# MetaRAG: Metamorphic Testing for Hallucination Detection in RAG Systems*

Channdeth Sok[1,2,*], David Luz[1] and Yacine Haddam[1]

[1]Forvia Paris Tech Center, GIT, Immeuble Lumière, 40 avenue des Terroirs de France, 75012 Paris, France
[2]ENSAE Paris, Institut Polytechnique de Paris

**Abstract**

Large Language Models (LLMs) are increasingly deployed in enterprise applications, yet their reliability remains limited by *hallucinations*, i.e., confident but factually incorrect information. Existing detection approaches, such as SelfCheckGPT and MetaQA, primarily target standalone LLMs and do not address the unique challenges of Retrieval-Augmented Generation (RAG) systems, where responses must be consistent with retrieved evidence. We therefore present **MetaRAG**, a **meta**morphic testing framework for hallucination detection in **R**etrieval-**A**ugmented **G**eneration (RAG) systems. MetaRAG operates in a real-time, unsupervised, black-box setting, requiring neither ground-truth references nor access to model internals, making it suitable for proprietary and enterprise deployments. The framework proceeds in four stages: (1) decompose answers into atomic factoids, (2) generate controlled mutations of each factoid using synonym and antonym substitutions, (3) verify each variant against the retrieved context (synonyms are expected to be entailed and antonyms contradicted), and (4) aggregate penalties for inconsistencies into a response-level hallucination score. MetaRAG further localizes unsupported claims at the span level, enabling transparent visualization of potentially hallucinated segments and supporting configurable safeguards in sensitive use cases. Experiments on a proprietary enterprise dataset demonstrate the effectiveness of **MetaRAG** for detecting hallucinations and enabling trustworthy deployment of RAG-based conversational agents. We also outline a topic-based deployment design that translates MetaRAG's span-level scores into identity-aware safeguards; this design is discussed but not evaluated in our experiments.

**Keywords**

Large Language Models, Retrieval-Augmented Generation, Hallucination Detection, Metamorphic Testing, Trustworthy AI

## 1. Introduction

Large Language Models (LLMs) such as GPT-4 and Llama-3 are transforming enterprise applications in healthcare, law, and customer service [1, 2, 3]. They power chatbots and virtual assistants that interact in natural language, offering unprecedented convenience and efficiency [4]. However, as these systems move into production, a persistent challenge emerges: *hallucinations*, i.e., responses that are fluent and convincing but factually incorrect or unsupported by evidence [5, 6].

In domains such as healthcare, law, and finance, hallucinations are not merely a nuisance but a critical barrier to reliable adoption, raising concerns about user trust, regulatory compliance, and business risk [7]. Moreover, hallucinations are not uniformly risky: the same unsupported claim can differentially affect specific populations. In healthcare (e.g., pregnancy/trimester-specific contraindications), migration and asylum (e.g., protections for LGBTQ+ refugees), or labor rights (e.g., eligibility by status), ungrounded spans can cause disproportionate harm. Rather than treating users as homogeneous, hallucination detection methods should make such spans *reviewable* at the factoid level so downstream systems can apply identity-aware policies (e.g., stricter thresholds, forced citations, or escalation to a human) when the topic indicates elevated risk. This perspective connects hallucination detection to identity-aware deployment, where span-level evidence enables topic-conditioned safeguards that reduce disproportionate risk.

Ji et al. [6] categorize hallucinations into two types:

*Identity-Aware AI workshop at 28th European Conference on Artificial Intelligence, October 25, 2025, Bologna, Italy*
*Corresponding author.
channdeth.sok@forvia.com (C. Sok)
0009-0008-4547-5946 (C. Sok)

- **Intrinsic hallucination**: fabricated or contradictory information relative to the model's internal knowledge.
- **Extrinsic hallucination**: generated information that conflicts with, misrepresents, or disregards externally provided context or retrieved documents.

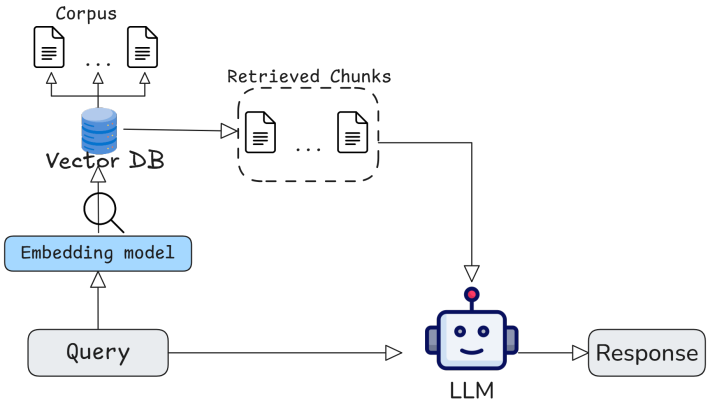



**Figure 1:** Standard Retrieval-Augmented Generation (RAG) workflow. A user query is encoded into a vector representation using an embedding model and queried against a vector database constructed from a document corpus. The most relevant document chunks are retrieved and appended to the original query, which is then provided as input to a large language model (LLM) to generate the final response.

Retrieval-Augmented Generation (RAG) [8] aims to mitigate hallucinations by grounding model outputs in retrieved, up-to-date documents, as illustrated in Figure 1. By injecting retrieved text from reliable external sources and proprietary documents, into the prompt, RAG improves factuality and domain relevance. While effective against intrinsic hallucinations, RAG remains susceptible to *extrinsic* hallucinations, especially when retrieved evidence is ignored, misinterpreted, or insufficient [9].

Detecting hallucinations is particularly challenging in real-world settings, where RAG-based chatbots must respond to queries about unseen, proprietary, or confidential content where gold-standard references are typically unavailable [10]. Many existing hallucination detection methods rely on gold-standard reference answers [5, 11], annotated datasets [12], or access to model internals such as hidden states or token log-probabilities [13, 14]. However, in enterprise settings, such internals are often inaccessible: many state-of-the-art LLMs (e.g., GPT-4, Claude) are proprietary and only accessible via APIs that expose the final output text but not intermediate computations, limiting the feasibility of these methods in practice [10].

To address these challenges, we introduce **MetaRAG**: a metamorphic testing framework for detecting hallucinations in RAG-based conversational agents. **MetaRAG is a zero-resource, black-box setting** that decomposes answers into atomic factoids, applies controlled mutations (e.g., synonym and antonym substitutions), and verifies each mutated factoid against the retrieved context. Synonyms are expected to be *entailed*, while antonyms are expected to be *contradicted*. Hallucinations are flagged when outputs violate these well-defined metamorphic relations (MRs). Unlike prior approaches, MetaRAG does not require ground-truth labels, annotated corpora, or access to model internals, making it suitable for deployment in proprietary settings.

We evaluate MetaRAG on a proprietary corpus, thus unseen during model training. Our results show that MetaRAG reliably detects hallucinations, providing actionable insights for enhancing chatbot reliability and trustworthiness. These results establish MetaRAG as a practical tool for reliable deployment, and its span-level detection opens the door to identity-aware safeguards.

**Our contributions include:**

- We introduce **MetaRAG**, a reference-free, black-box setting, metamorphic testing framework for hallucination detection in RAG systems. It decomposes answers into factoids, applies linguistic

transformations (synonym and antonym), and verifies them against retrieved context to produce a hallucination score.

- We implement a prototype and evaluate MetaRAG on a proprietary dataset, demonstrating its effectiveness in detecting hallucinations that occur when segments of generated responses diverge from the retrieved context.
- We analyze the performance–latency/cost trade-offs of MetaRAG and provide a consistency analysis to guide future research and practical deployment.
- We outline identity-aware safeguards (topic-aware thresholds, forced citation, escalation) that can consume MetaRAG's scores; these safeguards are a deployment design and are not part of our empirical evaluation.

## 2. Related Works

### 2.1. Definitions of Hallucination

The term *hallucination* has been used with varying scope across natural language generation tasks. Some studies emphasize ***factuality***, describing hallucinations as outputs that contradict established facts, i.e., inconsistencies with world knowledge or external ground truth [15, 16]. Others highlight ***faithfulness***, where hallucinations occur when generated responses deviate from the user instruction or a reference text, often producing plausible but ungrounded statements particularly in source-conditioned tasks such as summarization or question answering [17]. Beyond these two dimensions, researchers also note cases of incoherent or nonsensical text that cannot be clearly attributed to factuality or faithfulness criteria [6, 5].

