Hallucination Span Detection in Large Language Models

Arijit Gupta

Supervisor: Dr. Carolina Scarton

Co-Supervisor: Dr. Fatima Haouari

A report submitted in fulfilment of the requirements
for the degree of MSc in Computer Science with Speech and Language Processing

in the

Department of Computer Science

September 13, 2025

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Arijit Gupta

Signature: Arijit Gupta

Date: September 13, 2025

Abstract

This dissertation investigates the efficacy of open-source large language models (LLMs) for hallucination span detection tasks. While proprietary, state-of-the-art models have demonstrated strong capabilities in identifying factual inconsistencies, their closed-source nature presents limitations for research and application. This study evaluates how open-source LLMs compare against these proprietary models in hallucination detection and assesses their performance as context retrieval engines. Furthermore, we explore the effects of prompt modification and model ensembling on task performance. Our key findings reveal that open-source models are comparable to their larger, proprietary counterparts in hallucination detection, facing similar limitations. They prove to be highly effective context retrievers, with the overall pipeline performance dropping by only 1-2% when proprietary retrieval engines are replaced. We also demonstrate that ensembling the best-performing model using varied temperatures yields a slight boost in performance. Notably, our proposed model with an open-source LLM is the 3rd best compared to the best performing participating models, all using closed models, for the English language in the Mu-SHROOM shared task. Based on our metrics and a manual error classification, we conclude by suggesting future directions to improve hallucination detection techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large Language Models (LLMs), have introduced a lot of changes in the world or artificial intelligence and how it is used in day to day lives. It defines the way that people and computers interact in the modern day and age. There have been great advancements in language technology over the last few years and commercial models like OpenAI's GPT series and Google's Gemini family are incredibly good at understanding, translating, summarizing and generating text that sounds incredibly human [7]. Innovation in many areas has been driven due to the availability of these technologies, like software development, content creation and even scientific research [31]. And while they do have a lot of good points, LLMs come with their own set of shortcomings despite being a very powerful technology. The most major ongoing problem is that LLMs tend to 'hallucinate'. Hallucinations are answers produced by these models that are semantically correct and sound convincing, but are factually wrong. They either don't make any sense (e.g. Berlin is the capital of France) or are not grounded to the information that was given as context. These hallucinations or incorrect answers are a major roadblock in the adoption of thse technologies in critical and high stakes environments in the real world [15].

This problem with incorrect answers and hallucinations isn't simply bound to being a technical issue, it's a core problem in increasing the trust and safety in LLMs. Information that is false, or made up by an LLM can have severe consequences in specific fields. For instance, if a model hallucinates a drug dosage value in a healthcare use case, it could harm a patient and be the difference between life and death. This risk has leds to the creation of domain specific benchmarks like MedHalu [2] and MediQ [19] to study these failures. Similarly, a hallucinated market analysis in finance could lead to significant monetary losses [22]. In legal scenarios there have been situations where LLMs have generated citations to non-existent court cases, potentially leading to legal trouble and professional sanctions [25]. As these models are being used more and more in daily lives to make trivial and important decisions, it is required to ensure that their outputs are grounded in evidence. Every hallucinated answer given by a model reduces its trustworthiness as a system and makes it less useful. Hence, detecting and mitigating these hallucination is a crucial task in order to integrate LLMs into tasks where they have the potential to be very useful.

To solve this problem, we first need to understand its different forms. Researchers generally group hallucinations into two main categories. Factuality hallucinations happen when a model's output contradicts a known fact about the world, like saying that Paris is the capital of Spain and faithfulness hallucinations happen when the output doesn't stick to the source text or the user's instructions [12]. This can mean the model ignores a command, contradicts the provided information, or makes a logical error [23]. These mistakes can be hard to spot because they are often mixed into well-written, coherent text. The goal isn't just to know that an error happened, but to find the exact "hallucination span"—the specific words or phrases that are wrong.

