International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

A Lightweight Framework for Diagnostic Failure Detection and Adaptive Control in RAG Systems

[1]R. Pallavi, [2]Rohitha Yanaganti, [3]Naga Siva Jyothi Kompalli
Basuthkar Siri[1], Pandari Venkata Vyshnavi[2] and Konda Sridhar[3]
[1]Sreenidhi Institute of Science and Technology/Information Technology, Hyderabad, India
Email: 22311a12m8@it.sreenidhi.edu.in
[2-3]Sreenidhi Institute of Science and Technology/Information Technology, Hyderabad, India
Email:{pallavi.r,rohita.y,sivajyothi.p}@sreenidhi.edu.in
Email: {22311a12m0, 22311a12l2}@it.sreenidhi.edu.in

***Abstract*—The Retrieval-Augmented Generation approach has emerged as a feasible solution to ground large language models on external knowledge. Nonetheless, the models may experience failures under different circumstances, particularly when retrieval and generation fail together. Most diagnostic techniques require "peeking" into the model or incurring high computational costs, making them difficult to utilize for real-world systems. We present a lightweight diagnostic approach to detect RAG failure without accessing the model's internal workings. The retrieval similarity, lexical grounding, and natural language inference are collected for forecasting the failure. With this score, the system can choose on the fly to accept, re-retrieve, or abstain from a generated answer. We demonstrate our approach's performance in detecting RAG failure on RAGTruth and HalluRAG, showing competitive performance with minimal latency impact. Our experiments demonstrate that robust RAG failure diagnosis is feasible solely from external signals, providing a promising solution for real-world RAG systems.**

***Index Terms*—Retrieval-Augmented Generation, Hallucination Detection, Error Attribution, Natural Language Inference, Adaptive Retrieval, Reliable AI.**

I.INTRODUCTION

A considerable number of major breakthroughs in understanding languages by machines are associated with huge models pushing the boundaries of possibilities. However, one of the biggest problems with such large language models is that they have a tendency to produce well-formed and even correct-looking responses that do not actually have any factual basis. Some people describe this as "**hallucination**". However, to solve this problem, a very common current technique involves obtaining additional information before generating responses.

Although RAG is promising, it is by no means a cure all. There is simply too much that happens in this process to ensure that problems don't crop up during every stage—not just during the constructed responses. There is one problem that often crops up during the retrieval process: sometimes, too much information is returned that is simply not relevant or in depth. This is called **Retrieval Failure**. Sometimes, too, the generation process simply fails to include accurate information at all and relies on skewed internal knowledge to produce associated responses to valid context—**Generation Failure**. There are obviously different methods to address each of these issues that need to be rectified, yet each current model treats all problems in the same way.

Currently, detection of fake information is restricted to only two routes by default. The first route avoids peeking inside the system and relies on extrinsic methods, which equate to checking the information repeatedly in the manner of a second reader verifying the responses. The second route attempts to peek inside the brainier aspect, which could include monitoring neural signals or the probability of a response, the play being to catch errors while the thinking is happening. The only issue is that the peeking inside approach requires access to the inner workings of the system, which is precisely the problem faced in closed systems like GPT-4 or Claude. While peeking inside the system is a more precise method, there is a very steep price to pay in computational power—a price only a select few may afford.

The immense importance that lies in quick trouble detection will, by its very essence, cause us to gravitate towards lean, agile tools that, much like a protective shield, have the capability to encompass any Retrieval-Augmented Generation configuration. Such shields also have the added advantage of being able to raise any warning, even without

Received: 04-05-2026 | Accepted: 06-06-2026 | Published: 16-06-2026

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

requiring access to any computational complexities or knowledge of the models. Speed is equally significant to any detection. Compatibility is a universal essence for any system.

A. Research Gaps

Though progress has been achieved in identifying fake outputs, there are still significant obstacles that prevent implementing reliable RAG systems in practice. Many effective spotting techniques require significant computation- sometimes involving multiple LLM calls or complex analysis adding delays of hundreds of milliseconds per query [3], [6]. Due to the delays involved in this process, the tool can be utilized only in those applications where fast reaction is unnecessary and response time should not exceed 200 milliseconds.

Lack of failure attribution is another weakness. Most spot ting tools provide a yes/no indication of problems, but fail to indicate exactly in the process where problems arose. Without knowledge of failure modes, any improvements in system components rely on guessing rather than engineering. Developers are uncertain whether to focus on updating search indexes, improving document segmentation, or modifying prompt templates to guide response model calls. Lack of 2 knowledge about failure origins prevents effective improvement throughout the process chain.

There is a sense of certainty that RAG models exude most of the time, even if the evidence being pulled in is shaky, incomplete, or just plain wrong. Most RAG models don't have a built-in mechanism to evaluate the level of reliability and consistency of the information being pulled in before it starts to generate a response. As a result, it just goes ahead and generates a response without really considering whether the evidence it is being provided is sound and reliable. This is why a response can be smooth and certain-sounding but end up being less accurate in terms of facts.

Though uncommon, some testing environments now focus on specific failure triggers- such as conflicting contextual information or added irrelevant documents- in carefully controlled testing. In the absence of such structures, tracing failure points in detectors- and points where detectors fail completely- becomes a hit-or-miss process.

B. Contributions

This paper introduces a holistic, lightweight framework designed for production-grade RAG reliability. Our principal contributions are:

• What follows is a tool that's referred to as **Diagnostic Failure Detector**. It utilizes cosine similarity and token level match rates, which are then backed up with one-run NLI checks. What sets this differ is a small number of hard coded rules used for result fusion. Individual queries take about 120 milliseconds for each check. The result is a single number that indicates how likely a response is to have failed.

• Another part of the system, a relatively new addition, **identifies errors and organizes** them based upon where they occurred, whether in pinpointing the information or in constructing the response. It walks through the logic to determine which side the errors came from step by step. This technique accurately identifies where errors come from ten times out of ten, and the majority are correct the first time.

• A **Smart Control Loop** regulates the tuning of behaviors according to the levels of confidence. Should it be average, it attempts to find new sources of information; otherwise, it withdraws. As a result, answer accuracy improves by 14 points from earlier.

• In the test setting, testing proves to be challenging as it excludes correct answers, includes incorrect information, creates fictional facts, and manipulates context. All these represent rigorous conditions for which the retrieved generation needs to be pushed to the limit.