Alternative terms have also been introduced. ***Confabulation*** draws on psychology to describe fluent but fabricated content arising from model priors [18], while ***fabrication*** is preferred by some to avoid anthropomorphic connotations [19, 20]. More recently, Chakraborty et al. [21] propose a flexible definition tailored to deployment settings, defining a hallucination as *a generated output that conflicts with constraints or deviates from desired behavior in actual deployment, while remaining syntactically plausible under the circumstance.*

### 2.2. Hallucination Detection in LLMs

Building on these definitions, hallucinations have been recognized as a major challenge in text generation. Early work in machine translation and abstractive summarization described them as outputs that are not grounded in the input source [5, 11, 22], motivating the development of evaluation metrics and detection methods for faithfulness and factual consistency across natural language generation tasks.

More recent *reference-free* (unsupervised or zero-reference) methods aim to detect hallucinations without gold-standard labels by analyzing the model's own outputs. A prominent method is **SelfCheck-GPT** [23], a zero-resource, black-box approach that queries the LLM multiple times with the same prompt and measures semantic consistency across responses. The intuition is that hallucinated content often leads to instability under stochastic re-generation; true facts remain stable, while fabricated ones diverge. Manakul et al. show that SelfCheckGPT achieves strong performance in sentence-level hallucination detection compared to gray-box methods, and emphasize that it requires no external database or access to model internals [23]. However, SelfCheckGPT may struggle when deterministic decoding or high model confidence leads to repeating the same incorrect output.

### 2.3. Metamorphic Testing

Metamorphic Testing (MT) [24] was originally proposed in software engineering to address the *oracle problem* in which the correct output is unknown. MT relies on *metamorphic relations* (MRs): transformations of the input with predictable effects on outputs, enabling error detection without access to

ground truth [25]. In machine learning, MT has been applied to validate models in computer vision [26] (e.g., rotating an image should not change its predicted class) and NLP [27]

In hallucination detection for LLMs, **MetaQA** [28] leverages MRs by generating paraphrased or antonym-based question variants and verifying whether answers satisfy expected semantic or logical constraints. Relying purely on prompt mutations and consistency checks, MetaQA achieves higher precision and recall than SelfCheckGPT on open-domain QA.

Researchers have also adapted MT for more complex conversational and reasoning settings. **MORTAR** [29] applies dialogue-level perturbations and knowledge-graph-based inference to multi-turn systems, detecting up to four times more unique bugs than single-turn MT. **Drowzee** [30] uses logic programming to construct temporal and logical rules from Wikipedia, generating fact-conflicting test cases and revealing rates of 24.7% to 59.8% across six LLMs in nine domains [28].

These works highlight the promise of MT for hallucination detection, but they primarily target open-book QA or multi-turn dialogue, often over short, single-sentence outputs. Prior studies have not addressed hallucination detection in *retrieval-augmented generation* (RAG) scenarios over proprietary corpora, a setting in which ground-truth references are unavailable and model internals are inaccessible. **MetaRAG** builds on MT by decomposing answers into factoids and designing MRs tailored to factual consistency against retrieved evidence in a zero-resource, black-box setting.

# 3. MetaRAG: Methodology

## 3.1. Overview

Building on the metamorphic testing (MT) methodology to detect hallucinations in LLMs introduced by MetaQA [28], **MetaRAG** advances this approach to detect hallucinations in retrieval-augmented generation (RAG) settings by introducing a context-based verification stage. A metamorphic testing layer operates on top of the standard RAG pipeline to automatically detect hallucinated responses. Figure 2 outlines the workflow.

Given a user query $Q$, the system retrieves the top-$k$ most relevant chunks from a database, forming the context $C = \{c_1, c_2, \ldots, c_k\}$. The LLM generates an initial answer $A$ using $(Q, C)$ as input. MetaRAG then decomposes $A$ into factoids, applies controlled metamorphic transformations to produce variants (synonym and antonym), verifies each variant against $C$, and aggregates the results into a hallucination score (Algorithm 1).

## 3.2. Step 1: Factoid decomposition

Given an answer $A$, we first decompose it into a set of *factoids*, defined as atomic, independently verifiable facts, denoted by $\mathscr{F} = \{F_1, \ldots, F_M\}$. Each factoid $F_j$ corresponds to a single factual statement that cannot be further divided without losing meaning, such as a subject-predicate-object triple or a scoped numerical or temporal claim. Representing an answer $A$ at the factoid level enables fine-grained verification in subsequent steps, allowing localized hallucinations to be marked inside longer answers.

We obtain $\mathscr{F}$ using an LLM-based extractor with a fixed prompt that enforces one proposition per line, prohibits paraphrasing or inference beyond $A$, and co-reference resolution. The full prompt template is provided in the supplementary material.

## 3.3. Step 2: Mutation Generation

Each factoid (hereafter, *fact*) from Step 1, MetaRAG applies metamorphic mutations to generate perturbed variants of the original claim. This step is grounded in the principle of metamorphic testing, where controlled semantic transformations are used to probe model consistency and expose hallucinations [28].

Formally, for each factoid $F_i \in \{F_1, \ldots, F_M\}$, we construct variants using two relations:

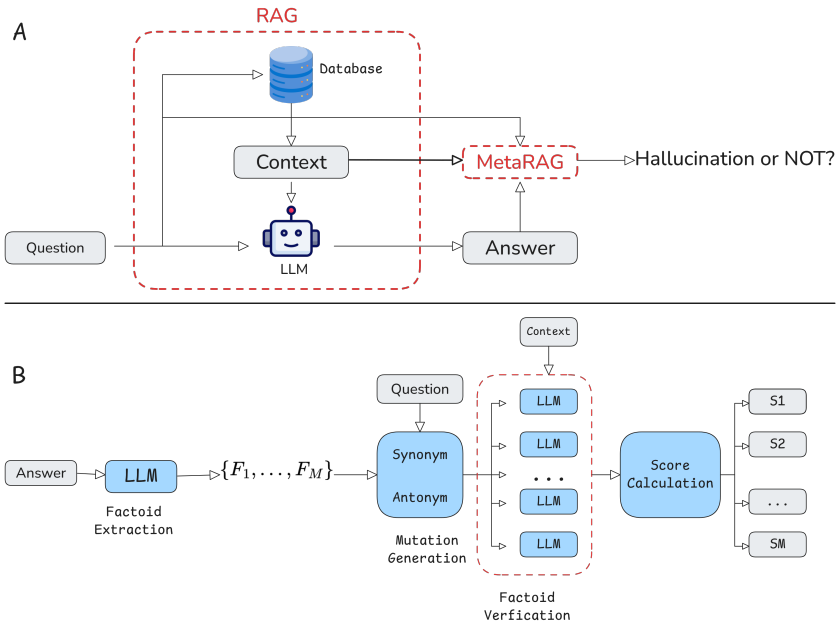



**Figure 2:** Overview of the **MetaRAG** workflow. **(A)** Integration of MetaRAG into a standard RAG pipeline: given a user question, the RAG retrieves context and generates an answer, which is then passed to MetaRAG for hallucination detection. **(B)** Internal MetaRAG pipeline: the answer is decomposed into atomic factoids, each factoid is mutated through synonym and antonym substitutions, and verified against the retrieved context using entailment/contradiction checks. Penalties are assigned to inconsistencies, and scores are aggregated into a response-level hallucination score.

- **Synonym Mutation**: This relation substitutes key terms in $F_i$ with appropriate synonyms, yielding paraphrased factoids $F_{i,j}^{\text{syn}}$ that preserve the original semantic meaning. These assess the model's ability to recognize reworded yet factually equivalent statements.
- **Antonym Mutation**: This relation replaces key terms in $F_i$ with antonyms or negations, producing factoids $F_{i,j}^{\text{ant}}$ that are semantically opposed to the original. These serve as adversarial tests to ensure the model does not support clearly contradictory information.

Let $N$ denote the number of mutations generated by *each* relation. The mutation set for $F_i$ is therefore

$$\mathscr{F}_i = \{ F_{i,1}^{\text{syn}}, \ldots, F_{i,N}^{\text{syn}}, \; F_{i,1}^{\text{ant}}, \ldots, F_{i,N}^{\text{ant}} \}.$$

By construction, if $F_i$ is correct and supported by the retrieved context $C$, then $F_{i,\cdot}^{\text{syn}}$ should be *entailed* by $C$, whereas $F_{i,\cdot}^{\text{ant}}$ should be *contradicted* by $C$.

Mutations are generated by prompting an LLM with templates that explicitly instruct synonymous or contradictory outputs while preserving atomicity and relevance; the exact prompt templates appear in the supplementary material.

