

Hallucination Detection Framework for RAG Systems

Claim-Level Multi-Source Verification using Similarity and NLI

Chaudhary Hammad Raza¹, Maria Robin²

Department of Computer Science, Patkar-Varde College, University of Mumbai

Abstract— Retrieval-augmented generation (RAG) has become a standard way to attach an external store to a large language model (LLM). The usual story is that retrieval grounds the answer, and grounding removes hallucination. The story is incomplete. An LLM inside a RAG pipeline can still misread the retrieved passage, mix the passage with unsupported memory from pretraining, or invent a clause when the retriever returns weak text. The leftover risk is worse than ungrounded generation in one respect: a citation sits next to the sentence, so a reader who cannot open the source treats the line as audited. This paper writes up the hallucination detection and multi-source verification framework from our M.Sc. project. Multi-format documents (PDF, DOCX, TXT, images, scanned pages) are ingested, run through OCR when needed, chunked, embedded with Sentence Transformers, and indexed in FAISS, with metadata in MongoDB. A locally hosted generator (Ollama with Qwen3 8B) writes an answer from retrieved chunks. That answer, and independently an uploaded document, is split into atomic factual claims. Each claim is scored with semantic similarity and classified with a natural-language-inference (NLI) model. When the internal store cannot support a decision, a reliability-tiered web fallback is used. Claims receive one of five labels — Supported, Partially Supported, Contradicted / Likely Hallucination, Unverifiable, or Out of Knowledge Base — with evidence, a confidence figure, and a short justification. The intended interface is a React client and a Flask service for a single authenticated role, including an analytics view. Evaluation is specified as an ablation across baseline RAG, similarity-only checking, NLI-only checking, combined signals, and combined signals plus web fallback, using precision, recall, F1, error rates, retrieval metrics, and confidence calibration.

Keywords—retrieval-augmented generation, hallucination detection, natural language inference, claim-level verification, FAISS, explainable AI, multi-source evidence.

1. INTRODUCTION

Large Language Models (LLMs) have made significant progress in producing fluent and coherent text. However, the ability to generate convincing language does not necessarily mean that the information produced is factually correct. When an LLM relies only on the knowledge learned during training, it may generate hallucination statements that sound reasonable but are inaccurate, unsupported, or completely false. Retrieval-Augmented Generation (RAG) was developed to reduce this problem by allowing an LLM to retrieve relevant information

from an external knowledge source during generation. In this way, the model can use retrieved evidence instead of depending entirely on the information stored in its parameters [1].

Although RAG can substantially improve factuality, it does not completely prevent hallucinations. A model may still misunderstand the retrieved information, draw conclusions that are not justified by the evidence, or generate statements that conflict with the retrieved sources. The presence of citations or supporting documents can make such responses appear more trustworthy than they actually are. This creates a significant problem in areas such as academic research, legal analysis, technical documentation, enterprise knowledge management, and medical information, where even one incorrect statement can have serious consequences. In many cases, users would need to manually compare every generated statement with the original sources to identify such errors, which is both time-consuming and impractical.

Current approaches for detecting and reducing hallucinations in RAG systems have several limitations. One common approach evaluates the generated response as a whole rather than examining its individual factual statements. As a result, an incorrect claim can remain hidden within an otherwise accurate response. Another limitation is the reliance on semantic similarity alone. Although similarity measures can determine whether a claim and a retrieved passage discuss similar topics, they cannot always determine whether the two statements actually agree. For example, a generated statement may closely resemble the source while incorrectly changing a date, numerical value, person, organization, or other important detail. A third limitation is the use of a simple hallucination/non-hallucination decision. Such a binary classification does not distinguish between a claim that is contradicted by available evidence and a claim for which no supporting information exists in the knowledge base. These situations are fundamentally different because the absence of evidence does not necessarily mean that a claim is false.

