets facts wrong, from faithfulness errors, where it drifts from the prompt or context.

2. Dig into why hallucination happens: This means looking at the full LLM life cycle from how data is collected and cleaned to model architecture choices, training methods, and what goes on during text generation.

3. Review the main ways to detect hallucination: We will cover approaches like retrieval-based checks, uncertainty estimation, using internal model states, training classifiers to spot hallucinations, and directly verifying the answers.

4. Test out strategies to reduce hallucination: We will compare techniques like better prompt design, retrieval augmented generation, reasoning-based verification, and tweaks to the model itself to see what actually helps.

5. Go over the tools we use to measure truthfulness: This includes benchmarks and metrics like truthful create HaluEval, VAR, and CIBER that tell us how factual and reliable an LLM outputs really are.

6. Highlight what problems are still open and where to go next: We will focus on tough areas like hallucination during long context reasoning, in agentic workflows, getting models to know when they are unsure, and performance in multilingual or low resource setups.

IV.METHODOLOGY

The proposed verification detection system for LLM that works like a step-by-step pipeline instead of using just one trick, it pulls together a few proven methods, checking facts against retrieved sources, looking at the model's internal signals, estimating uncertainty and verifying the final answers. The whole process follows the album from start to finish. It begins by breaking the model's response into individual claims, then grabs evidence to check them, runs them through a classifier, and finally tags each part as alternated or not with the confidence score.

A. System Architecture:
Other hallucination detection framework for LLM is split into 4 other hallucination detection framework for LLM is split into four connected models: input module, processing module, detection module, and output model. Each one has a clear job, so we can systematically catch and report any made-up claims.

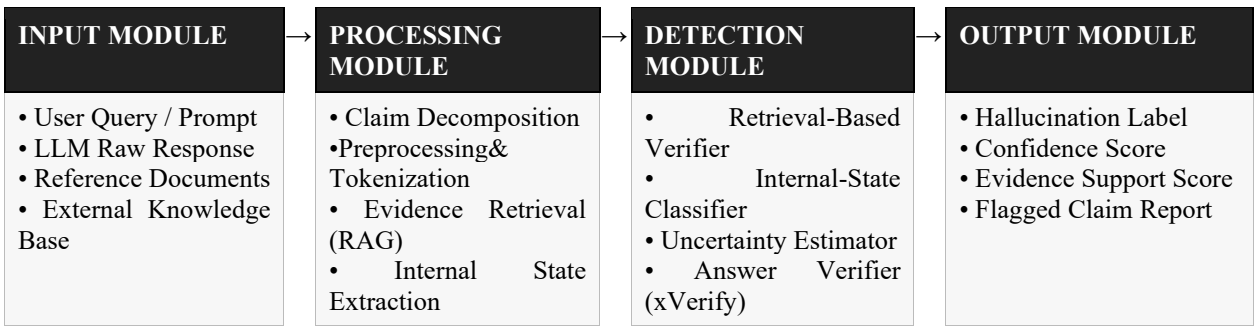

Figure 1: Block Diagram of System Architecture

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15574

1) Input Module
As shown in Figure 1 above, the input module starts. It pulls in the user prompt, the wrong answer from the LLM, any reference documents, and the external knowledge basis. Basically, it gathers all the data we need to check if the model's claims are actually true.

2) Processing Module
Each year, we break the LLMS response down into individual facts or claims. Then we clean them up and tokenise them. After that, we use rag to pull in evidence for each claim and also grab hidden state data from the model to see what's going on internally, by referring to Figure 1 above.

3) Detection Module
This is the code checking stage, as shown in Figure 1 above. We run each claim through four different detectors:
a) A retrieval-based verifier double-checks claims against real-world evidence.
b) internal state classifier looks at the model's hidden activations to spot signals that might hint at lying.
c) uncertainty estimator measures how confident the model is and how fuzzy its prediction is.
d) Answer verifier xVerify makes sure that the reasoning actually matches up and makes sense.

4) Output Module
As shown in Figure 1 above, finally we pull everything together. This model combines the results from all the detectors and labels each claim, adds a confidence score, and rates how well the evidence supports it. It then splits out a clear report that flags risky claims and shows how reliable the facts are.

B. Workflow of the proposed system:
The workflow follows a sequential pipeline that processes the LLM response claim by claim, as shown in Figure 2 below:
1) The system takes in the user's query and the raw response from the LLM.
2) It breaks the response into small, individual factual claims using a claim decomposition model.
3) For each claim, it uses RAG to pull relevant evidence from an external knowledge base.
4) It extracts the LL Mississippi hidden state activations and sends them to an internal state classifier.
5) It calculates token-level entropy to estimate how uncertain the model was about each word.
6)For reasoning-based answers, it uses the xVerify verifier to check if the model's conclusion matches the reference answer.
7) It combines output from all four detectors with a weighted ensemble to assign a final hallucination label and confidence score.
8) It returns a flagged claim report that shows which claims are risky and unreliable the facts are.