To meet this critical need, researchers have developed many different detection methods. One popular approach involves model-internal and self-supervised techniques. These methods use the model's own internal data, like the probability of certain words appearing, to guess if it is uncertain about its answer [27]. Other methods use self-consistency checks, based on the idea that if a model truly knows something, it will give similar answers when asked the same question multiple times [24]. Some techniques even try to stop hallucinations before they happen by analyzing the model's internal state as it generates a response [4]. While these ideas are clever, they are limited because a model's internal signals don't always point to a factual error, and they can't verify information the model wasn't trained on.

This dissertation will focus on a different approach that is more robust and easier to verify externally: retrieval-augmented and knowledge-based detection methods. The main idea behind this method, which is the foundation of modern systems like Retrieval-Augmented Generation (RAG) [17], is to connect the LLM's answers to a trusted external source of knowledge. Instead of just relying on the information stored in the model's memory, these systems first search for relevant documents from a knowledge base (like Wikipedia or company files). They then use this information to guide and check the model's response. For example, methods like HalluSearch break down a model's answer into small, verifiable claims and check each one against an external source [1]. By connecting the generation process to a source of truth, these methods provide a powerful way to improve accuracy and detect hallucinations more precisely.

Despite the potential of these techniques, there is a major gap in our understanding of how they work across the growing variety of LLMs. Most cutting-edge research uses powerful, proprietary models from companies like OpenAI or Google. While these models are the most advanced, they are also "black boxes" and expensive to use, which limits access for many researchers. At the same time, the open-source community has released many powerful, smaller, and more transparent models (like Llama, Mistral, and Qwen) that are available to everyone [28, 16].

By achieving these goals, this dissertation will make an important contribution to the field of trustworthy AI. It will give developers and organizations valuable, evidence-based insights into how these critical safety tools can be scaled and adapted. The results will help guide the choice and use of hallucination detection strategies, making it possible to responsibly use both large-scale proprietary models and more accessible open-source alternatives.

This leaves us with a critical, unanswered question: How well do the best retrieval-based hallucination detection methods work when used with these more accessible, open-source models? Are there major differences in accuracy, cost, or complexity?

This dissertation aims to fill this gap with a systematic, hands-on comparison. The main goals are:

- To set up and test a group of leading retrieval-augmented and knowledge-based methods for detecting and locating hallucination spans.
- To perform a detailed comparison of how well these methods work on two different types of models: top-tier, closed-source models and several popular, high-performing open-source models.
- To identify how well the open-source models do on the task of context retrieval given a query and answer, especially compared to proprietary models developed for this specialised task
- To identify the effects of prompt complexity, LLM ensembles and majority voting methodologies on the specific task of hallucination span detection

This dissertation is organized as follows: Chapter 2 reviews the literature on hallucination benchmarks and detection methods. Chapter 3 explains the experimental setup, including the models, dataset, and detection pipelines. Chapter 4 presents the results of the comparison, and Chapter 5 concludes with a discussion of the findings, their meaning, and ideas for future work.

Chapter 2

Literature Survey

2.1 Benchmarks and Datasets for Hallucination Evaluation

2.1.1 Multilingual and General-Purpose Datasets

A significant challenge in the study of hallucinations in LLMs is the development of robust and comprehensive benchmarks for their evaluation. Early efforts in this area have led to the creation of several key datasets, each with its own focus and methodology. Among these, the Mu-SHROOM and HaluEval benchmarks have emerged as important resources for researchers.

The Mu-SHROOM dataset, developed for the SemEval-2025 shared task, specifically addresses the need for multilingual evaluation of hallucination detection [14]. A key contribution of this benchmark is its focus on multilingual, token-level span annotation. This method allows for a granular analysis of where hallucinations occur within a text, moving beyond simple binary classifications. Crucially, the evaluation is self-contained, assessing an LLM's output against a provided source text, which removes the dependency on external verification tools during the annotation and detection process. The focus on 14 languages is also vital for understanding how hallucinations manifest across different linguistic contexts.