To address these limitations, this paper proposes a claim-level verification framework that treats factual verification as an essential part of the RAG pipeline rather than an optional post-processing step. The proposed system divides a generated

response into individual factual claims and evaluates each claim separately before the final response is presented to the user. The same verification process can also be applied to documents uploaded by users for fact-checking. Each claim is analysed using multiple forms of evidence and assigned to one of five interpretable categories: Supported, Partially Supported, Contradicted/Likely Hallucination, Unverifiable, or Out of Knowledge Base.

The main contributions of this work are summarized below:

- **Claim-level verification:** The framework evaluates individual factual claims instead of treating the entire generated response as a single unit. This makes it possible to identify specific unsupported or incorrect statements while retaining the valid portions of the response.
- **Combination of semantic similarity and NLI:** The proposed approach combines semantic similarity with Natural Language Inference (NLI). This provides a stronger verification mechanism because similarity alone may fail when a generated claim is closely related to the source but contains a contradiction, such as an incorrect date, number, or named entity.
- **Five-category verification scheme:** The system introduces five outcome categories—Supported, Partially Supported, Contradicted/Likely Hallucination, Unverifiable, and Out of Knowledge Base. This classification provides more useful information than a simple true/false or hallucinated/not-hallucinated decision and, importantly, distinguishes contradiction from the absence of relevant evidence.
- **Multi-source web verification:** When the internal knowledge base cannot provide sufficient evidence, the system can use external web sources as a secondary verification layer. These sources are organized into different reliability levels so that information from more authoritative sources can be given greater weight than information from less reliable sources.
- **Reproducible local implementation:** The complete system is implemented using locally hosted and open-source technologies, including Flask, React, MongoDB, FAISS, Sentence Transformers, and Ollama. This design reduces dependence on paid commercial APIs and makes the proposed framework easier to reproduce and evaluate in different environments.

2. RELATED WORK

2.1 Retrieval-Augmented Generation

Lewis et al. [1] introduced Retrieval-Augmented Generation (RAG) as a method that combines a pretrained language model with an external information retrieval system. Instead of depending only on the knowledge learned during model training, the system first searches an external collection of documents and retrieves information that is relevant to the user's query. The retrieved content is then provided to the language model to help it generate a more evidence-based response.

RAG has become an important approach for applications that require access to specific or up-to-date information. It is widely used in question answering, document search, enterprise knowledge systems, and other knowledge-intensive applications. However, later research has shown that providing external context does not completely prevent hallucinations. A model can still misunderstand the retrieved content or generate information that is not supported by it. This has led to increased research into methods for detecting factual errors within RAG-generated responses.

2.2 Hallucination Benchmarks

Several datasets have been developed to study and evaluate hallucinations in language generation systems. RAGTruth [2] is one of the important benchmarks designed specifically for RAG-based applications. It contains approximately 18,000 span-level annotations covering tasks such as question answering, summarization, and data-to-text generation. An important feature of RAGTruth is that it identifies the particular portions of a response that contain hallucinated information rather than assigning a single label to the entire response. This makes the dataset particularly relevant to approaches that perform verification at the claim or statement level.

HaluEval [3] is another large-scale resource developed for hallucination evaluation. It provides examples that can be used to assess whether generated content contains factual errors. However, later studies have suggested that some of the hallucinations represented in HaluEval are easier to identify than errors that occur naturally in real RAG applications. RAGTruth has therefore been considered useful for studying more detailed and realistic hallucination cases. Research has also reported annotation issues in both datasets. This suggests that benchmark results should be interpreted carefully and, where possible, supported by independently prepared test data instead of assuming that benchmark annotations are always completely accurate.

2.3 Detection Signals: Similarity and Natural Language Inference

Semantic similarity is one of the commonly used methods for checking whether generated information is related to retrieved evidence. In many systems, sentence embeddings are generated for both the claim and the retrieved passage, after which their similarity is calculated using measures such as cosine similarity. Sentence-BERT [4] made this type of comparison

more practical by providing a model architecture capable of generating meaningful sentence-level embeddings.