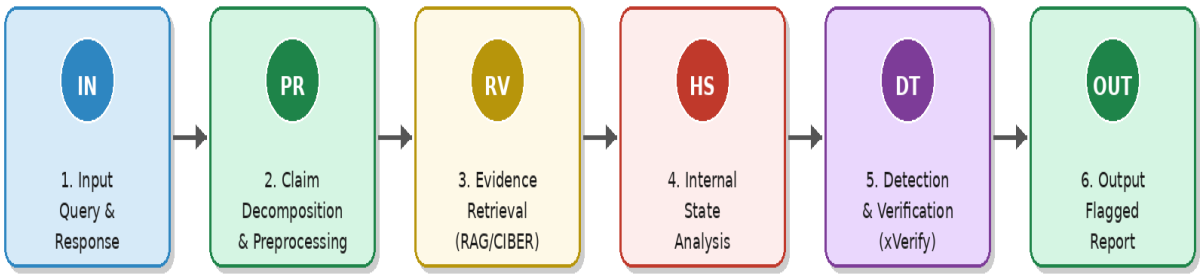

Figure 2: Workflow of the proposed system

C. Integration with knowledge base:
The system uses a Hallucination Knowledge Base (HKB)to store the structured information needed by the detection pipeline.
1) The HKB contains factual documents that are indexed for dense retrieval. This allows the retrial-based verifier to find evidence that supports or contradicts each claim.
2) The HKB stores pre-trained weights for the internal state classifier, which were trained on the SAPLMA data set. Their weights let the system analyse the LLMs' hidden states without relying on external evidence.
3) The HKB includes the model weights for xVerify and Compass Verifier. These tools are used to check answer equivalence in reasoning model evaluation pipelines

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15574

4)The HKB stores benchmark data sets such as truthful QA. These data sets are used for evaluation, calibration, and tuning detection thresholds.

The proposed methodology gives an efficient and systematic way to detect hallucination verification in LLMs. By combining retrieval-based grounding, internal state analysis, uncertainty estimation, and learning the answer verification module pipeline, the system provides a comprehensive solution. Integrating the calculation knowledge base also makes the system adaptable to new benchmarks and domain-specific knowledge.

V.CONCLUSION

This survey gives a structured look at destination verification detection in large language models. It covered the taxonomy, root causes, fire detection paradigms, ways to reduce hallucinations, and the benchmarks used to evaluate them. The modular detection framework we proposed to combine 4 methods into one pipeline: retrieval-based verification, internal state classification, uncertainty estimation, and answer verification. No single method can catch every type of isolation. The best results come from combining these complementary approaches. As LLMs keep evolving, detecting the hallucination will remain a key challenge in building AI systems that people can trust.

VI.ACKNOWLEDGMENT

We wish to extend our sincere appreciation to **Mrs Asha Sattigeri**, Assistant Professor, Department of Computer Science and Engineering, for the invaluable and constructive input provided throughout the planning and development of this project. We are truly grateful for her generous dedication of time. Additionally, we would like to express our thanks to the esteemed professors of KSIT for their unwavering support and encouragement.

REFERENCES

[1]. V. Rawte, A. Sheth, and A. Das, "AI Hallucinations in Large Language Models: A Comprehensive Literature Review," arXiv preprint arXiv:2309.05922, 2023.
[2]. L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A Comprehensive Survey of Hallucination in Large Language Models: Causes, Detection, and Mitigation," arXiv preprint arXiv:2311.05232, 2023.
[3]. D. Chen, Q. Yu, P. Wang, W. Zhang, B. Tang, F. Xiong, X. Li, M. Yang, and Z. Li, "xVerify: Efficient Answer Verifier for Reasoning Model Evaluations," arXiv preprint arXiv:2504.10481, 2025.
[4]. A. Pesaranghader and E. Li, "Hallucination Detection and Mitigation in Large Language Models," arXiv preprint arXiv:2601.09929, 2025.
[5]. A. Azaria and T. Mitchell, "The Internal State of an LLM Knows When It's Lying," in Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 967–976, 2023.
[6]. A. Alansari and H. Luqman, "Large Language Models Hallucination: A Comprehensive Survey," arXiv preprint arXiv:2510.06265, 2025.
[7]. S. Wang, J. R. Foulds, M. O. Gani, and S. Pan, "LLM-based Corroborating and Refuting Evidence Retrieval for Scientific Claim Verification," arXiv preprint arXiv:2503.07937, 2025.
[8]. S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," in Proc. 60th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 3214–3252.
[9]. X. Jiang et al., "CompassVerifier: A Unified and Robust Verifier for LLMs Evaluation and Outcome Reward," arXiv preprint arXiv:2508.03686, 2025.
[10]. S. M. T. I. Tonmoy, S. M. M. Zaman, V. Jain, A. Rani, V. Rawte, A. Chadha, and A. Das, "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models," arXiv preprint arXiv:2401.01313, 2024.