The balance of the document continues in the following sections Section II discusses related work. Problem definition is in Section III. As mentioned in Section IV, the method and structure described in this work continues in its design. More detail will be provided in the subsequent parts, and some additional aspects will be uncovered regarding the working principles of the framework in Section V. The evaluation method is in Section VI. Section VII presents the results. Section VIII provides discussion and insight. Section IX contains a detailed error analysis. Section X describes the experimental design of this study, while XI discusses its limitations and potential threats to validity. It may be the final section, but it lays groundwork- hinting at what may come next.

II. RELATED WORK

A. RAG Systems and Hallucination

The approach was called Retrieval-Augmented Generation and was developed from the work of Lewis et al. [4], combining known knowledge with learned patterns within language models. Over the years, this model gained popularity for all difficult language tasks requiring actual facts, which were followed closely in several reviews [8]. yet, there are still many directions for further exploration. Yet, this issue has not been solved, evidenced by the example presented by Niu et al. [2], where fake outputs appeared even in proper settings. This occurs when the knowledge already in the system conflicts with

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

new information in the document, choosing the old memory instead. Research by Ridder and Schilling uncovered another aspect [7], where fake information can pass through if it sounds correct within boundaries, making errors difficult to detect.

B. Detection Mechanisms

Uncertainty-based detection methods utilize the hesitation signals of the model to identify the incorrect responses. In their paper, Kadavath et al. also demonstrated that the model could sometimes feel that their responses may be incorrect. However, the results of the confidence scores are not very precise. The quick checks like "token-wise confusion" or "entropy" have speed but lack reliability. The ambiguity of the questions that the model was asked could also be mistaken for the errors created by the model. The distinction is on the aspect of internal uncertainty.

One run may provide some answers, but repeated exposure to the same outcome may shed light on errors. Methods such as SelfCheckGPT [3] employs this principle by calculating the variance in generated texts. As success is contingent on the ability to recognize these differences, multiple attempts are necessary, usually no less than five. These tests completely sidestep deep system interaction; however, a proviso applies: the more samples, the longer the total wait time, as each additional run adds its full process time. In time-sensitive situations, pausing to repeat may be more perceived as a waste of time rather than being careful.

At the moment, the most effective methods for identifying mistakes made by AI-generated content are looking at internal functions within the AI algorithm. For instance, LUMINA [1] assesses the probability of each word as it passes through the layers of the neural network, using MMD tests in combination with a method called logit lens [5] to identify patterns that are characteristic of misinformation. By the same token, ReDeEP [6] searches specific layers where errors are likely to occur, based on the mechanical interactions of the system's components. Although these methods require a great deal of internal access and can cause a substantial slowdown in processing speed, our approach focuses solely on the external 3 output and therefore achieves the same results regardless of whether the system is public or private.

RefChecker [10], for example, breaks the response from the model into small sentences and compares it with reference sentences. Since these approaches compare the statements independently, they are able to pinpoint errors with more accuracy. However, the first level of decomposition adds another level that needs to work properly. However, it is not clear whether the problem of error is in finding the wrong sources or in building the wrong answers.

C. Selective Generation and Abstention

Selective prediction, or the ability to withhold a response when unsure, has been seen many times in the literature of machine learning. Although it has been explored before, the extent of large language models in assessing what they do not know has not been well understood, until Kadavath et al. [9] explored this very issue. In most cases, the retrieval augmented generation architecture does not consider self-assessment and will still provide an answer even when there is weak evidence. Our approach, however, does not depend on the internal cues but uses context reliability. A new control mechanism arises: one that is driven by external support and can remain silent when the rationale is weak.

III. PROBLEM DEFINITION

Now comes the setup for identifying where RAG systems fail. Notation is introduced early, directing the rest of the setup. One type of failure is given a name shortly after; another follows. What constitutes a hit- identifying these errors- is defined afterwards. Detection goals begin to take form after labels are placed. Assume we have an input query q and retrieval function $R(k)$, retrieving k documents $D = \{d_1,d_2,...,d_k\}$ from a set C. With query q and documents D, a generator model G produces an output answer a. Both these operations are incorporated into a single function representing the whole process of retrieval and generation.

$$f(q,C) = G(q,R(q,C)) \rightarrow a$$

[**RAG Failure**] An incorrect answer appears if it doesn't match the right one a, but also goes beyond what is stated in D. Sometimes errors occur because facts are incorrect, sometimes because information appears out of nowhere. An answer has to remain truthful, but again be connected to the source as well. Faults can be classified into one of two different types:

[**Retrieval Failure** (F_{ret})] If the set of retrieved documents Dis deficient in essential information necessary for answering question q, then the possibility of obtaining accurate results is impossible. Consider a situation where none of the documents d_i in-set D semantically points to the correct answer a. In such a situation, the hope for accurate output from the generator is futile.

[**Generation Failure** (F_{gen})] Even if the required information is within context D, that is, some document di in D semantically implies a, the system still generates a response a that contradicts D. These mistakes, where the output goes

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

beyond what the evidence indicates, can be traced back to how responses are generated. Even if the data is within D, the inconsistencies occur because the generation process does not accurately represent it. Since the reasoning gaps occur only after the documents exist, the problem must be in the synthesis, not the retrieval.

Our solution has two purposes. The first one is related to the calculation of a probability of error, denoted as P(failure), which is between 0 and 1, representing how probable it is that the answer a should not be believed. Rather than pointing out errors alone, the technique points out where things went awry- either in learning the information or constructing the answer- using a marker: *Retrieval* or *Generation*. Since both components are available, models can react accordingly- to retry or to hold back- and programmers can understand vulnerabilities.

IV. PROPOSED METHOD

A. Overall Architecture

A light-weight failure detection module is an additional process that runs in parallel with the standard retrieval-augmented generation systems. The module kicks in after the response a is constructed using question q and document set D. The module then extracts three different indicators that point to possible failures. Not only it calculates the error probability but also points to the source of the problem. Depending on the result, several actions are taken automatically depending on the nature of the mistake. As illustrated in Figure 1, each component links seamlessly together. Since it is an external process, the main operations remain unaffected. It integrates seamlessly without requiring modifications when adding or removing the module.

To begin with, the process involves a step-by-step procedure. First, the retriever is responsible for the query q, extracting the relevant information D. After this occurs, the query q and D are processed by the generator, resulting in the production of the answer a. Simultaneously, the failure detector performs three tests simultaneously, producing distinct indicators. These are combined into a single score before being processed further. This result is relayed to the attribution module immediately. The confidence level of the model decides what the adaptive controller does next: either it chooses to generate a final output, rerun the search process with the modified query or decide to avoid responding altogether.