Although similarity is useful for finding relevant information, it cannot always determine whether two statements actually agree. Two sentences may have a high similarity score because they discuss the same subject while still containing an important factual difference. For example, a generated statement could change a date, numerical value, location, or named entity while remaining very similar to the original evidence.

Natural Language Inference (NLI) provides another way to examine the relationship between two statements. It determines whether a hypothesis is supported by a premise, contradicted by it, or unrelated to it. Combining NLI with semantic similarity can therefore provide more information than either method on its own. Previous research has reported improvements when these two signals are used together. Based on this observation, the proposed framework also combines similarity-based retrieval with NLI-based verification.

2.4 Existing RAG Evaluation and Hallucination Detection Tools

A number of open-source frameworks have been developed to evaluate RAG systems. Ragas, for example, provides measures such as faithfulness and context precision. These measures are generally used to determine how well a generated response is connected to the retrieved information. However, some reported limitations include difficulties in correctly identifying and extracting individual statements from generated responses.

TruLens is another framework used for evaluating RAG applications. It uses what is commonly referred to as the RAG Triad, which considers context relevance, groundedness, and answer relevance. Although this provides a useful overall assessment of a RAG system, groundedness is generally represented as an aggregate evaluation rather than a detailed verification of each individual claim.

More focused approaches have also been proposed. LettuceDetect [5], for example, treats hallucination detection as a token or span classification problem and is trained using data from RAGTruth. This approach is closely related to the claim-level verification idea because it attempts to identify specific portions of generated text that may contain unsupported information. However, it uses a specially trained encoder for hallucination detection, whereas the framework presented in this work combines general semantic similarity methods with NLI.

Another group of approaches does not depend primarily on external evidence. Instead, these methods examine the internal behavior of the language model. ReDeEP [6], for example, uses information from attention heads to calculate measures related to external context and the model's internal knowledge. Such approaches can provide information about how a model produces an answer, but they may require adjustments based on the particular model architecture, including the selection of appropriate layers or attention heads. The approach presented

in this paper does not require this type of model-specific analysis because it verifies generated claims using available evidence.

Overall, existing research provides several useful methods for evaluating factuality and detecting hallucinations in RAG systems. However, there are still areas that can be improved. Many existing approaches evaluate responses at a relatively broad level rather than examining every factual claim separately. Some methods depend heavily on one type of verification signal, while others do not clearly distinguish between a claim that is contradicted by available evidence and a claim for which no evidence can be found. In addition, verification is often limited to the information retrieved from the system's internal knowledge base.

2.5 Research Gap

Taken together, this literature leaves a specific gap. Existing tools tend to work at answer level, use a single detection signal, give a binary verdict that merges contradiction with missing evidence, and stop at the boundary of the retrieved knowledge base. The framework in this paper is designed to close these gaps together, and Table 1 summarizes how it differs from the approaches above.

Approach	Granularity	Signals used	Separates contradiction from missing evidence	External verification
Ragas	Response level	LLM-based judgments	No	No
TruLens	Response level	LLM-based groundedness	No	No
LettuceDetect	Token / span level	Trained encoder	No	No
ReDeEP	Response level	Internal model states	No	No
Proposed framework	Claim level	Similarity + NLI	Yes (five-way scheme)	Yes (reliability-tiered)

3. PROPOSED FRAMEWORK

This section gives an overview of the proposed framework and its main components. The detailed working of each stage is explained in Section 4, while Section 5 discusses the implementation of the system.

3.1 Overview

The proposed framework makes verification an important part of the RAG pipeline. In a typical RAG system, relevant

information is retrieved, an answer is generated, and the answer is directly shown to the user. In the proposed system, an additional verification stage is placed between answer generation and final delivery. The generated claims are checked before the response is presented to the user. The same verification process can also be used independently for documents uploaded by the user.