### 3.4. Step 3: Factoid Verification

Each mutated factoid $F_{i,j}^{\text{syn}}$ and $F_{i,j}^{\text{ant}}$ is then verified by LLMs conditioning on the context $C$ (treated as ground truth). The LLM returns one of three decisions: Yes (entailed by $C$), No (contradicted by $C$), or Not sure (insufficient evidence). We then assign a penalty score $p \in \{0, 0.5, 1\}$ based on the decision and the mutation type:

This penalty assignment quantifies semantic (in)consistency at the variant level: correct entailment for synonyms and correct contradiction for antonyms yield zero penalty, while the opposite yields maximal penalty. In Step 4, we aggregate these penalties over all variants of each $F_i$ to compute a fact-level hallucination score.

**Algorithm 1** MetaRAG Hallucination Detection

**Input:** Generated answer $A$, query $Q$, context $C$, number of mutations $N$, threshold $\tau$
**Output:** Hallucination score $H(Q, A, C)$, factoid scores $\{S_i\}$
**Factoid Extraction:**
$\mathscr{F} \leftarrow$ FactoidsDecomposition$(A)$
**for** each factoid $F_i$ in $\mathscr{F}$ **do**
  **Mutation Generation:**
  *Synonyms* $\leftarrow$GenerateSynonymMutations$(F_i, Q, N)$
  *Antonyms* $\leftarrow$GenerateAntonymMutations$(F_i, Q, N)$
  **Verification:**
  **for** each $F^{syn}$ in *Synonyms* **do**
    *result* $\leftarrow$ VerifyWithLLM$(F^{syn}, C)$
    *SynResults*.append(*result*)
  **end for**
  **for** each $F^{ant}$ in *Antonyms* **do**
    *result* $\leftarrow$ VerifyWithLLM$(F^{ant}, C)$
    *AntResults*.append(*result*)
  **end for**
  **Scoring:**
  *SynScores* $\leftarrow$ [MapSynonymScore$(r)$ for $r$ in *SynResults*]
  *AntScores* $\leftarrow$ [MapAntonymScore$(r)$ for $r$ in *AntResults*]
  $S_i \leftarrow$ Mean(*SynScores* $\cup$ *AntScores*)
**end for**
**Aggregation:**
$H(Q, A, C) \leftarrow \max_i S_i$
Return $H(Q, A, C), \{S_i\}$

**Table 1**
Penalty scheme for metamorphic verification (lower is better).

| | Penalty $p$ | |
|---|---|---|
| Decision | Synonym | Antonym |
| Yes | 0.0 | 1.0 |
| Not sure | 0.5 | 0.5 |
| No | 1.0 | 0.0 |

## 3.5. Step 4: Score Calculation

To quantify hallucination risk, we calculate a hallucination score, $S_i$, for each factoid $F_i$. This yields a granular diagnostic that pinpoints which claims are potentially unreliable. The score for each factoid $i$ is defined as the average penalty across the $2N$ metamorphic transformations (synonym and antonym) of $F_i$:

$$S_i = \frac{1}{2N}\left(\sum_{j=1}^{N} p_{i,j}^{\text{syn}} + \sum_{j=1}^{N} p_{i,j}^{\text{ant}}\right), \tag{1}$$

where $p_{i,j}^{\text{syn}}$ and $p_{i,j}^{\text{ant}}$ are the penalties assigned in Step 3 to the $j$-th synonym and antonym variants of $F_i$, respectively. By construction, $S_i \in [0, 1]$: $S_i = 0$ indicates a perfectly consistent, well-grounded factoid, thus no hallucination, while $S_i = 1$ indicates a highly probable hallucination.

**Response Hallucination score:** Instead of a simple average, the hallucination score for the entire response $A$ is defined as the maximum score found among all the individual factoids. This metric ensures that a single, severe hallucination in any part of the response will result in a high overall score, accurately reflecting the unreliability of the entire answer.

$$H(Q, A, C) = \max_{1 \le i \le M} S_i, \tag{2}$$

where $M$ is the number of decomposed factoids. A response can be flagged as containing hallucination if $H(Q, A, C)$ exceeds a predefined confidence threshold $\tau \in [0, 1]$ (e.g. 0.5).

### 3.6. Identity-Aware Safeguards for Deployment

While MetaRAG is a general-purpose hallucination detector, its factoid-level scores can be directly integrated into *identity-aware deployment policies*. Importantly, no protected attributes are inferred or stored; instead, only the *topic of the query or retrieved context* (e.g., pregnancy, refugee rights, labor eligibility) is used as a deployment signal. *Scope.* The safeguards described here represent a deployment design that consumes MetaRAG's scores; they are not part of the empirical evaluation reported in Section 4.

Each factoid receives a score $S_i \in [0, 1]$, where $S_i = 0$ indicates full consistency with the retrieved context and $S_i = 1$ indicates strong evidence of hallucination. The overall response score $H(Q, A, C)$ thus represents the risk level of the most unreliable claim: higher values correspond to higher hallucination risk.

These scores could enable deployment-time safeguards through the following hooks:

1. **Topic detection.** A lightweight topic classifier or rule-based tagger can assign coarse domain labels (e.g., healthcare, migration, labor) to the query or retrieved context.

2. **Topic-aware thresholds.** A response is flagged if $H(Q, A, C) \ge \tau$. Thresholds can be adapted by domain, e.g., $\tau_{\text{general}} = 0.5$ for generic queries, and a stricter $\tau_{\text{identity}} = 0.3$ for sensitive domains.

3. **Span highlighting and forced citation.** For flagged responses, MetaRAG highlights unsupported spans and enforces inline citations to retrieved evidence, to improve transparency and calibrate user trust.

4. **Escalation.** If hallucinations persist above threshold in identity-sensitive domains, the system may abstain, regenerate with a stricter prompt, or escalate to human review.

5. **Auditing.** Logs of flagged spans, hallucination scores, and topic labels can be maintained for post-hoc fairness, compliance, and safety audits.

In this way, higher hallucination scores are systematically translated into stronger protective actions, with more conservative safeguards applied whenever queries touch on identity-sensitive contexts.

## 4. Experiments

We conducted experiments to evaluate **MetaRAG** on its ability to *detect* hallucinations in retrieval-augmented generation (RAG). The evaluation simulates a realistic enterprise deployment setting, in which a chatbot serves responses generated from internal documentation. Our focus is on the detection stage, that is, identifying when an answer contains unsupported (hallucination) or fabricated information. Prevention and mitigation are important but they are outside the scope of this work.

### 4.1. Dataset

The evaluation dataset is a proprietary collection of **23 internal enterprise documents**, including policy manuals, procedural guidelines, and analytical reports, none of which were seen during LLM training. Each document was segmented into chunks of a few hundred tokens, and retrieval used cosine similarity over `text-embedding-3-large`, with the top-$k = 3$ chunks appended to each query.

We then collected a set of user queries and corresponding chatbot answers. Each response was labeled by human annotators as either hallucinated or not, using the retrieved context as the reference. The final evaluation set contains **67 responses**, of which **36** are labeled as *not hallucinated* and **31** as *hallucinated*.

To preserve confidentiality, we do not release the full annotated dataset. However, the complete annotation guidelines are included in the supplementary material.

### 4.2. Evaluation Protocol

MetaRAG produces **fine-grained, factoid-level hallucination scores**, whereas the available ground truth labels are at the **response level**. To align with these existing labels, we evaluate MetaRAG as a binary classifier by thresholding the hallucination score $H(Q, A, C)$ at $\tau = 0.5$. We report standard classification metrics: Precision, Recall, F1 score and accuracy. Latency is also recorded to assess feasibility for real-time deployment.

#### 4.2.1. Case Studies in Identity-Sensitive Domains

Beyond quantitative evaluation, we also provide qualitative illustrations of MetaRAG in identity-sensitive scenarios. To illustrate how MetaRAG's span-level scores can enable identity-aware safeguards without inferring protected attributes, we present two stylized examples. These are not part of the quantitative evaluation in Section 4, but highlight potential deployment scenarios.

**Healthcare (pregnancy).** A user asks: "Can pregnant women take ibuprofen for back pain?" The model answers: "Yes, ibuprofen is **safe throughout pregnancy**." However, the retrieved context specifies that ibuprofen is contraindicated in the third trimester. MetaRAG flags the span **"safe throughout pregnancy"** with a high factoid score ($S_i = 0.92$), yielding a response-level score $H = 0.92$. Under the policy hooks described in Section 3.6, the topic tag *pregnancy* triggers a stricter threshold ($\tau_{\text{identity}} = 0.3$, lower than the general case), span highlighting, a forced citation requirement, and possible escalation to human review.