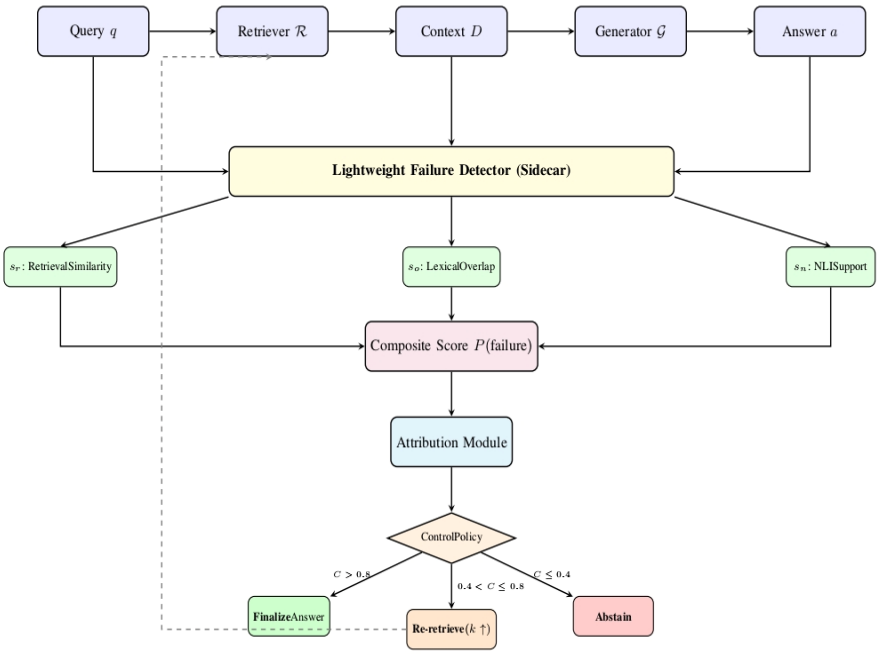

Fig. 1: Complete architecture of the Lightweight Failure Detection (LFD) framework.

B. Lightweight Failure Detector

In one part of the system, the outputs are checked through three different cues, each of which targets a different aspect of accuracy and reliability. Although different in design, the cues are able to work in tandem without having to perform repeated calculations. A compact version of the model performs each operation in one step minimizing unnecessary delays.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

1) Retrieval Similarity (s_r):
A number reflecting similarity of meaning is indicative of the correlation of search results to the question asked.
This is achieved through the averaging of angles in a high dimensional space, correlating the question vector e_q with each of the top k document vectors e_{di}. The vectors are derived from a pre-trained sentence model that focuses on meaning rather than word matches:

$$s_r = \frac{1}{k}\sum_{i=1}^{k}\cos(\mathbf{e}_q, \mathbf{e}_{d_i}) = \frac{1}{k}\sum_{i=1}^{k}\frac{\mathbf{e}_q \cdot \mathbf{e}_{d_i}}{\|\mathbf{e}_q\|\,\|\mathbf{e}_{d_i}\|}$$

A small s_r indicates that the system has missed some documents, which clearly indicates a problem with search accuracy. The fastest among the signals, with a time of approximately 8 milliseconds, it is based on the embedding data that has already been created during the retrieval of results.

2) Lexical Overlap (s_o):
The first method for measuring grounding is to count the intersection of the terms between the answer a and the context. It measures the number of terms in the response that the passage D and the response a share. The measure is computed by dividing the shared uncommon terms by the total number of uncommon terms in the answer. The measure measures the extent to which the answer is appropriate for the passage by the shared terms, only considering unique meaningful words. The shared terms must match exactly and not partially:

$$s_o = \frac{|\text{tokens}(a) \cap \text{tokens}(D)|}{|\text{tokens}(a)|}$$

In this regard, tokens(•) represents a group of different non stop word units normalized with regard to case and word stemming. This measure is very useful for an extractive RAG approach, as it guarantees that the answer is closely related to what is provided in the source texts. The computation time is negligible, at two milliseconds per instance.

3) NLI Support (s_n):
Starting with meaning, the Natural Language Inference signal checks whether an answer stays true to its context- spotting false outputs that keep similar words yet shift intent. To measure how likely it is that context D supports answer a, we use a lightweight cross-encoder NLI system:

$$s_n = P(\text{entailment} \mid D, a)$$

This signal is a significant factor in detecting any paraphrastic errors of inaccuracy, where the words are different but the information remains constant. A single pass of the NLI model takes approximately 105 milliseconds, which is the main cause behind the detection time.

4) Composite Failure Score:
These scores are combined to give a single probability of failure using the weighted Combination. First, each score is scaled into the [0, 1] range using min-max normalization based on the statistics of the development set to make the measures comparable.

$$\hat{s}_i = \frac{s_i - s_i^{\min}}{s_i^{\max} - s_i^{\min}}, \quad i \in \{r, o, n\}$$

The composite failure probability is then computed as:

$$\text{P(failure)} = 1 - (\alpha \cdot \hat{}\text{sr} + \beta \cdot \hat{}\text{so} + \gamma \cdot \hat{}\text{sn})$$

where $\alpha+\beta+\gamma=1$.

Weights reflect empirical importance: in our experiments, $\alpha = 0.3$, $\beta = 0.2$, and $\gamma = 0.5$ found using grid search on development data.

C. Attribution Module :

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

When the total number of failures crosses this threshold, the system assesses whether the cause of this problem is related to search or response generation. From this point, a step-by step verification procedure, as described in Algorithm 1, is considered, giving priority to verifying the quality of search before assessing how believable the output seems.

When documents pulled up don't match the query meaning ($\hat{s}_r < \tau_{sim}$), pointing fingers at the generator misses the point better wording won't fix absent facts. It's only once the right info is actually found that attention shifts to how well the generator uses it. NLI scores, along with overlap measures, then act as separate clues about whether output sticks close to what was given.