3.2 Two Operating Modes

The framework supports two main modes of operation. In RAG answer verification, the user first submits a question. The system retrieves relevant sections from the knowledge base, and a locally hosted LLM uses this information to generate an answer. The generated answer is then divided into individual claims, and each claim is checked against the available evidence.

In document fact-checking mode, the user selects a document that has already been uploaded to the system. The framework identifies the factual claims made in the document and verifies them using the available evidence. Both modes use the same claim extraction and verification process. This allows the system to apply the same verification method regardless of where the claim originally came from.

3.3 Framework Components

The main components of the proposed framework are:

- Document ingestion and knowledge base: The system accepts PDF, DOCX, TXT, image, and scanned documents. OCR is used when required to extract text from scanned inputs. The extracted text is cleaned, divided into smaller chunks, and stored in MongoDB. The corresponding embeddings are stored in FAISS for efficient retrieval.
- Retrieval and generation: The user's question is converted into an embedding and compared with the stored document embeddings. The most relevant chunks are retrieved and provided as context to a locally hosted LLM, which generates the answer.
- Claim extraction: The generated answer or uploaded document is divided into smaller factual claims. Each claim is designed to be independently checked against available evidence.
- Verification engine: Each claim is compared with the available evidence using semantic similarity and Natural Language Inference (NLI). The results are then processed using predefined decision rules.
- Web verification: If the internal knowledge base does not provide enough evidence, the system can search external sources. These sources are assigned different reliability levels, and their evidence is considered according to those levels.
- Classification and explanation: After verification, each claim receives one of five outcomes along with

a confidence score, relevant supporting evidence, and a simple explanation of the result.

3.4 Classification Outcomes

Each claim is assigned exactly one of the five outcomes defined in Table 2. The framework keeps Unverifiable and Out of Knowledge Base as separate categories. Unverifiable means that the system searched for supporting evidence but could not reach a clear conclusion. Out of Knowledge Base means that no relevant evidence was available in the internal knowledge base and that external verification was either not used or did not provide a clear result. Neither category automatically means that the claim is false.

Classification	Meaning
Supported	Available evidence supports the claim.
Partially Supported	Part of the claim is supported while another part lacks sufficient evidence.
Contradicted / Likely Hallucination	Reliable evidence contradicts the claim.
Unverifiable	Evidence was sought but is insufficient to decide.
Out of Knowledge Base	The uploaded documents hold no relevant evidence. This is not automatically false.

METHODOLOGY

4.1 Problem Formulation

Consider a RAG system that generates an answer A for a user query Q using a set of retrieved evidence chunks $E = \{e_1, \dots, e_n\}$ from a knowledge base K . The generated answer is divided into individual factual claims. The same process is applied to claims extracted from an uploaded document D . The main objective is to determine whether each claim c is supported by the available evidence.

A simple binary decision is not sufficient for this task. The system should be able to identify different levels of support and should not depend only on a similarity score. It should also be able to search for additional evidence when the internal knowledge base does not contain enough relevant information. Based on these requirements, the methodology focuses on four main tasks: extracting individual claims, determining whether enough evidence is available, combining different verification signals, and using external sources when internal evidence is insufficient.

4.2 Claim Extraction

The first step is to identify the individual factual claims present in an answer or document. An LLM is prompted to separate the text into independent factual statements. A rule-based sentence splitter is also used as a fallback and for basic cross-checking.

Simply dividing text at full stops is not always sufficient because a single sentence can contain multiple facts. For example, a sentence may mention both when an event occurred and where it occurred. If the entire sentence is treated as one claim, it becomes difficult to determine which part is supported and which part may be incorrect. Breaking the text into smaller claims allows each factual statement to be verified separately.

4.3 Evidence Retrieval and the Sufficiency Gate

After claims are extracted, each claim is converted into an embedding using a Sentence Transformer. The resulting embedding is compared with the vectors stored in the knowledge base using cosine similarity. This process is used to find the document chunks that are most closely related to the claim.