**Migration (refugee rights).** A user asks: "Do LGBTQ+ refugees **automatically receive protection** in country X?" The model claims that such protections are **automatic**, but the retrieved legal text provides no evidence of this. MetaRAG flags the unsupported claim **"automatically receive protection"** with a moderate score ($S_i = 0.5$), yielding a response-level score $H = 0.5$. Although this score would sit at the decision boundary under a general threshold ($\tau_{\text{general}} = 0.5$), the stricter identity-aware threshold ($\tau_{\text{identity}} = 0.3$) ensures it is flagged for this case. Under the policy hooks, the topic tag *asylum/refugee* enforces citation and may escalate the response to a human reviewer. In a chatbot deployment, the system would abstain from returning the unsupported answer and instead notify the user that expert verification is required.

These qualitative vignettes complement our quantitative evaluation by showing how MetaRAG's flagged spans can be turned into concrete safeguards in identity-sensitive deployments.

## 5. Ablation Study

To understand the contribution of individual design choices, we perform a set of ablation experiments using the private dataset.

## 5.1. Ablation Study Design

We evaluate 26 configurations of MetaRAG, each defined by a combination of:

- Number of variants per relation $N \in \{2, 5\}$
- Factoid-decomposition model: *gpt-4.1 or gpt-4.1-mini* from OpenAI
- Temperature for mutation generation: $T \in \{0.0, 0.7\}$
- Mutation–generation model: *gpt-4.1 or gpt-4.1-mini*
- Verifier model: *gpt-4.1-mini*, *gpt-4.1*, or the *multi* ensemble (gpt-4.1-nano, gpt-4.1-mini, gpt-4.1, Claude Sonnet 4)

Since the evaluation task is binary classification, we report **Precision**, **Recall**, **F1 score**, and **Accuracy**, along with **latency** (lower is better).

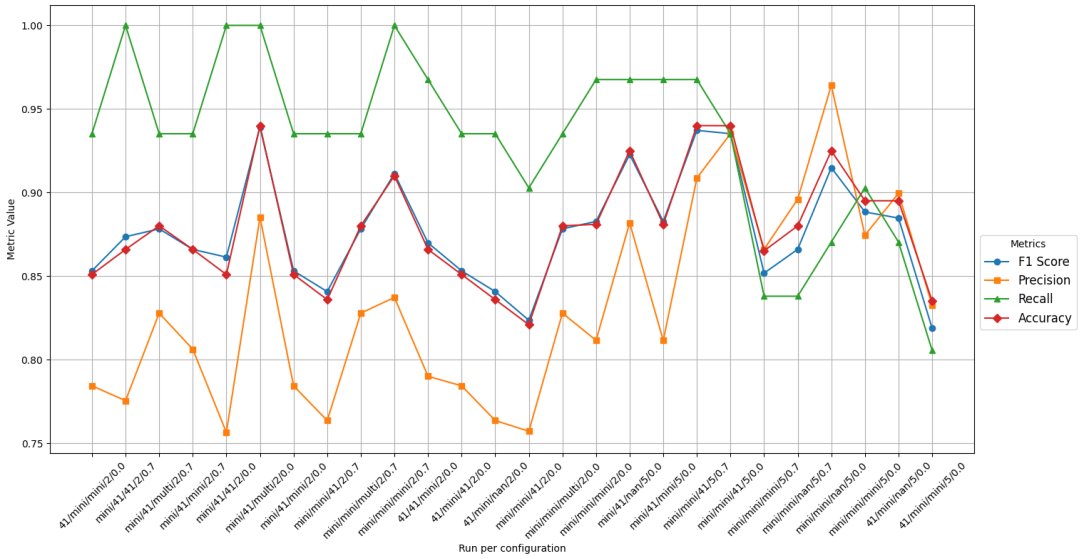



**Figure 3:** Evaluation metrics for all 26 **MetaRAG** configurations.

## 5.2. Results

To provide a comprehensive view of performance trade-offs, we report the **Top-4 configurations separately** for each of three primary metrics: F1 score, Precision, and Recall (Table 2). The configuration notation follows the format:

`Decomposition Model / Generation Model / Verifier /` $N$ `/ Temperature`.

For example, `mini/41/multi/2/0` indicates that the factoid decomposition model is "mini", the variant generation model is "41", the verifier is "multi", there are $N = 2$ variants per relation, and the temperature is 0.0.

Several configurations appear in more than one top-4 list, reflecting balanced performance across metrics. For instance, ID 5 (`mini/41/multi/2/0`) ranks first in both F1 score and Recall, while maintaining competitive Precision.

The most promising configurations are further examined in Section 5.3 to verify stability under multiple seeds.

**Table 2**
Validation leaderboards for 26 **MetaRAG** configurations, showing the top–4 for each metric.

| ID | Config. | F1 | Prec. | Rec. | Acc. |
|---|---|---|---|---|---|
| **Top-4 by F1** | | | | | |
| 5 | mini/41/multi/2/0 | 0.939 | 1.000 | 0.885 | 0.940 |
| 18 | mini/mini/41/5/0.7 | 0.937 | 0.909 | 0.968 | 0.940 |
| 19 | mini/mini/41/5/0 | 0.935 | 0.935 | 0.935 | 0.940 |
| 16 | mini/41/multi/5/0 | 0.923 | 0.882 | 0.968 | 0.925 |
| **Top-4 by Precision** | | | | | |
| 22 | mini/mini/multi/5/0 | 0.915 | 0.964 | 0.870 | 0.925 |
| 19 | mini/mini/41/5/0 | 0.935 | 0.935 | 0.935 | 0.940 |
| 18 | mini/mini/41/5/0.7 | 0.937 | 0.909 | 0.968 | 0.940 |
| 24 | 41/mini/multi/5/0 | 0.885 | 0.900 | 0.870 | 0.895 |
| **Top-4 by Recall** | | | | | |
| 1 | mini/41/41/2/0.7 | 0.874 | 0.776 | 1.000 | 0.866 |
| 5 | mini/41/multi/2/0 | 0.939 | 0.885 | 1.000 | 0.940 |
| 9 | mini/mini/mini/2/0.7 | 0.911 | 0.837 | 1.000 | 0.910 |
| 4 | mini/41/41/2/0 | 0.861 | 0.757 | 1.000 | 0.851 |

**Config legend**: `Decomp/GenModel/Verifier/`$N$`/Temp`.

## 5.3. Consistency Check

To verify the robustness of our results, each top configuration (selected based on F1 score, Precision, Recall, and token usage) is rerun under identical conditions using five different random seeds. This procedure serves three purposes:

- To ensure that high performance is not attributable to random initialization or favorable seeds.
- To quantify variability across runs with the same configuration by reporting the standard deviation for each metric.
- To assess stability using the coefficient of variation (CV) defined as the ratio of the standard deviation to the mean ($\text{CV} = \sigma/\mu$), where lower values indicate greater consistency.

**Table 3**
Run-to-run consistency for top configurations (mean ± standard deviation over 5 seeds) and coefficient of variation (CV) for F1.

| ID | F1 | Precision | Recall | CV (F1) |
|---|---|---|---|---|
| 16 | 0.9397 ± 0.0123 | 0.9198 ± 0.0243 | 0.9610 ± 0.0144 | 1.31% |
| 18 | 0.9356 ± 0.0089 | 0.9322 ± 0.0439 | 0.9413 ± 0.0278 | 0.95% |
| 19 | 0.9347 ± 0.0305 | 0.9410 ± 0.0357 | 0.9286 ± 0.0272 | 3.26% |
| 5 | 0.9108 ± 0.0346 | 0.8463 ± 0.0503 | 0.9869 ± 0.0179 | 3.80% |

Across all metrics, the top configurations demonstrate strong reproducibility, with the majority exhibiting a CV below 2%. In particular, configurations 18 and 16 achieve both high F1 scores and low variability, indicating that they are not only accurate but also stable across repeated trials.