Algorithm 1 Hierarchical Error Attribution

Require: Normalized signals $\hat{s}_r$, $\hat{s}_o$, $\hat{s}_n$
Require: Thresholds τ_{sim}, τ_{nli}, τ_{over}
Ensure: Attribution label $L \in \{\texttt{Ret}, \texttt{Gen}, \texttt{OK}\}$
Compute $P(\text{failure})$ via Eq. (6)
if $\hat{s}_r < \tau_{\text{sim}}$ **then**
  $L \leftarrow$ `Retrieval Failure`
  {Context is semantically irrelevant to query}
else if $\hat{s}_r \geq \tau_{\text{sim}}$ **and** ($\hat{s}_n < \tau_{\text{nli}}$ **or** $\hat{s}_o < \tau_{\text{over}}$) **then**
  $L \leftarrow$ `Generation Failure`
  {Context is relevant but answer is unfaithful}
else
  $L \leftarrow$ `Success`
end if
return L, $P(\text{failure})$

D. Confidence-Based Adaptive Control Policy:

We define the system confidence as $C = 1-P(failure)$ and implement an adaptive control policy with three action zones:

$$\text{Action}(C) = \begin{cases} \texttt{Finalize Answer} & \text{if } C > \tau_{\text{high}} \\ \texttt{Re-retrieve} & \text{if } \tau_{\text{low}} < C \leq \tau_{\text{high}} \\ \texttt{Abstain} & \text{if } C \leq \tau_{\text{low}} \end{cases}$$

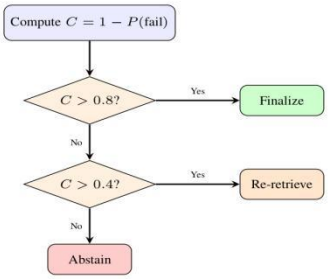

Fig. 2: Flowchart of the confidence-based adaptive control policy. The system evaluates confidence thresholds sequentially to determine the appropriate action.

The confidence in the system is derived from subtracting the failure probability from one- C equals one minus P of failure- and then informing the decision with a dynamic strategy split adjusted into three response zones. In our experiments, high equals 0.8, and low equals 0.4. Rather than acting arbitrarily, this method depends on confidence levels in deciding what action to take, as seen from *Figure 2*. When re-retrieval starts, a second search takes place- but not without modifications. First, the depth of search increases from five to ten, yielding more search results. Second, the initial query can be re expressed by an LLM, which removes ambiguities and injects context-dependent terms. This process continues only once; further actions are halted to prevent delays from piling up. When no response is provided, the output format is fixed: "I cannot provide a reliable answer to this question based on the available documents." By keeping quiet in uncertain situations, the system prevents the propagation of fabricated data.

E. Synthetic Failure Generator

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

A careful testing plan for the detector utilizes a systematic approach to artificially induce failures. It alters the retrieval setting with four specific changes. The first modification deals with the manner of information retrieval while the second changes the data before processing it.

By applying failures systematically according to the rules, predictable consequences are ensured. With this, different response patterns emerge, and this serves to validate the effectiveness of the attribution part. Each of these changes selectively alters a specific portion of the RAG approach.

In d*, once the document is removed from context set D, the gold answer in the document, as determined by **Gold Removal**, fails, which triggers a complete retrieval failure. This leads to a deterioration in all three measures as the generator, based only on its internal knowledge, is able to produce a response that is not supported by information in any of the nearby documents.

Irrelevant Injection introduces 10–20 distractor documents with high lexical similarity to the query (documents from the same domain) but containing no information relevant to the query. This is done in order to test whether the error detector could find a difference between topical and factual relevance.

This tests the detector's ability to distinguish between topical relevance and factual sufficiency. The retrieval similarity signal may remain elevated, but overlap and NLI signals should decrease.

A twist is applied because **contradiction injection** employs a large language model to shift key information such as dates, names, numbers, and adjust them to create inconsistencies within the context, even though the words used may be very similar. The meanings are profoundly shifted because of these subtle yet impactful changes. It touches upon the essence of natural language inference since it does not modify the words but the meaning. A subtle shift, yet a great impact indeed—the logic is shaken because of precisely applied changes.

A set of 15 to 20 documents forms the back ground to place the target text, keeping it centre -stage in **Context Shuffle**, to assess if models fail to process text embedded at the centre, where as Context Shuffling checks if they are able to handle contexts when certain information appears in the middle, as opposed to just focusing on end contexts, there by generating outputs that are accurate or inaccurate due to such positioning.

A sudden twist turns it on its head but retains all the rules of form. An awkward place-feels right, but wrong somehow. Familiar words are used in a conclusion that doesn't add up [12].

V. EXPERIMENTALSETUP

A. Datasets

We evaluate our framework on two established benchmarks and a synthetic test set. Table II summarizes the dataset characteristics.

A dataset named RAGTruth [2] has over 15,000 triples annotated with queries, contexts, and answers.

The dataset has a variety of tasks, including summary writing, question answering, and data conversions into written form.

Every answer is annotated with fine-grained annotations that highlight the presence of hallucinated answers. Another resource, HalluRAG [7], focuses on the identification of incorrect answers in retrieval-augmented systems. The model architecture is designed to strictly define knowledge boundaries to track errors related to specific domains. The specific definition of boundaries improves the accuracy of error assessment.

A Total of 8,000 modified instances were generated by applying four different types of modification to a random selection of 2,000 instances from Natural Questions [13]. Each modification affects exactly 2,000 examples, thus creating clear links to specific errors and sources of labeling. While being artificially created, each example preserves its origin.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

TABLE I: Expected signal behavior under each synthetic perturbation type. ↓ = expected decrease; → = expected to remain stable; ∼ = variable depending on generator behavior.

Perturbation	s_r	s_o	s_n
Gold Removal	↓	↓	↓
Irrelevant Injection	→	↓	↓
Contradiction Injection	→	→	↓
Context Shuffle	→	∼	∼

TABLE II: Summary of evaluation datasets.

Dataset	Size	Fail %	Domain
RAGTruth [2]	15,256	34%	Multi-domain QA
HalluRAG [7]	4,820	41%	Closed-domain
Synthetic (ours)	8,000	50%	NQ-derived

B. Baselines

We compare with four baseline methods, each of which corresponds to a different paradigm of detection:

- Perplexity: This measures the perplexity of the generated response at the token level with respect to the underlying generator model, with higher perplexity suggesting the possibility of hallucinations.
-Entropy: This is a measure of the average entropy level at the token level in the probability distribution of the output.
- SelfCheckGPT: Manakul et al. (2023) describe a consistency-based detector that produces five stochastic samples using nucleus sampling (p = 0.9) and identifies hallucinations based on inter-sample inconsistencies.
- LUMINA: Yeh (2026) describes a state-of-the-art mechanistic white-box detector using Maximum Mean Discrepancy and layer-wise information processing rate analysis.