A sufficiency check is then applied to the retrieved evidence. If the highest similarity score is below the selected threshold, the retrieved content is considered insufficient for reliable verification. Instead of making a decision based on weakly related information, the claim is sent for external web verification when that option is enabled. An initial threshold of approximately 0.45 is proposed, although this value can be adjusted using the labeled evaluation data.

4.4 Multi-Signal Verification

When relevant internal evidence is available, the system uses two main signals to evaluate the claim. The first is the semantic similarity between the claim and the retrieved evidence. The second comes from Natural Language Inference (NLI), where the retrieved document chunk is treated as the premise and the generated claim is treated as the hypothesis.

The two signals are considered together when assigning the final result. When the NLI model predicts entailment and the similarity score is sufficiently high, initially around 0.65 or above, the claim is classified as **Supported**. If the NLI model identifies a contradiction, the claim is classified as **Contradicted / Likely Hallucination**, even when its similarity score is high. This is because direct contradiction provides stronger evidence of an error than general topical similarity.

If the NLI result is neutral, or if the claim is classified as entailment but has only moderate similarity with the evidence, the claim is classified as **Partially Supported**. When multiple document chunks are used to verify one claim, both the strongest supporting result and the strongest contradicting result are retained. This prevents a contradictory piece of evidence from being overlooked simply because several other retrieved passages are neutral.

4.5 Reliability-Tiered Web Verification

If the internal knowledge base does not provide sufficient evidence and web verification is enabled, the system performs an additional search using external sources. The retrieved sources are divided into three reliability levels.

Tier 1 contains government websites, official organizational sources, and established reference sources. **Tier 2** includes recognized and established news organizations. **Tier 3** consists mainly of blogs, forums, and domains where the reliability of the information cannot be established easily.

NLI is applied between the claim and the relevant text obtained from each external source. The result from each source is then given a weight based on its reliability tier. Therefore, agreement from a Tier 1 source has more influence than agreement from several Tier 3 sources. The reliability level is determined using a curated list of domains that can be updated as required. This approach is intentionally kept simple because automatically determining the reliability of online sources is itself a separate research problem.

4.6 Confidence Scoring

Along with the final classification, the system generates a confidence score for each claim. The initial confidence calculation combines the NLI probability, semantic similarity, and reliability of the source using the following formula:

$$\begin{aligned}\text{Confidence} = & 0.5 \times (\text{NLI probability}) \\ & + 0.3 \times (\text{similarity score}) \\ & + 0.2 \times (\text{source weight})\end{aligned}$$

For the source weight, a value of **1.0** is used for evidence from the internal knowledge base or Tier 1 web sources. A value of **0.6** is assigned to Tier 2 sources, while Tier 3 sources receive **0.3**. If no evidence is available, the source weight is set to **0**.

These weights are initial assumptions rather than experimentally established values. The evaluation described in Section 6 includes a calibration analysis to determine whether the confidence values produced by the system correspond reasonably well with its actual verification accuracy.

5. SYSTEM IMPLEMENTATION

5.1 Architecture

The proposed framework is implemented as a full-stack web application. The system separates vector-based searching from general data storage, as illustrated in Figure 2. The React frontend provides the main user interface, including document uploading and management, a chat interface, claim-level verification results, document fact-checking, and an analytics dashboard.

The Flask backend handles the main processing tasks of the system. These include document extraction and OCR, text chunking, embedding generation, FAISS index management, RAG processing, claim extraction, claim verification, web-based verification, and final classification. Some of these operations, particularly document processing, LLM generation, and NLI prediction, may require several seconds to complete. For this reason, longer operations are intended to run asynchronously so that the application remains responsive while processing is in progress.