## 5.4. Pareto Front Analysis

Following the consistency check (Section 5.3), we restrict the Pareto front analysis to the four most stable top-performing configurations selected by F1 score. We analyze the trade-off between hallucination detection performance and efficiency using Pareto frontiers. A configuration is **Pareto-optimal** if

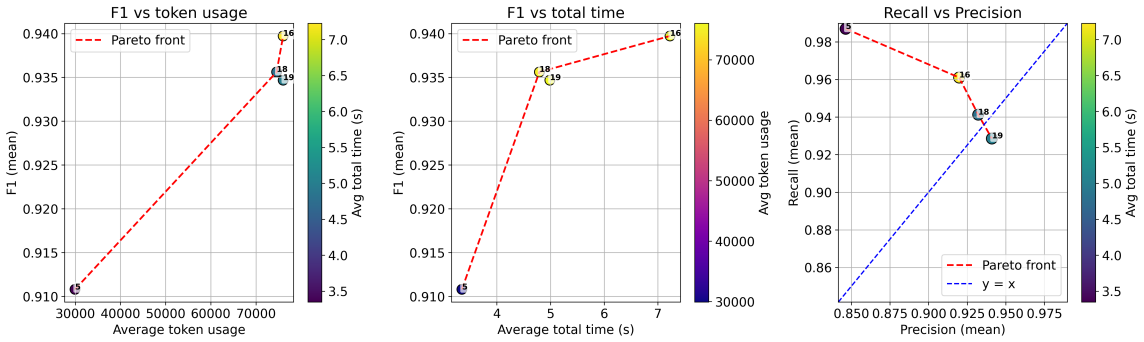



**Figure 4:** Pareto front analysis for hallucination detection performance. Each point represents a MetaRAG configuration; Pareto-optimal points (non-dominated) are highlighted. Subplots show: (Left) F1 vs. average token usage, (Center) F1 vs. average total execution time, (Right) Precision vs. Recall. Pareto-optimal points represent configurations with no strictly better alternative in both accuracy and cost. Configuration IDs correspond to Table 2.

no other configuration achieves strictly higher F1 while being no worse in cost metrics; similarly, for precision–recall trade-off.

Figure 4 presents the Pareto fronts for our primary detection metric (F1 score) with respect to (i) average token usage, (ii) average total execution time (second), and (iii) the precision–recall trade-off. The Pareto front highlights configurations that offer the best possible balance between accuracy and efficiency, enabling deployment choices aligned with cost or latency constraints.

Several top-ranked configurations (IDs 5, 18, 19, 16) lie on the Pareto front across these views, indicating that they offer competitive accuracy without excessive cost. The corresponding Pareto analyses for precision and recall metrics are provided in the Supplementary Material.

# 6. Discussion

## 6.1. Practical Implications

Integrating hallucination detection into enterprise RAG systems offers several advantages:

- **Risk Mitigation**: Early detection of unsupported answers mitigates the spread of misinformation in both customer-facing and internal applications.
- **Regulatory Compliance**: Many industries, such as healthcare and finance, require verifiable information; automated detection supports regulatory compliance.
- **Operational Efficiency**: Detecting hallucinations simultaneously with content delivery reduces the need for costly downstream human verification.

## 6.2. Ethical Considerations

Beyond technical performance, hallucination detection intersects directly with questions of fairness, accountability, and identity harms. Hallucinations in chatbot systems pose risks that extend beyond factual inaccuracies: they can reinforce harmful stereotypes, undermine user trust, and misrepresent marginalized communities in identity-sensitive contexts.

- **Reinforced stereotypes:** Language models are known to reproduce and amplify societal biases, as demonstrated by benchmarks such as StereoSet [31] and WinoBias [32]. In identity-sensitive deployments, hallucinated outputs risk reinforcing these biases in subtle but harmful ways.
- **Trust erosion:** Chatbots are only adopted at scale in high-stakes domains if users trust their outputs. Surveys on hallucination consistently highlight that exposure to unsupported or fabricated content undermines user trust in LLM systems [6, 7].

- **Identity harms:** Misrepresentations in generated responses may distort personal narratives or marginalize underrepresented groups, aligning with broader critiques that technical systems can reproduce social inequities if identity considerations are overlooked [33, 34].

By detecting hallucinations in a black-box, reference-free manner, **MetaRAG** can support safer deployment of RAG-based systems, particularly in settings where fairness, identity, and user well-being are at stake.

### 6.3. Limitations and Future Work

While MetaRAG demonstrates strong hallucination detection performance, several limitations remain:

- **Dataset Scope:** The study relies on a private, domain-specific dataset. This may limit external validity. *Future work should focus on curating or constructing public benchmarks designed to avoid overlap with LLM pretraining corpora, enabling more robust generalization.*
- **Annotation Granularity:** We lack factoid-level ground truth, which reduces our ability to assess fine-grained reasoning accuracy. *Providing such annotations in future datasets would support deeper consistency evaluations.*
- **Policy Hooks Not Evaluated:** The identity-aware deployment hooks introduced in Section 3.6 are presented only as a design concept. In our implementation, we used a fixed threshold of $\tau = 0.5$ across all queries. *Future research should implement and measure the effectiveness of topic-aware thresholds, forced citation, and escalation strategies in real-world chatbot deployments.*
- **Topic as Proxy (Design Limitation):** In Section 3.6, we suggest topic tags (e.g., pregnancy, asylum, labor) as privacy-preserving signals for stricter safeguards, rather than inferring protected attributes. This was not implemented in our experiments. As a design idea, it may also miss cases where risk is identity-conditioned but the query appears generic. *Future work should explore how to operationalize such topic-aware safeguards and investigate richer, privacy-preserving signals that better capture identity-sensitive risks.*
- **Model Dependency:** Current findings hinge on specific LLMs (GPT-4.1 variants). As models evolve, the behavior of MetaRAG may shift. *Future efforts should validate MetaRAG across open-source and emerging models to reinforce its robustness.*
- **Efficiency and Cost:** The verification steps add computational overhead, possibly impacting deployment in latency sensitive environments. *Investigating lighter-weight verification strategies and adaptive scheduling techniques could help mitigate this trade-off.*
- **Context Modality:** Our current formulation assumes that the retrieved context $C$ is textual, enabling direct comparison through language-based verification. However, RAG pipelines increasingly operate over multimodal contexts such as tables, structured knowledge bases, or images. *Future work should extend MetaRAG to handle non-textual evidence, requiring modality-specific verification strategies (e.g., table grounding, multimodal alignment).*

Together, these limitations highlight both immediate boundaries and promising future directions for enhancing MetaRAG's reliability, fairness, and efficiency.

## 7. Conclusion

Hallucinations in RAG-based conversational agents remain a significant barrier to trustworthy deployment in real-world applications. We introduced **MetaRAG**, a metamorphic testing framework for hallucination detection in retrieval-augmented generation (RAG) that operates without requiring ground

truth references or access to model internals. Our experiments show that MetaRAG achieves strong detection performance on a challenging proprietary dataset. Beyond general reliability, MetaRAG's factoid-level localization further supports identity-aware deployment by surfacing unsupported claims in sensitive domains (e.g., healthcare, migration, labor). Looking ahead, we see MetaRAG as a step toward safer and fairer conversational AI, where hallucinations are not only detected but also connected to safeguards that protect users in identity-sensitive contexts. This connection to identity-aware AI ensures that hallucination detection does not treat all users as homogeneous but provides safeguards that reduce disproportionate risks for identity-specific groups.

## Acknowledgment

This work was carried out during Channdeth's internship at Forvia. The authors thank the Forvia team in Bercy, Paris, for their guidance and support.

## Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly to improve the clarity and grammar of certain sentences and to rephrase text for better readability. After using this tool, the authors reviewed, edited, and verified all content to ensure accuracy and originality, and they take full responsibility for the publication's content.

## References

[1] O. et al, Gpt-4 technical report, 2024. URL: https://arxiv.org/abs/2303.08774. arXiv:2303.08774.

[2] A. G. et al, The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

[3] R. Bommasani, D. A. Hudson, E. Adeli, et al., On the opportunities and risks of foundation models, arXiv preprint arXiv:2108.07258 (2021).

[4] K. Singhal, S. Azizi, T. Tu, et al., Large language models encode clinical knowledge, Nature 620 (2023) 472–480.

[5] J. Maynez, S. Narayan, B. Bohnet, R. McDonald, On faithfulness and factuality in abstractive summarization, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 1906–1919. URL: https://aclanthology.org/2020.acl-main.173/. doi:10.18653/v1/2020.acl-main.173.

[6] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, P. Fung, Survey of hallucination in natural language generation, ACM Comput. Surv. 55 (2023). URL: https://doi.org/10.1145/3571730. doi:10.1145/3571730.

[7] X. Lyu, L. Zheng, Z. Wang, et al., Trustworthy and responsible large language models: A survey, arXiv preprint arXiv:2402.00176 (2024).

[8] P. Lewis, E. Perez, A. Piktus, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 9459–9474.

[9] T. Gao, C. Zhu, Z. Zhang, et al., Rag can still hallucinate: Faithfulness evaluation for retrieval-augmented generation, arXiv preprint arXiv:2304.09848 (2023).

[10] P. Liang, R. Bommasani, J. Lee, et al., Holistic evaluation of language models (2023). URL: https://arxiv.org/abs/2211.09110. arXiv:2211.09110.

[11] W. Kryscinski, B. McCann, C. Xiong, R. Socher, Evaluating the factual consistency of abstractive text summarization, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 9332–9346. URL: https://aclanthology.org/2020.emnlp-main.750/. doi:10.18653/v1/2020.emnlp-main.750.