In a similar vein, HaluEval provides a comprehensive, large-scale benchmark for evaluating the ability of LLMs to recognize hallucinations [18]. HaluEval is composed of a substantial collection of both generated and human-annotated samples of hallucinated content, covering both general user queries and task-specific examples in areas such as question answering, knowledge-grounded dialogue, and text summarization. While HaluEval provides a broad foundation, its framework has inspired further research into domain-specific testing, leading to the development of benchmarks in specialized fields such as finance and medicine to assess LLM reliability in high-stakes applications. The findings from HaluEval suggest that while current LLMs face significant challenges in identifying hallucinations, their performance can be improved by providing them with external knowledge or by prompting them to engage in reasoning steps.

Together, the Mu-SHROOM and HaluEval benchmarks represent significant strides in the development of resources for the evaluation of hallucinations in LLMs. While Mu-SHROOM's strength lies in its multilingual focus and self-contained, token-level approach, HaluEval's contribution is its comprehensive and diverse collection of hallucinated samples that has paved the way for more domain-specific evaluations. Both datasets are invaluable for researchers working to understand and mitigate the phenomenon of hallucination in LLMs.

2.1.2 Task-Specific Benchmarks

While general-purpose benchmarks provide a broad overview, researchers have also developed task-specific datasets to evaluate hallucinations in more focused applications. These benchmarks are crucial for understanding how hallucinations manifest in different contexts, such as summarization, dialogue, and multimodal tasks.

For text summarization, FaithBench offers a diverse benchmark designed to test for hallucinations in summaries generated by modern LLMs [6]. Summarization is particularly prone to hallucinations, as models may invent details to create a more coherent narrative. FaithBench addresses this by providing human-annotated ground truth for summaries from a wide range of modern LLMs. It introduces a nuanced taxonomy that distinguishes between intrinsic hallucinations (contradicting the source text) and extrinsic hallucinations (adding information not found in the source). This allows for a more detailed analysis of the types of errors models make.

In the area of conversational AI, HalluDial serves as a large-scale benchmark for evaluating hallucinations at the dialogue level [23]. Unlike single-turn question answering, dialogue systems must maintain factual consistency over multiple turns. HalluDial addresses this by creating realistic dialogue scenarios where LLMs are prone to making mistakes. The benchmark includes detailed annotations that not only identify whether a hallucination occurred but also pinpoint its exact location in the conversation and provide an explanation. This helps researchers evaluate a model's ability to stay grounded in fact throughout an extended interaction.

Addressing the challenges of newer models, LongHalQA evaluates hallucinations in multimodal and long-context scenarios [26]. As LLMs are increasingly asked to process long documents or a combination of text and images, the risk of hallucination grows. LongHalQA is specifically designed to test a model's ability to remain factually grounded when dealing with large amounts of complex information. It features long and complex hallucinatory texts generated by GPT-4V and uses a multiple-choice question format for evaluation, which avoids the need for computationally expensive LLM-based evaluators.

Finally, TruthfulQA takes a different approach by measuring how well models avoid mimicking human falsehoods [20]. The benchmark consists of questions designed to trigger "imitative falsehoods"—common misconceptions or false beliefs that are prevalent on the internet. To perform well, a model must avoid repeating these falsehoods, even if they appear frequently in its training data. TruthfulQA assesses a model's "truthfulness" by rewarding answers that are factually correct and penalizing those that repeat common myths. This provides a unique perspective on the hallucination problem by focusing on a model's tendency to learn and reproduce incorrect information from its training data.

2.1.3 Domain-Specific Datasets

Beyond task-specific evaluations, researchers have also created benchmarks for high-stakes domains where factual accuracy is critical. The healthcare sector is a primary example, as misinformation can have severe consequences for patient safety.