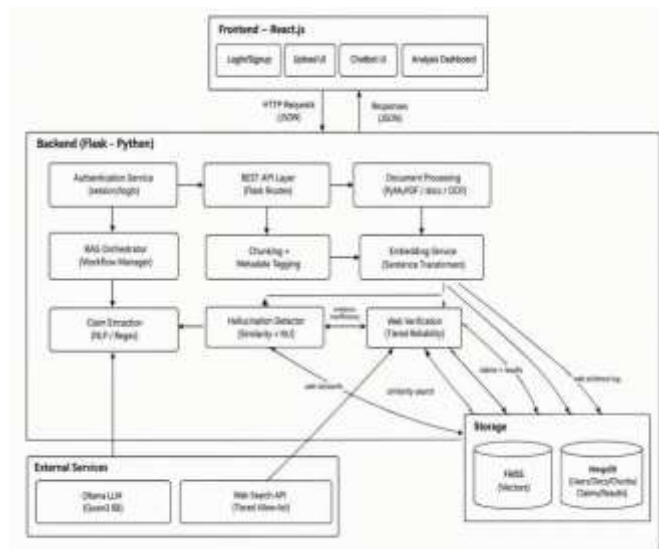


Figure 1. System architecture showing the separation between FAISS for vector search and MongoDB for general data storage.

FAISS is used only for storing and searching document vectors. Its main purpose is to identify the stored chunks that are most similar to a given query. MongoDB handles the remaining application data, including user information, document details, chunk text, page numbers, queries, generated responses, claims, and verification results. The results returned by FAISS contain references that are used to retrieve the corresponding records from MongoDB.

User authentication is implemented using a basic session-based login system with a single user role. This setup is sufficient for the current research prototype and keeps the implementation focused on the verification framework.

5.2 Technology Stack

The main technologies used in the system are listed in Table 3. Open-source tools were selected wherever possible to make the project easier to reproduce and avoid recurring software costs. Established technologies were used for the basic infrastructure so that the main focus of the implementation remains on the claim verification process.

The models used in the initial implementation are also relatively lightweight, which allows the system to run on practical hardware and leaves room for testing stronger models in future experiments. For vector search, FAISS uses an exact flat index. Since the expected dataset is relatively small and is

expected to contain fewer than 100,000 chunks, exact search is considered practical and provides complete recall.

Table 3. Technology stack

Component	Technology
Frontend	React.js
Backend	Flask (Python)
Primary database	MongoDB (PyMongo)
Vector search	FAISS (IndexFlatIP, exact search)
Embeddings	Sentence Transformers, all-MiniLM-L6-v2
Local LLM	Ollama serving Qwen3 8B
NLI model	roberta-large-mnli
OCR	Tesseract (pytesseract)
Document parsing	PyMuPDF (PDF), python-docx (DOCX)

5.3 Procedural Flow

The overall processing flow is divided into three main workflows, as shown in Figure 4. These are knowledge-base creation, verified question answering, and document fact-checking. Although each workflow begins differently, they share the same final stages for classification, result display, and data storage.

The Sentence Transformer model is responsible for creating embeddings from the processed text and user queries. Ollama is used specifically to run the local language model responsible for answer generation.