[12] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, T. B. Hashimoto, Benchmarking large language models for news summarization, Transactions of the Association for Computational Linguistics 12 (2024) 39–57. URL: https://aclanthology.org/2024.tacl-1.3/. doi:10.1162/tacl_a_00632.

[13] H. Rashkin, V. Nikolaev, M. Lamm, L. Aroyo, M. Collins, D. Das, S. Petrov, G. S. Tomar, I. Turc, D. Reitter, Measuring attribution in natural language generation models, Computational Linguistics 49 (2023) 777–840. URL: https://aclanthology.org/2023.cl-4.2/. doi:10.1162/coli_a_00486.

[14] N. Dziri, E. Kamalloo, S. Milton, O. Zaiane, M. Yu, E. M. Ponti, S. Reddy, FaithDial: A faithful benchmark for information-seeking dialogue, Transactions of the Association for Computational Linguistics 10 (2022) 1473–1490. URL: https://aclanthology.org/2022.tacl-1.84/. doi:10.1162/tacl_a_00529.

[15] e. a. Wang, Factuality of large language models: A survey, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 19519–19529. URL: https://aclanthology.org/2024.emnlp-main.1088/. doi:10.18653/v1/2024.emnlp-main.1088.

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

[17] V. Rawte, A. Sheth, A. Das, A survey of hallucination in large foundation models, 2023. URL: https://arxiv.org/abs/2309.05922. arXiv:2309.05922.

[18] S. et al., Confabulation: The surprising value of large language model hallucinations, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 14274–14284. URL: https://aclanthology.org/2024.acl-long.770/. doi:10.18653/v1/2024.acl-long.770.

[19] R. Azamfirei, S. R. Kudchadkar, J. Fackler, Large language models and the perils of their hallucinations, Critical Care 27 (2023).

[20] N. Maleki, B. Padmanabhan, K. Dutta, Ai hallucinations: A misnomer worth clarifying, 2024. URL: https://arxiv.org/abs/2401.06796. arXiv:2401.06796.

[21] N. Chakraborty, M. Ornik, K. Driggs-Campbell, Hallucination detection in foundation models for decision-making: A flexible definition and review of the state of the art, ACM Comput. Surv. 57 (2025). URL: https://doi.org/10.1145/3716846. doi:10.1145/3716846.

[22] P. Koehn, R. Knowles, Six challenges for neural machine translation, in: T. Luong, A. Birch, G. Neubig, A. Finch (Eds.), Proceedings of the First Workshop on Neural Machine Translation, Association for Computational Linguistics, Vancouver, 2017, pp. 28–39. URL: https://aclanthology.org/W17-3204/. doi:10.18653/v1/W17-3204.

[23] P. Manakul, A. Liusie, M. J. F. Gales, Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models, 2023. URL: https://arxiv.org/abs/2303.08896. arXiv:2303.08896.

[24] T. Y. Chen, S. C. Cheung, S. M. Yiu, Metamorphic testing: A new approach for generating next test cases, 2020. URL: https://arxiv.org/abs/2002.12543. arXiv:2002.12543.

[25] S. Segura, G. Fraser, A. B. Sanchez, A. Ruiz-Cortés, A survey on metamorphic testing, IEEE Transactions on Software Engineering 42 (2016) 805–824. doi:10.1109/TSE.2016.2532875.

[26] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, S. Podder, Identifying implementation bugs in machine learning based image classifiers using metamorphic testing, in: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA '18, ACM, 2018, p. 118–128. URL: http://dx.doi.org/10.1145/3213846.3213858. doi:10.1145/3213846.3213858.

[27] M. T. Ribeiro, T. Wu, C. Guestrin, S. Singh, Beyond accuracy: Behavioral testing of NLP models with CheckList, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4902–4912. URL: https://aclanthology.org/2020.acl-main.442/. doi:10.

18653/v1/2020.acl-main.442.
[28] B. Yang, M. A. A. Mamun, J. M. Zhang, G. Uddin, Hallucination detection in large language models with metamorphic relations, 2025. URL: https://arxiv.org/abs/2502.15844. arXiv:2502.15844.
[29] G. Guo, A. Aleti, N. Neelofar, C. Tantithamthavorn, Y. Qi, T. Y. Chen, Mortar: Multi-turn metamorphic testing for llm-based dialogue systems, 2025. URL: https://arxiv.org/abs/2412.15557. arXiv:2412.15557.
[30] N. Li, Y. Li, Y. Liu, L. Shi, K. Wang, H. Wang, Drowzee: Metamorphic testing for fact-conflicting hallucination detection in large language models, 2024. URL: https://arxiv.org/abs/2405.00648. arXiv:2405.00648.
[31] M. Nadeem, A. Bethke, S. Reddy, StereoSet: Measuring stereotypical bias in pretrained language models, in: C. Zong, F. Xia, W. Li, R. Navigli (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021. URL: https://aclanthology.org/2021.acl-long.416/. doi:10.18653/v1/2021.acl-long.416.
[32] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, K.-W. Chang, Gender bias in coreference resolution: Evaluation and debiasing methods, in: M. Walker, H. Ji, A. Stent (Eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 15–20. URL: https://aclanthology.org/N18-2003/. doi:10.18653/v1/N18-2003.
[33] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, K. Crawford, Datasheets for datasets, Commun. ACM 64 (2021) 86–92. URL: https://doi.org/10.1145/3458723. doi:10.1145/3458723.
[34] A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, J. Vertesi, Fairness and abstraction in sociotechnical systems, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 59–68. URL: https://doi.org/10.1145/3287560.3287598. doi:10.1145/3287560.3287598.

# A. Prompt Templates

## A.1. Factoid Decomposition Prompt

We provide the exact template used to extract atomic factoids from model responses.

```
export const factExtractionPrompt = (inputText: string) => `
You are a fact extraction assistant.
Your task is to extract all specific factual propositions from the given text.

Instructions:
1. Extract every distinct factual statement present in the input, even if the statement is incorrect, ambiguous,
     or nonsensical.
2. Each extracted proposition must be a complete, standalone sentence.
3. Each sentence must express only one atomic fact. (An atomic fact cannot be split into simpler factual
     statements.)
4. If a sentence contains multiple facts, split them into multiple atomic fact sentences.
5. Do not paraphrase, rewrite, summarize, interpret, infer, or judge any part of the input. Only extract and
     restate what is explicitly written.
6. Do not omit or correct any statements, regardless of their factual accuracy.
7. Output your answer as a JSON array of strings, with each element being one atomic factual sentence.

Example:
Input:
Marie Curie discovered polonium and radium, and Albert Einstein developed the theory of relativity in 1905.

Output:
[
  "Marie Curie discovered polonium.",
  "Marie Curie discovered radium.",
  "Albert Einstein developed the theory of relativity in 1905."
```

```
]

Now, extract atomic facts from this text:

Input:
${inputText}
`;
```

## A.2. Mutation Generation Prompts

We provide both synonym and antonym generation templates

```
export const antonymPrompt = (
 count: Integer,
 question: string,
 factoid: string
) => `
You will be given a question and a factual answer (factoid).

Your task is to generate ${count} *negations* (contradictory statements) of the factoid, based on the context of
     the question.

Instructions:
- Each negation must directly contradict the factoid, focusing on what the question asks.
- Do not add new information not present in the factoid or question.
- Do not use double negations or wording that preserves the original meaning.
- Each negation must be a meaningful, grammatically correct sentence.
- Do not introduce unrelated facts.
- Ensure that each negation is relevant to the 'questions context.
- **Just return the sentences, one per line, without numbers or bullets, and nothing else.**

Example:
Question: Where was Einstein born?
Factoid: Einstein was born in Germany.
Good Antonym: Einstein was not born in Germany.
Bad Antonym: Einstein visited Germany. (not a contradiction)
Bad Antonym: Einstein was born in Austria. (adds new information)
Bad Antonym: Einstein was not not born in Germany. (double negation)
Bad Antonym: Was not born in Germany. (missing subject)

Question: ${question}

Factoid: ${factoid}
`;
```

```
export const synonymPrompt = (
 count: Integer,
 question: string,
 factoid: string
) => `
You will be given a question and a factual answer (factoid).

Your task is to generate ${count} *synonyms* (paraphrased statements with the same meaning) of the factoid, based
      on the context of the question.

Instructions:
- Each output must be a single atomic factual claim (cannot be split into smaller facts).
- Use only information explicitly present in the question or factoid. Do not invent, infer, or add external
     knowledge.
- A correct synonym is a statement that means exactly the same thing as the factoid, even if the wording is
     different.
- Do not output partial phrases, keywords, or combine/split facts.
- Each synonym must be a complete, grammatically correct sentence.
- Just return the sentences, one per line, without numbers, bullets, or any other output.

Example:
Question: Where was Einstein born?
Factoid: Einstein was born in Germany.
Good Synonym: Germany is the country where Einstein was born.
Bad Synonym: Einstein visited Germany. (not equivalent)
Bad Synonym: Einstein was born. (incomplete)
```

```
Question: ${question}

Factoid: ${factoid}
`;
```