In this area, MedHalu provides the first benchmark focused specifically on hallucinations in response to healthcare queries [2]. The dataset contains a diverse set of health-related questions and corresponding hallucinated answers from LLMs. Each hallucination is carefully labeled by type and includes annotations for the specific text spans that are fabricated or incorrect. A key finding from the study is that LLMs perform poorly when asked to detect medical hallucinations, often performing no better than a layperson and significantly worse than a medical expert. This highlights the risks of using LLMs for medical advice and underscores the need for specialized detection methods in sensitive domains.

2.2 Methodologies for Hallucination Detection

2.2.1 Model-Internal and Self-Supervised Approaches

In addition to creating benchmarks for evaluating hallucinations, researchers have developed various methods for detecting them. These techniques can be broadly categorized into those that analyze the model's internal state and those that rely on external consistency checks.

One approach to hallucination detection involves analyzing the token probabilities generated by the model [27]. This method is based on the idea that when a model hallucinates, it is often less "confident" in the tokens it generates. By examining the probability distribution of the output, researchers can identify tokens or sequences of tokens with low probabilities. These low-probability tokens are considered indicators of potential hallucinations, as they suggest that the model is generating text that is not well-supported by its training data. This technique is a form of intrinsic evaluation, as it relies on the model's internal state rather than external knowledge sources. Suppose the input query is “What is the currency of Wakanda?” and the model answers “The currency of Wakanda is the Wakandan Dollar.” The token “Wakandan” receives a very low probability under the model (since Wakanda is fictional). As the paper notes, such low-confidence tokens often signal hallucinations, indicating that this answer is not grounded in real knowledge.

Inspired by human interrogation techniques, InterrogateLLM is a zero-resource method for detecting hallucinations [29]. This approach works by first generating an answer to a given query. Then, it uses an LLM to generate a set of new questions that could have led to that answer. These new questions are compared to the original query. If there is a significant mismatch, it suggests that the answer has drifted from the original topic and is likely a hallucination. This method is notable for its ability to detect hallucinations without relying on external databases or fact-checking tools. Consider the query “Who discovered penicillin?”

The LLM might answer “Alexander Fleming discovered penicillin in 1928, working with his colleagues to study bacteria on moldy petri dishes.” InterrogateLLM then uses this answer to reconstruct the question, prompting the model to generate possible queries like “When was penicillin discovered?” or “Who worked with Fleming on penicillin?” If the reconstructed questions diverge from the original (e.g. asking about Fleming’s partner), this inconsistency flags the answer as hallucinated.

The Attention-Guided Self-Reflection (AGSER) method uses the model’s internal attention mechanisms to detect hallucinations [21]. The technique first identifies which parts of the input query the model paid the most attention to (the ”attentive” query) and which parts it paid less attention to (the ”non-attentive” query). It then generates separate answers for each of these partial queries and compares them to the original answer. A large difference in consistency between the attentive and non-attentive responses is used as an indicator of a hallucination. The underlying theory is that hallucinations are often triggered by the parts of a prompt that the model pays less attention to. Take the query “Who is the author of the book At Home in Mitford, and what year was it published?” The model’s answer is “Jan Karon, in 1996.” They split the query into an attentive part (“At Home in Mitford”) and a non-attentive part (other words). When they ask the attentive query “Who is the author of At Home in Mitford, published what year?”, the model answers “Jan Karon, in 1996.”, consistent with the original. But when they ask a nonsensical non-attentive query “Who is book The what year was it published?”, the model answers “The author of The Nightingale is Kristin Hannah, 2015.”, a different book. Because the attentive answer matches the original while the non-attentive answer is unrelated, AGSER correctly identifies this as a non-hallucination case.