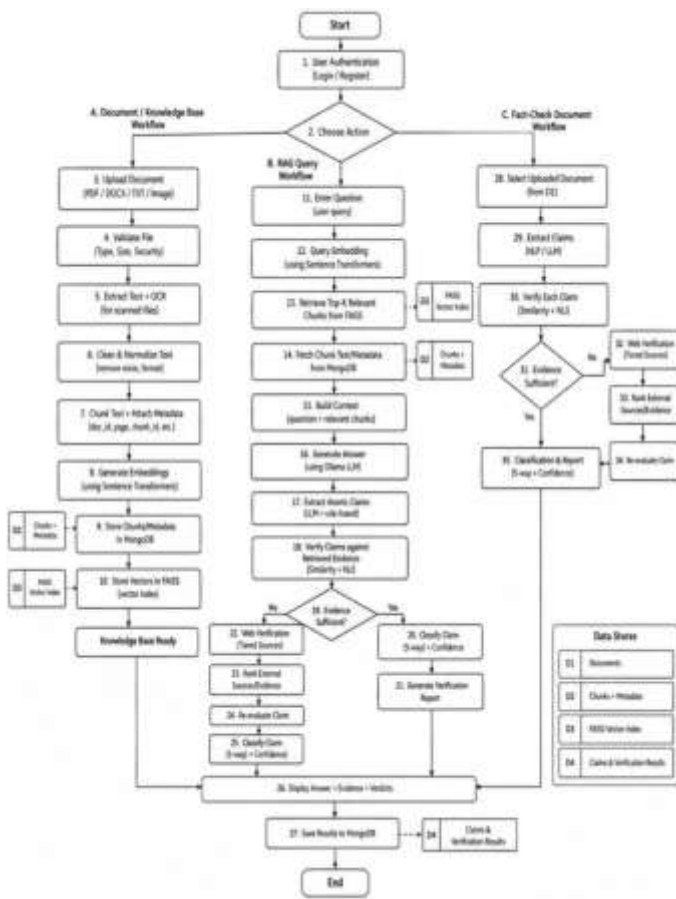


Figure 2. Procedural flow of the three main system workflows.

6. EVALUATION METHODOLOGY

The proposed framework is evaluated using an ablation-based approach to understand the contribution of each verification component. Instead of testing only the complete system, different configurations are compared. Five setups are considered: a basic RAG pipeline without verification, similarity-only verification, NLI-only verification, the combined similarity and NLI approach, and the combined approach with the tiered web verification described in Section 4.5.

6.1 Metrics

The following measures are used to evaluate the system:

- **Precision, recall, and F1-score:** These are calculated separately for each classification category to show how well the system handles different types of claims.
- **False positive and false negative rates:** These are measured for the **Contradicted/Likely Hallucination** category because incorrect decisions in this category are particularly important.
- **Recall@K and Mean Reciprocal Rank (MRR):** These metrics evaluate the retrieval stage by checking whether relevant evidence is retrieved and how highly it is ranked.

- **Average verification latency:** The processing time is measured per claim and for the complete query to understand the cost of running the locally hosted models.

6.2 Dataset Construction

The evaluation mainly uses a custom labelled dataset instead of depending only on existing benchmarks. This is partly motivated by reported annotation issues in datasets such as HaluEval and RAGTruth. The custom dataset is created from a small document collection and includes questions that are directly answerable from the corpus, partially answerable, and outside the available information.

Each extracted claim is manually assigned one of the five classification categories used by the proposed framework. A separate set of intentionally false claims is also created by modifying correct answers or generating incorrect statements. This set is used to test the system's ability to identify contradictions and likely hallucinations.

A filtered subset of an existing benchmark may also be included as a secondary test set to examine how well the system performs on data outside the custom corpus.

6.3 Confidence Calibration

The evaluation also checks whether the confidence scores produced by the system reflect its actual performance. For example, predictions assigned around 80% confidence should be correct at approximately the same rate.

The confidence formula described in Section 4.6 is therefore treated as an initial assumption that needs to be tested. This analysis helps determine whether the system produces reasonable confidence values or tends to be overly confident or insufficiently confident in its verification results.

7. LIMITATIONS AND FUTURE WORK

7.1 Limitations

Although the proposed framework provides several levels of verification, it still has some limitations.

- **Dependence on models:** The overall performance depends on the embedding model, NLI model, and LLM used by the system. Errors from any of these components can affect the final verification result. Therefore, the framework is intended to reduce hallucinations rather than completely remove them.
- **Difficult factual cases:** Some incorrect claims can be very similar to the available evidence, making them difficult to detect using embeddings alone. NLI may also have difficulty identifying small changes in numbers, dates, or named entities.