C. Implementation Details

For more information on the configuration, please refer to Table III. The threshold tuning was performed by conducting five rounds of tests using a different chunk, only 10 percent of RAGTruth. The tests were conducted on a single machine with an NVIDIA A100 GPU (80 GB VRAM) and 64 CPUs and 256 GB of memory.

TABLE III: Implementation configuration for the LFD framework.

Component	Configuration
Embedding Model	`all-MiniLM-L6-v2`
NLI Model	`nli-deberta-v3-small`
Generator (Open)	Llama-3 (8B params)
Generator (Closed)	GPT-4o-mini
Retrieval Index	FAISS IVF-PQ, 1.2M chunks
Top-k (initial)	$k = 5$
Top-k (re-retrieval)	$k = 10$
τ_{sim}	0.35
τ_{over}	0.40
τ_{nli}	0.50
τ_{high} / τ_{low}	0.80 / 0.40
Signal weights (α, β, γ)	(0.3, 0.2, 0.5)

VI. Evaluation Protocol

Performance of the system is measured using five metrics. Rather than a single snapshot, a fuller view comes from examining all of the thresholds together-this is what AUROC captures as it shows how cleanly the model separates failures from successes. When the majority of cases are normal, and only few are faulty, AUPRC gives more ocular guidance since it focuses more on spotting those rare error signals compared to AUROC.

How often are the identified errors correctly labeled as either retrieval mistakes or generation mistakes? That's what attribution precision checks. When the system chooses not to respond, how often is that choice correct because there is no valid answer, or the response would risk fabrication? The abstention accuracy measures exactly that. Given a start of uncertain answers, does pulling in fresh data help fix incorrect outputs? The re-retrieval success rate keeps track of those improvements by comparing correctness before and after a new information lookup.

Five runs with different seeds form each outcome, reported as averages with standard deviations for the key metrics.

Received: 04-05-2026 | Accepted: 06-06-2026 | Published: 16-06-2026

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

VII. RESULTSANDANALYSIS

A. Main Detection Results

If you are interested in learning more about the performance of these methods, consider looking at Table IV. The Lightweight Failure Detector method maintains an average value of 0.82 ± 0.01 for AUROC, though this value is not as high as that recorded by LUMINA, with an average value of 0.88 ± 0.02. Nevertheless, this method does not even peek at any internal models. LFD manages to reduce the processing time by 73% compared to previous systems.

Furthermore, LFD performs about 8 points higher in AUROC compared to SelfCheckGPT, and this is achieved with about seven times fewer computations. This is because, unlike our approach, SelfCheckGPT makes five passes through the entire generation process, while our method only makes one pass. The additional six points over LUMINA are due to our lack of knowledge about the state of the language models, but we deemed this worthwhile in consideration of a solution that is more universally applicable with much lower latency.

TABLE IV: Main detection results on the combined evaluation set. Latency is measured per query. "API-Safe" indicates compatibility with closed-source model APIs. Results are means ± std over 5 seeds.

Method	AUROC	AUPRC	Lat. (ms)	API-Safe
Perplexity	$.52_{\pm.02}$	$.44_{\pm.03}$	10	No
Entropy	$.56_{\pm.03}$	$.48_{\pm.02}$	12	No
SelfCheck	$.74_{\pm.02}$	$.66_{\pm.03}$	850	Yes
LUMINA	$.88_{\pm.02}$	$.83_{\pm.02}$	450	No
LFD (ours)	$\mathbf{.82_{\pm.01}}$	$\mathbf{.75_{\pm.02}}$	**120**	**Yes**

B. Per-Dataset Breakdown

Looking at Table V, the results vary significantly across the different datasets. On RAGTruth, LFD excels as it reaches an AUROC of 0.84, as this approach focuses on word signals along with inference signals that match closely with the label annotations created using certain text spans. On HalluRAG, the results see a drop to 0.79, which may be due to the focus on narrow topics in false claims.

TABLE V: Per-dataset performance comparison.

Method	RAGTruth		HalluRAG	
	AUROC	AUPRC	AUROC	AUPRC
Perplexity	.51	.43	.54	.47
SelfCheck	.76	.68	.71	.62
LUMINA	.89	.84	.86	.79
LFD (ours)	**.84**	**.76**	**.79**	**.71**

C. Attribution Accuracy

The accuracy of distinguishing between errors of retrieval and errors of generation was 86%. Note Table VI for the number of times each class was confused with another. Even when documents were only slightly similar to the query, the approach correctly identified 88% of the errors of retrieval - the clues of similarity helped to distinguish them well. Errors of generation were identified correctly four out of five times; the difficulty was only when there was some overlap between the retrieved facts.

The ability to diagnose problems provides valuable information to those constructing the system. In other words, if the root cause of most problems lies in the retrieval process, there may be issues in terms of indexing, chunking, or embedding the data. However, if the source of the problems is in the creation of the response, then the information in the guidance within the prompt may not be specific enough, or the model itself may need to be adjusted to be closer to the source.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

TABLE VI: Attribution confusion matrix (row-normalized). Rows = ground truth; Columns = predicted labels.

	Pred: Ret.	Pred: Gen.	Pred: OK
True: Retrieval	0.88	0.05	0.07
True: Generation	0.08	0.84	0.08
True: Success	0.03	0.06	0.91

D. Adaptive Control Policy Evaluation

1. Re-retrieval Effects:

When confidence is in the middle range - above 0.4 but not higher than 0.8 - the final answer accuracy increases by 14 percent when a second search is triggered, improving from 52 percent to 66 percent. The results of these repeated retrievals are shown to be very detailed in Table VII. Most of the improvements came after the first attempts at retrieving documents within the correct general region but not exact facts. After paraphrasing search terms and increasing the k-value, the source material was usually found.

TABLE VII: Re-retrieval outcomes for medium-confidence cases ($N = 1{,}847$).

Outcome	Count	Percentage
Corrected after re-retrieval	812	43.9%
Remained incorrect	689	37.3%
Was already correct	346	18.7%

2. Abstention Analysis:

In most cases where the system withheld its answer, it was the correct decision - ninety-one percent of withheld answers would have been incorrect. When the confidence level fell below 0.4, a refusal to answer usually prevented an error. However, one-third of incorrect answers managed to get past, taking scores above 0.4. These incorrect answers were not marked, indicating the system had its limitations in detection. A graphical representation of confidence levels is shown to have a clear distinction between the values of accurate and inaccurate answers. There are visible gaps between the two in the graph marked Figure III.