### A.3. Factoid Verification Prompt

The verification step compares mutated factoids against retrieved context using entailment/contradiction/neutral checks.

```
export const verifyPrompt= (
  statement: string,
  context: string
) => `
You will be given a statement and passages that represent the ground truth.

Determine if the statement is supported by the passage, either explicitly or through clear implication.

Answer with one of the following **only**:
- YES: if the statement is clearly and completely supported by the passages.
- NO: if the statement is contradicted or directly refuted by the passages.
- NOT SURE: if the passage does not contain enough information to confirm or deny the statement.

Respond with YES, NO, or NOT SURE. Then, in one short sentence, explain the reason for your answer.

Examples:

Passages (Ground Truth): "Alice was born in Paris and moved to New York at the age of five."
Statement: "Alice spent her early childhood in France."
Answer: YES. The passage states Alice was born in Paris, which is in France.

Passages (Ground Truth): "Bob has never visited Japan but plans to travel there next summer."
Statement: "Bob visited Japan last year."
Answer: NO. The passage says Bob has never visited Japan

Passages (Ground Truth): "Carol enjoys outdoor activities like hiking and cycling."
Statement: "Carol loves swimming."
Answer: NOT SURE. There is no information in the passages about Carol and swimming.

Now, perform the task:

Passages (Ground Truth): ${context}
Statement: ${statement}
Answer:`;
```

All verification LLMs were run with temperature $T = 0.0$ to ensure determinism.

## B. Dataset and Annotation

### B.1. Dataset

Our evaluation dataset consists of 23 internal enterprise documents, unseen during LLM training. Each document was segmented into chunks of approximately a few hundred tokens, and we retrieved up to $k = 3$ chunks per query. Retrieval used cosine similarity over OpenAI `text-embedding-3-large`.

Figure 5 further illustrates the token length distributions of generated answers and retrieved contexts. Generated answers are typically shorter (median $\approx$ 83 tokens), while retrieved contexts are longer (median $\approx$ 572 tokens), reflecting the compression and grounding challenges faced by the RAG system.

### B.2. Human Annotation Protocol

The annotation dataset was constructed in three steps. First, we collected responses produced by the RAG system on enterprise documents. Second, we used an LLM-based verifier to provide an initial label (**faithful** or **hallucinated**) for each response based on its retrieved context. Finally, human

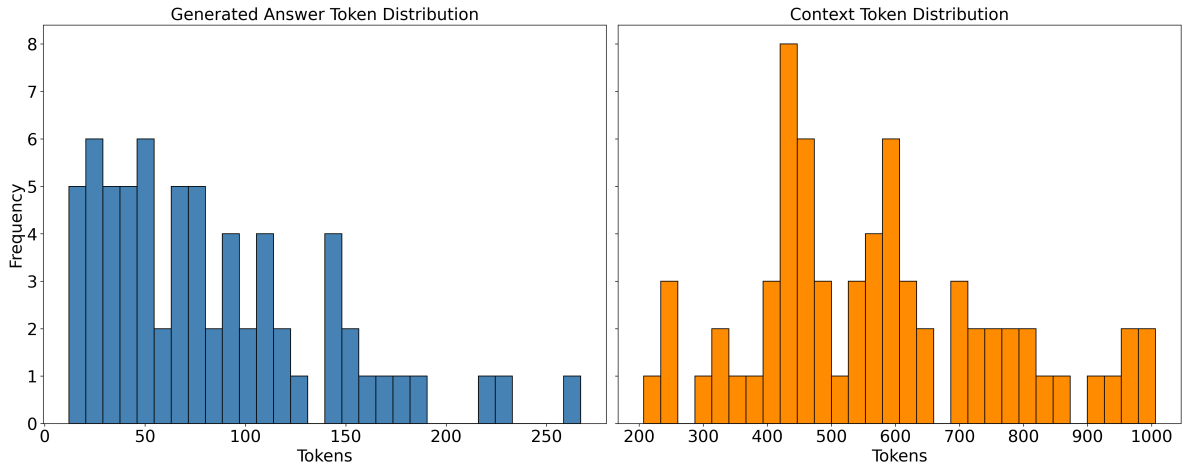



**Figure 5:** Token length distributions of generated answers (left) and retrieved context passages (right).

annotators reviewed the RAG responses together with their retrieved evidence and assigned gold labels. Annotators were instructed to:

- Mark each response as **faithful** or **hallucinated**.
- Consider a response hallucinated if any atomic factoid was not supported by retrieved evidence.
- Resolve ambiguous cases by majority vote.

To ensure class balance across conditions, a subset of responses was lightly edited (e.g., by introducing or removing unsupported factual details) so that hallucinated and non-hallucinated examples were more evenly represented. These edits were applied before annotation, and annotators labeled both original and modified responses using the same guidelines. Figure 6 illustrates the final label distribution in our dataset, confirming that hallucination and not hallucination cases are reasonably balanced.

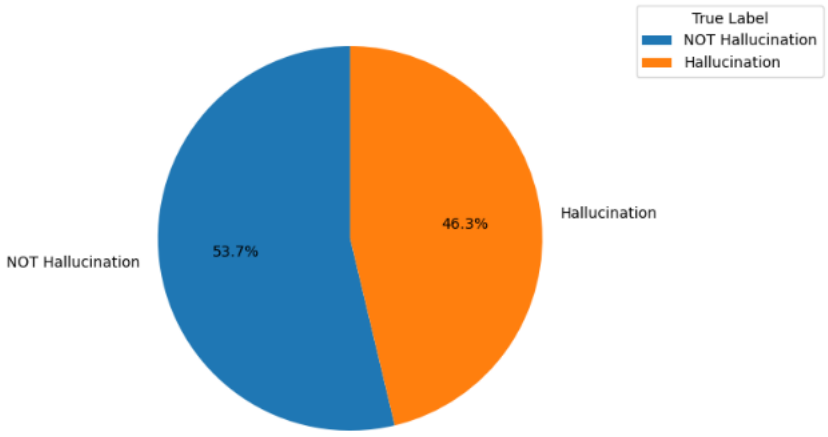



**Figure 6:** Proportion of hallucination vs. non-hallucination labels in the annotated dataset.

# C. Extended Results

## C.1. Result

Table 4 reports the full ablation results across prompt settings, mutation counts, and verifier models.

**Table 4**
Complete MetaRAG ablation results across 26 configurations. Configuration format: Decomposition Model / Mutation Generation Model / Verifier Model / Number of Mutations ($N$) / Temperature. *Total (avg)* denotes the average execution time, and *Cost (avg)* denotes the average token usage per run.