- **Limited evidence coverage:** A claim cannot be reliably verified when relevant information is not available in the knowledge base or the selected web sources. The manually maintained source reliability list may also fail to include useful websites or newly available information.
- **Type of claims:** The current system mainly focuses on factual statements that can be checked against textual evidence. It does not specifically handle mathematical calculations, long reasoning processes, or claims that require information to be combined from several unrelated sources.
- **System scale:** The current evaluation is carried out on a relatively small research dataset and uses a single user role. As a result, the findings may not directly represent the performance of the system in large-scale or multi-user environments.
- **Hardware requirements:** Since the models are hosted locally, system speed depends on the available hardware. GPU capability can have a significant effect on response and verification time.
- **Evaluation dataset:** The custom dataset is relatively small and is labeled by a limited number of annotators. A larger dataset with more independent annotations would provide stronger evidence about how well the framework generalizes.

7.2 Future Work

Several improvements can be explored in future versions of the framework:

- Test stronger embedding and NLI models, along with cross-encoder re-ranking, to determine whether they improve verification accuracy.
- Add consistency checking by comparing the results of multiple independent generations of the same answer.
- Investigate the use of internal model information as an additional verification signal, while considering the extra model-specific tuning that may be required.
- Use a stronger LLM as an additional judge when semantic similarity and NLI produce conflicting results.
- Develop a more dynamic method for estimating source reliability instead of depending only on a manually maintained domain list.
- Extend the framework to handle multi-step reasoning and claims that require evidence from multiple documents.
- Add role-based access, support for multiple users, and larger document collections to evaluate the

framework in more realistic deployment environments.

8. CONCLUSION

This paper presented a claim-level framework for detecting and verifying hallucinations in RAG systems. The main idea is to check individual factual claims rather than treating the entire generated response as one unit. The framework combines semantic similarity and Natural Language Inference (NLI) to improve verification and uses different outcome categories to distinguish between supported claims, contradictions, partial support, and cases where sufficient evidence is not available. When the internal knowledge base cannot provide enough information, the system can also use external web sources based on their reliability.

The proposed system is designed using locally hosted and open-source technologies, making it suitable for reproducible research without relying on paid APIs. An ablation-based evaluation plan has also been defined to compare the individual verification components with the complete framework. Confidence calibration is included to check whether the confidence values produced by the system match its actual results.

At the current stage, the system is still under implementation and the planned experiments have not yet been completed. Therefore, this work does not claim measured improvements in accuracy or hallucination detection. The next step is to complete the implementation, conduct the planned evaluation, and analyze both the strengths and limitations of the proposed approach based on experimental results.

ACKNOWLEDGMENT

We thank Ms. Karishma Jain, Department of Computer Science, Patkar-Varde College, for guidance on the project that this paper condenses, and the college for laboratory access. We also thank the Research Methodology faculty for the chapter structure that separated problem, method, and deferred work.

REFERENCES

- [1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020.
- [2] C. Niu et al., "RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models," in Proc. ACL, 2024.
- [3] J. Li, X. Cheng, W. X. Zhao, J. Nie, and J. Wen, "HaluEval: A large-scale hallucination evaluation benchmark for large language models," in Proc. EMNLP, 2023.
- [4] MultiHal and related 2025 studies on semantic similarity, NLI, and embedding limits for RAG hallucination detection.
- [5] "Rethinking evaluation for LLM hallucination detection," preprint, 2026.

- [6] Cleanlab, “Notes on annotation quality in public hallucination benchmarks,” 2024.
- [7] A. Williams, N. Nangia, and S. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in Proc. NAACL-HLT, 2018.
- [8] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in Proc. EMNLP-IJCNLP, 2019.
- [9] Á. Kovács and G. Recski, “LettuceDetect: A hallucination detection framework for RAG applications,” 2025.
- [10] Ragas project documentation.
- [11] TruLens and DeepEval project documentation.
- [12] Sun et al., “ReDeEP: Detecting hallucination in retrieval-augmented generation via external and parametric knowledge scores,” in Proc. ICLR, 2025.
- [13] P. Manakul, A. Liusie, and M. Gales, “SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models,” in Proc. EMNLP, 2023.
- [14] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” arXiv:1907.11692, 2019.
- [15] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans. Big Data, 2019.