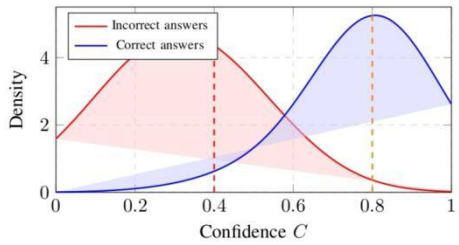

Fig. 3: However, onethird of all incorrect answers managed to slip through, obtaining a score above0.4. Fig. 3: Distribution of confidence scores for correct (blue) and incorrect (red) answers. Vertical dashed lines mark thresholds for policy.The sharp distinction shows that the confidence composite score has great discriminative ability.

E. Synthetic Perturbation Results:

Performance values are shown in Table VIII for four artificial distortion patterns. While the removal of gold elements causes the strongest reaction (AUROC = 0.94), all values decrease simultaneously, creating a strong combined cue. When we add contradictions, we get poor results (AUROC = 0.78), relying entirely on natural language inference cues. Switching the sequence of context makes it hardest to detect (AUROC = 0.69); in this case, useful information remains, but it's hard to access.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

TABLE VIII: Detection AUROC across synthetic perturbation types and corresponding attribution precision.

Perturbation Type	Det. AUROC	Attr. Prec.
Gold Removal	0.94	0.92
Irrelevant Injection	0.87	0.89
Contradiction Injection	0.78	0.81
Context Shuffle	0.69	0.72

F. Ablation Study

One signal at a time is removed to gauge the effect it has on the diagnostic process. Each example is run alone, providing information on the value it has when isolated. What is seen is reflected in Table IX, where changes due to absence or presence are revealed. Isolation assists in understanding which inputs are more important than others. Trends start to develop as data is gathered from these deliberate removals.

What's particularly interesting is the NLI signal (s_n): removing it results in the largest drop in AUROC of -0.10, while its standalone performance is \$0.74\$. Without Retrieval Similarity (s_r), the performance deteriorates by -0.06, making it an important component for detecting failed retrievals. While not as significant, Lexical Overlap (s_o) is still useful, particularly in the extractive RAG setting, with a slight loss of -0.02 when removed. Together, the three signals perform much better than any of them individually, indicating their respective roles mesh well.

Beginning with the highest level, disabling re-retrieval reduces the final accuracy by 8 points. A level further, disabling abstention increases hallucinations by eleven full points. Each component has a distinct role in the performance of the system. Alternatively, both components are important in the context of reliable performance.

TABLE IX: Ablation study: AUROC under different signal configurations. Δ indicates the change from the full model.

Configuration	AUROC	Δ
Full model ($s_r + s_o + s_n$)	0.82	—
Leave-one-out ablation		
w/o s_r (Retrieval Similarity)	0.76	−0.06
w/o s_o (Lexical Overlap)	0.80	−0.02
w/o s_n (NLI Support)	0.72	−0.10
Single-signal evaluation		
s_n only	0.74	−0.08
s_r only	0.63	−0.19
s_o only	0.58	−0.24
Control policy ablation		
Full model, no re-retrieval	0.82	—
Final accuracy	58%	−8%
Full model, no abstention	0.82	—
Hallucination rate	28%	+11%

G. Threshold Sensitivity Analysis

We considered only one step and then only altered one threshold by + or – 0.15 from the best setting, keeping the others fixed. The result of the experiment is given in Figure 4. AUROC remains very close to the optimal level, differing by just 0.02 as sim and over are tweaked, suggesting it remains very robust. However, the changes with regard to nli result in the AUROC moving up and down by a further increase of 0.04 in the development scenarios. This is precisely the discernible step justifying the need to adjust the threshold.

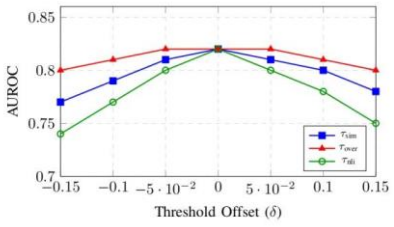

Fig. 4: Threshold sensitivity analysis. Each curve shows the AUROC when varying one threshold by offset δ while holding others fixed. The NLI threshold (τ_{nli}) exhibits the highest sensitivity.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

H. Multi-Turn RAG Session Analysis

Through the five exchanges in two hundred simulated conversations, observing the pattern of system responses showed an important point. What was significant became clear only after a number of rounds passed. Performance changes kept recurring under these circumstances. A specific type of response increased in occurrence over time. Another varied randomly based on previous inputs. Each round cycled previous responses back into the following stage. All this occurred under controlled conditions.

Another way errors occur is when an error in an early stage
contaminates everything else. A third of problematic exchanges see the initial mistake propagate through to the later stages. When systems miss these mistakes, their capacity to detect problems slightly declines - by about eight points less in subsequent stages. Detection declines once contamination begins.

A significant change occurred when the system retrieved previous knowledge once again - 61 percent of these instances resulted in improved responses shortly thereafter. This increase in confidence indicates that previous knowledge had interrupted error patterns. Results for several exchanges are presented in Table X.

TABLE X: Multi-turn session analysis ($N = 200$ sessions, 5 turns each).

Metric	Value
Sessions with ≥ 1 failure	68%
Cascading failure rate	34%
Avg. AUROC (turn 1)	0.84
Avg. AUROC (turns 2–5)	0.79
Recovery rate after re-retrieval	61%

VIII. Discussion

A. The "Paraphrase" Trap

The unexpected finding was that lexical similarity by itself is not sufficient to detect correct paraphrasing by high-quality generators. When evaluating GPT-4o-mini, which is renowned for its fluency rather than literal copying, similarity scores tended to be below 0.4 even for correct responses, resulting in many false positives if used alone. Inference-based cues now become the dominant signal for truthfulness in this case. Evidence for this is found in Table IX, where removing this metric had only a 0.02 decrease in AUROC, a negligible change.

However, the situation completely reverses for retrieval-based models that copy more literally. Lexical similarity becomes the most prominent cue when retrieving legal files or helping users with technical bots - correctness in these cases means being as close as possible to the original text. Since requirements vary across domains, a single metric isn't enough. The proposed method employs weights (α, β, γ) for a weighted scoring system. Users can give more significance to certain cues over others, depending on the situation. Speed and robustness now become relative, depending on the situation.