| ID | Configuration | Precision | Recall | F1 | Accuracy | Time (avg) | Tokens (avg) |
|---|---|---|---|---|---|---|---|
| 0 | 41 / mini / mini/2/0.0 | 0.784 | 0.935 | 0.853 | 0.851 | 2.490 | 35.5k |
| 1 | mini / 41 / 41/2/0.7 | 0.776 | 1.000 | 0.874 | 0.866 | 3.542 | 29.0k |
| 2 | mini / 41 / multi /2/0.7 | 0.828 | 0.935 | 0.878 | 0.880 | 3.205 | 28.8k |
| 3 | mini / 41 / mini/2/0.7 | 0.806 | 0.935 | 0.866 | 0.866 | 3.011 | 28.7k |
| 4 | mini / 41 / 41/2/0.0 | 0.757 | 1.000 | 0.861 | 0.851 | 2.705 | 28.9k |
| 5 | mini / 41 / multi/2/0.0 | 0.885 | 1.000 | 0.939 | 0.940 | 2.287 | 29.1k |
| 6 | mini / 41 / mini/2/0.0 | 0.784 | 0.935 | 0.853 | 0.851 | 2.581 | 29.0k |
| 7 | mini / mini / 41/2/0.7 | 0.764 | 0.935 | 0.841 | 0.836 | 2.064 | 29.1k |
| 8 | mini / mini /multi/2/0.7 | 0.828 | 0.935 | 0.878 | 0.880 | 2.033 | 28.9k |
| 9 | mini / mini / mini/2/0.7 | 0.837 | 1.000 | 0.911 | 0.910 | 1.814 | 29.1k |
| 10 | 41 / 41 / mini/2/0.0 | 0.790 | 0.968 | 0.870 | 0.866 | 3.471 | 36.0k |
| 11 | 41 / mini / 41/2/0.0 | 0.784 | 0.935 | 0.853 | 0.851 | 3.338 | 36.7k |
| 12 | 41 / mini / multi /2/0.0 | 0.764 | 0.935 | 0.841 | 0.836 | 3.204 | 37.1k |
| 13 | mini / mini / 41/2/0.0 | 0.757 | 0.903 | 0.824 | 0.821 | 2.200 | 28.7k |
| 14 | mini / mini / multi/2/0.0 | 0.828 | 0.935 | 0.878 | 0.880 | 2.834 | 28.7k |
| 15 | mini / mini / mini/2/0.0 | 0.812 | 0.968 | 0.883 | 0.881 | 2.267 | 29.1k |
| 16 | mini / 41 / multi/5 /0.0 | 0.882 | 0.968 | 0.923 | 0.925 | 7.671 | 76.5k |
| 17 | mini / 41 / mini/5/0.0 | 0.812 | 0.968 | 0.883 | 0.881 | 5.693 | 74.0k |
| 18 | mini / mini / 41/5/0.7 | 0.909 | 0.968 | 0.937 | 0.940 | 5.185 | 74.6k |
| 19 | mini / mini / 41/5 /0.0 | 0.935 | 0.935 | 0.935 | 0.940 | 5.215 | 77.5k |
| 20 | mini / mini / mini/5/0.7 | 0.866 | 0.838 | 0.852 | 0.865 | 3.710 | 77.4k |
| 21 | mini / mini / multi/5/0.7 | 0.896 | 0.838 | 0.866 | 0.880 | 5.903 | 80.2k |
| 22 | mini / mini / multi/5/0.0 | 0.964 | 0.870 | 0.915 | 0.925 | 5.812 | 76.2k |
| 23 | mini / mini / mini/5/0.0 | 0.874 | 0.903 | 0.888 | 0.895 | 3.515 | 75.1k |
| 24 | 41 / mini / multi /5/0.0 | 0.900 | 0.870 | 0.885 | 0.895 | 6.345 | 89.8k |
| 25 | 41 / mini / mini/5/0.0 | 0.833 | 0.806 | 0.819 | 0.835 | 4.631 | 88.2k |

## C.2. Consistency Study

To assess the robustness of MetaRAG to random initialization, we report mean and standard deviation of the main evaluation metrics over 5 different random seeds for the top configurations. We also include the coefficient of variation (CV) for Precision and Recall, which provides a normalized measure of variability relative to the mean.

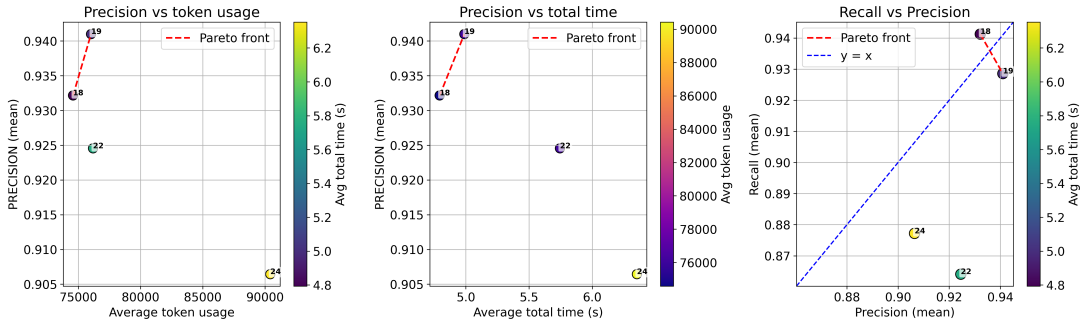
**Table 5**
Run-to-run consistency for top precision configurations (mean ± standard deviation over 5 seeds) and coefficient of variation (CV) for Precision.

| ID | F1 | Precision | Recall | CV (Prec.) |
|---|---|---|---|---|
| 19 | 0.9347 ± 0.0305 | 0.9410 ± 0.0357 | 0.9286 ± 0.0272 | 3.79% |
| 18 | 0.9356 ± 0.0089 | 0.9322 ± 0.0439 | 0.9413 ± 0.0278 | 4.71% |
| 22 | 0.8928 ± 0.0351 | 0.9246 ± 0.0359 | 0.8641 ± 0.0478 | 3.88% |
| 24 | 0.8911 ± 0.0272 | 0.9064 ± 0.0144 | 0.8772 ± 0.0475 | 1.59% |

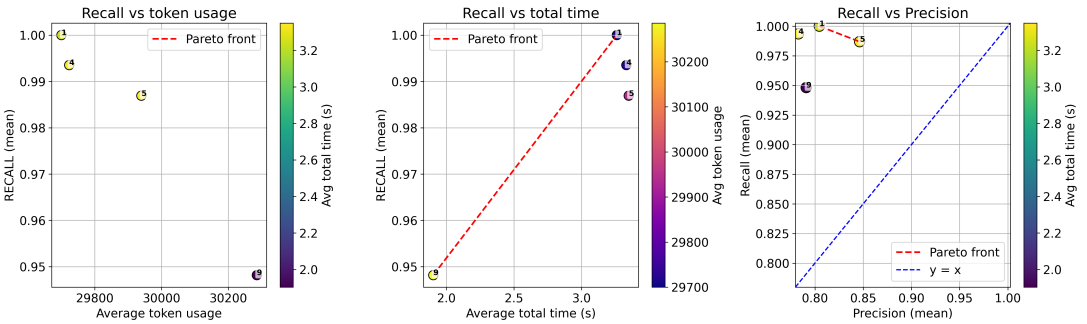
**Table 6**
Run-to-run consistency for top recall configurations (mean ± standard deviation over 5 seeds) and coefficient of variation (CV) for Recall.

| ID | F1 | Precision | Recall | CV (Recall) |
|---|---|---|---|---|
| 1 | 0.8910 ± 0.0298 | 0.8045 ± 0.0496 | 1.0000 ± 0.0000 | 0.00% |
| 4 | 0.8756 ± 0.0217 | 0.7829 ± 0.0283 | 0.9935 ± 0.0145 | 1.46% |
| 5 | 0.9108 ± 0.0346 | 0.8463 ± 0.0503 | 0.9869 ± 0.0179 | 1.82% |
| 9 | 0.8623 ± 0.0434 | 0.7910 ± 0.0414 | 0.9482 ± 0.0491 | 5.18% |

These results (Table 5 and Table 6) demonstrate that MetaRAG maintains stable performance across random seeds, particularly for high-precision configurations (e.g., IDs 24) and high-recall configurations




(a) Precision Pareto front (vs. cost/latency)




(b) Recall Pareto front (vs. cost/latency)

**Figure 7:** Pareto front analysis for retrieval budget. Each point corresponds to a MetaRAG configuration; non-dominated (Pareto-optimal) points are highlighted. Subfigure (a) shows Precision trade-offs, and Subfigure (b) shows Recall trade-offs.

(e.g., IDs 1, 4 and 5). This stability supports the reliability of the Pareto front analysis presented in the following section.

## C.3. Pareto Front Analysis

We further analyze robustness and metric-specific trade-offs in this Supplementary Material. A configuration is Pareto-optimal if no other configuration achieves strictly higher performance while being no worse in the cost metrics. Figures 7a and 7b present the corresponding Pareto fronts for Precision and Recall. These analyses confirm that the same top-ranked configurations (IDs 5, 18, 19, and 16) consistently offer strong performance–efficiency trade-offs across multiple evaluation criteria.

In our setting, the positive class corresponds to hallucinations, while the negative class corresponds to faithful responses (no hallucinations). Hence, **high Precision** means that flagged hallucinations are rarely false positives, which is critical in *safety-critical* and *trustworthy applications*. Conversely, **high Recall** ensures that most hallucinations are detected, though at the cost of occasionally misclassifying faithful responses. Such recall-oriented configurations may be advantageous in exploratory or diagnostic scenarios. In practice, high-precision operating points (e.g., IDs 18 and 19) reduce false alarms in safety-critical pipelines, while high-recall points (e.g., IDs 1 and 4) maximize coverage in exploratory settings. This mirrors standard alert-system trade-offs and clarifies how **MetaRAG** can be tuned for different deployment objectives. Selections based on F1 score represent a balanced compromise suitable for general-purpose use cases.

## D. Implementation Notes

All **MetaRAG** experiments were implemented in TypeScript with asynchronous API calls to the LLMs, allowing multiple requests to be processed concurrently. This parallelization reduced the average

end-to-end execution time per run, without affecting accuracy metrics. The reported runtime and cost results in Table 4 are therefore representative of a practical and scalable setup.