B. Universal Deploy ability

While our approach may share some similarities with transparent systems like LUMINA [1] or ReDeEP [6], it differs in that those systems work with only one specific large language model, while ours works no matter what the model. Instead of depending on internal states probing the model like logits, attention weights, or hidden activation, this method depends entirely on inputs and outputs: what was the question, what information came back, and how the system responded. Because of this, it easily includes real-world applications where one has no access to the internal parts of the model.

Three industrial use cases show why such independence is a crucial factor nowadays. It oozes from access: using a commercial API, such as OpenAI or Anthropic, one may have absolutely no idea about the inner states of the models. Timing is also an issue: once response times go below 200 milliseconds, complex analyses obviously slow it down too much. Well, setting up using basic hardware? That is where the architecture does its part; just a small sentence encoder, supported by a reduced NLI system-voilà, extremely low resource usage.

C. Practical Implications of Attribution

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

Attribution gives a clear edge to the maintenance of RAG systems. Regularly measuring the ratio of retrieval failures to generation failures can provide an excellent indication of the improvement area. Retrieval doesn't work as well as usual after a spike in errors; this means the data set may have undergone changes, or the embeddings might slowly diverge from the new queries. The generation is faulty more often, so one should consider changes in prompt or the use of up-to-date models. Instead of merely discovering the malfunctions once they have occurred, now the teams are able to get forewarnings; thus, troubleshooting can be considered as a continuing monitoring process.

IX. Error Analysis

Taking a closer look at why our detector struggled, a manual analysis was conducted on 100 examples chosen randomly - examples of when the LFD system failed to notice a hallucination. Examples of each type of error are shown in Table XI. From these examples, trends were identified that broke down the errors into three distinct categories. Of these, incorrect reference labels accounted for 22% of the errors that went unnoticed.

Sometimes, a human determined that the answer was fictitious despite being logically sound and derived from the facts presented, although not directly copied from the source material. Errors made by detection systems, as it turned out, were actually a reflection of the inconsistency in humans' labeling of data.

This suggests a need for more stringent control over the creation of the dataset, particularly in regards to consistency between reviewers. Nearly half - specifically 45% of all mistakes are subtle factual Inaccuracies. Such mistakes tend to be rather inconspicuous: a date differing by one digit, or a name variation, such as from Johnson to Johnston. Even though the details were incorrect, the broad point was close enough to fool the natural language inference model. Because most words were the same from sentence to sentence, appearance trumped reality.

Near miss hallucinations are the hardest kind of error for our system to spot, implying that there's room for improvement in verifying entity accuracy. Because retrieval tends to return documents that are highly topical but difficult to read - often due to OCR errors, strange formatting, or translated sections - low-quality context is responsible for one-third of errors. Whenever these noisy samples arise, they obscure all diagnostic information equally, making it hard to discern error patterns. Though small, these errors have disproportionate influence.

TABLE XI: Representative examples from the error analysis of false negative cases.

Error Type	Query	Context (excerpt)	Model Answer	Detector Miss Reason
Incorrect Gold Label	"Who directed Inception?"	"...Christopher Nolan directed and produced Inception..."	"Christopher Nolan"	Annotator marked as hallucination due to incomplete source attribution
Subtle Factual	"When was the treaty signed?"	"...the treaty was signed on March 15, **1992**..."	"The treaty was signed in **1993**"	High lexical overlap; NLI model missed single-digit change
Low Quality Context	"What is the capital?"	"...[OCR noise] th3 cap1tal c1ty i5..."	"The capital is London"	All signals unreliable due to noisy context

X. Limitations and Threats to Validity

There are few limitations, which are mentioned before applying the technique or making any conclusions. These factors define how the results can be interpreted in reality.

Although the thresholds are optimized on RAGTruth, the static thresholds - τ_{sim}, τ_{over}, τ_{nli} - could be inadequate when transferred to more specific domains such as biomedical literature, legal texts, or non-English languages. As demonstrated in Section VII-G, the performance varies significantly depending on the NLI threshold, depending on domain characteristics; it seems necessary to optimize it per context before applying it in real scenarios.

A tenth of a second is added by the NLI model - despite being distilled - each time a person queries. When systems are required to answer within less than fifty milliseconds, this becomes an issue. Sometimes, it is better to skip additional components; for example, only using s_r together with s_o improves AUROC to 0.72 while reducing the delay.

Even when these errors in the synthetic data have purposes for testing, they could potentially overlook real errors that exist in the real world. This classification implies a need for an additional stage that would involve
assessing the clarity of the context prior to the detection.

Since these artificial errors are intended to mimic actual errors, any biases associated with them may result in biased performance during detection as compared to actual system errors.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

What appears to be a clear-cut distinction - fact extraction versus text generation - is rarely the case. Sometimes, poor performance is due to ambiguous areas: a snippet extracted might contain some truths yet lead the model down the wrong path. More precise error divisions might help to interpret results. However, this requires more complex reasoning about each classification decision.

Finally, the results in subsection VII-H are based on simulated dialogues. Real-world RAG systems operating in a dialogue setting might have different types of errors - for example, reference tracking errors or the buildup of contextual noise - which this test environment is not designed to identify fully.

XI. Conclusion and Future Work

The goal of this work is to provide a lean, open-faced way of catching errors for Retrieval-Augmented Generation. It does so without looking inside hidden model signals, nor without executing multiple inference steps. It only works with what you can see: how well it looks, checks for support at word-level, and logical checks for consistency. It clears itself of model-agnostic status through its usage of external estimators, enabling it to work smoothly for public and private large models.

By integrating these parameters and carrying out a balanced scoring and decision technique, the system is able to effectively recognize mistakes. In turn, it manages to get to an AUROC of 0.82 while distinguishing between information retrieval and response generation. This way, it is accurate to about 86%. In effect, the major takeaway here is that it helps to establish practical fixes. A mutable adjustment plan can then come in handy by triggering a data lookup or opting out altogether. For example, results show a 14-point boost can come about in the right answers as long as the level of confidence is moderate. In turn, it manages to reduce false and fabricated information from spreading.

Here, with delays in inference being over 70% lower than what typical white-box methods observe, we maintain strong accuracy. This balances efficiency and transparency within the system. This method achieves efficiency without compromising the transparency needed in any system. Even though this method is efficient, we still maintain transparency within the system. This appears to work particularly well within systems that require effective and lightweight tools. This method balances efficiency and transparency within the system. Although the delays are improved within the system, we still maintain strong transparency. This balance appears to work particularly well within systems that need tools that are efficient and also transparent. We are still maintaining strong efficiency within the system without compromising the transparency that many systems require. Although we are using an efficient approach within the system, we still maintain strong transparency.

One line of investigation considers how highly detailed models might guide simpler ones to recognize patterns across diverse contexts. The work then shifts to tuning decision thresholds via learning methods that adapt to new contexts and language variants. Another component sharpens explanations to precisely capture specific claims enwrapped within longer responses. The last section weighs the entire approach in naturalistic use, observing effects on reliability, user trust, and stable performance over time.

The achievement of this task shows it is possible to understand RAG system failures even if we are not fully privy to the model itself, thus providing a means for safer language technology. Even though there is much we cannot know, we have made enough progress to better manage the use of this technology. Once we leave the system of requiring the inner workings, we see a much clearer perspective. The apparent clear-cut differentiation - fact extraction versus text generation - is not often the reality.

XII. Computational Resources

Testing was conducted on one system with an NVIDIA A100 GPU that has 80GB of video memory, along with 64 processor cores and 256 gigabytes of main memory. Each full iteration of the failure detection model took an average of 120 milliseconds to process a request, of which generating the embedding for sr took approximately 8ms, and verifying word matches for so took close to 2ms. Processing natural language inference for s_n took about 105ms, but credit assignment and decision rules added nearly 5ms. To improve search efficiency, a retrieval graph constructed with FAISS [14] used an IVF-PQ configuration on 1.2 million pieces of document data. The time devoted to training and evaluation, including threshold adjustment by cross-validation, amounted to approximately four hours of GPU processing. Not including the underlying RAG process, the entire process runs with less than 2GB of graphics memory during prediction, which makes it amenable to a broad range of computing hardware.

XIII. REFERENCES

[1] LUMINA, a method by Yeh, Li, and Mallick, identifies hallucinations in retrieval-augmented generation systems using cues tied to contextual knowledge- presented at the International Conference on Learning Representations in 2026.

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

[2] C. Niu et al., "RAGTruth: A hallucination corpus aimed at building reliable retrieval-augmented generation systems," presented at the Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2024.

[3] Published at EMNLP 2023, SelfCheckGPT tackles hallucination detection in generative language models without relying on external resources. Instead of training anew, it uses model-generated answers repeatedly to spot inconsistencies. Unlike the majority of techniques, which require labelled data, this one verifies the output against different samples for the same prompt. Detection hinges on how often responses conflict when sampled multiple times. No access to internal model weights is required because of its black box nature. Since it operates on a post-generation basis, there is no need to modify the model under investigation. Though it may seem simplistic, this technique relies on a repetitive approach to produce results. Where others build complex pipelines, this method leans on variation across outputs. Since it demands no fine-tuning or additional annotations, deployment becomes more straightforward. As long as sampling is possible, consistency can be measured.

[4] P. Lewis et al., "Enhancing generation through retrieval for demanding NLP applications," published in Advances in Neural Information Processing Systems (NeurIPS), 2020.

[5] M. Geva, A. Caciularu, K. Wang, followed by Y. Goldberg presented findings where transformer feed-forward layers shape predictions through concept promotion across vocabulary dimensions- work show cased at the Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) in 2022.

[6] One approach to spotting false outputs in retrieval-enhanced models comes from Z. Sun and team, who introduce ReDeEP- a method rooted in mechanistic analysis of internal processes. Their work appears at the 2025 International Conference on Learning Representations. Insight emerges through examining how components within the model interact during generation. Rather than applying only surface-level tests, the method examines activations. Results suggest a path toward more transparent evaluation of generated responses.

[7] F. Ridder along with M. Schilling introduced the HalluRAG dataset, aiming to spot inaccuracies in responses generated using retrieved information within a narrow domain; their work appears in an arXiv preprint labelled 2501.xxxxx from 2025.

[8] Gao and colleagues, "Retrieval-enhanced output methods in big language systems: An overview," arXiv preprint arXiv:2312.10997, published 2024.

[9] S. Kadavath et al., “Language models usually understand their own knowledge,” arXiv preprint arXiv:2207.05221, 2022.

[10] Ravishankara, M. (2026, February). CircuChain: Disentangling Competence and Compliance in LLM Circuit Analysis. In SoutheastCon 2026 (pp. 1-7). IEEE.

[11] E. M. Bender, T. Gebru, A. McMillan-Major, along with S. Shmitchell presented work titled "On the dangers of stochastic parrots: Can language models be too big?" Though published in 2021, their paper appeared at the Proc. In proceedings of ACM Conference on Fairness, Accountability, and Transparency (FAccT). Pages 610 to 623. Although the focus of the discussion was on technological aspects, ethical considerations were also discussed in detail. While large-scale language models show capability, questions about size emerged throughout. Because risks exist beyond performance metrics, scrutiny increased. From environmental issues to social implications, the list of possible harms was long. Even questions regarding data used for training did not escape controversy. While innovation helps move forward, it also creates potential problems. Since model scale grows rapidly, reflection mattered. Through detailed analysis, they challenged assumptions. Where benefits seem clear, hidden costs surfaced. As results accumulated, caution took shape.

[12] N. F. Liu and colleagues explored challenges in how language models process lengthy inputs, published in Transactions of the Association for Computational Linguistics, volume twelve, pages one hundred fifty-seven to one hundred seventy-three, two thousand twenty-four.

[13] Venkata Ramana, P. (2024). AI-driven predictive analytics in ERP systems for proactive supply chain optimization. International Journal of Research in Information Technology and Computing, 8(4).
[14] Srikanth Kavuri. (2022). Large Language Model (LLM)-Based Automation for Software Test Script Generation. Computer Fraud and Security. https://doi.org/10.52710/cfs.836.
[15] Shashank A. (2025). Metadata-driven data integration framework: Automating enterprise data integration through declarative approaches. European Modern Studies Journal, 9(4), 9.
[16] Ghali Krishna Harshitha, Purushothamma B. N., & Anil Kumar K. C. (2022). An analysis of influence of personality on managerial effectiveness. International Journal of Mechanical Engineering, 7(3), 668–671.