SelfCheckGPT is a widely cited method that relies on the principle of consistency [24]. It operates on the idea that if an LLM has factual knowledge about a topic, its responses will be consistent even when sampled multiple times with a degree of randomness. To verify a statement, SelfCheckGPT generates several alternative responses to the same prompt. It then checks whether the original statement is supported by these alternative responses. If the different responses contradict each other, the original statement is flagged as a potential hallucination. This approach is particularly useful as it can be applied to black-box models where internal token probabilities are not accessible. For the prompt “Who is Giuseppe Mariani?” the model might initially answer “Giuseppe Mariani was an Italian professional footballer who played as a forward. He was born in Milan, Italy, and died in Rome.” SelfCheckGPT generates alternative answers by sampling, for example: “Giuseppe Mariani was an Italian painter born in Naples in 1882.” and “Giuseppe Mariani was an Italian violinist born in Pavia in 1836.” These contradictory biographies demonstrate inconsistency, which strongly indicates the original answer contained hallucinated content.

Unlike the other methods, DoLa (Decoding by Contrasting Layers) is an inference-time technique designed to mitigate hallucinations before they occur [9]. This method is based on the finding that factual knowledge is often more robustly encoded in the later layers of a transformer model. DoLa works by contrasting the output distributions of a ”mature” (later) layer with a "premature" (earlier) layer. By amplifying the predictions from the mature layer, the model is encouraged to generate text that is more factually grounded. This approach effectively steers the model away from generating hallucinations during the decoding process itself. Given the query "Where is the capital of Washington State?", a standard LLM might answer "Seattle" (incorrectly, since Olympia is the capital). DoLa compares the model's output distributions from an earlier layer versus a later layer. In the higher layers of the model, the probability of "Olympia" increases, while "Seattle" is down-weighted. By amplifying the higher-layer signal, DoLa's decoding yields "Olympia" as the answer, correcting the hallucination in real time.

2.2.2 Retrieval-Augmented and Knowledge-Based Detection

While the previous methods focus on the internal state of the model, another important class of techniques involves using external knowledge to detect and mitigate hallucinations. These methods often integrate retrieval-augmented generation (RAG) and other verification pipelines.

Retrieval-Augmented Generation (RAG) is a common technique for grounding LLM outputs in external knowledge. However, standard RAG pipelines can still produce hallucinations. HalluSearch is a method designed to improve the reliability of RAG systems by incorporating a search-based verification step [1]. After an initial answer is generated, HalluSearch formulates search queries based on the answer's key claims. It then uses a search engine to find supporting evidence. By comparing the generated text against the search results, the system can identify and flag unsupported or contradictory information, thereby reducing the likelihood of hallucinations.

The quality of the context provided to an LLM is a critical factor in preventing hallucinations. The UCSC at SemEval system demonstrates the importance of both context retrieval and prompt optimization [13]. This approach focuses on refining the information that is fed into the model. It employs advanced techniques to retrieve the most relevant documents from a knowledge base. In addition, it uses carefully crafted prompts to guide the model toward generating factually accurate text. This two-pronged approach—improving both the information provided and the instructions given—is shown to be effective at reducing hallucinations in question-answering tasks.

KnowHalu is another method that emphasizes the importance of external knowledge for verification [30]. This technique operates by first breaking down a generated statement into individual knowledge triplets (subject, predicate, object). It then queries a knowledge graph or other structured database to verify each of these triplets. If a triplet cannot be found in the knowledge base, it is flagged as a potential hallucination. This fine-grained, knowledge-centric approach allows for a more precise identification of factual errors than methods that evaluate the entire text at once. For the query "Who wrote the novel 1964 and when was it first published?", an LLM might hallucinate: "The novel 1964 was written by John Doe and published in 2010." KnowHalu decomposes this into sub-queries "author of 1964" and "publication year of 1964." It retrieves knowledge for each: perhaps finding no record of

John Doe as author and that the novel 1964 was published in 2004. By checking each fact (using text and knowledge graph triplets), KnowHalu flags both "John Doe" and "2010" as unverified, correctly identifying these as hallucinated information

A key challenge with LLMs is ensuring that their outputs are attributable to reliable sources. RARR (Retrofit Attribution using Research and Revision) is an automated method that revises an LLM's output to make it attributable to supporting evidence [11]. The process involves a cycle of "research" and "revision." In the research phase, the system searches for documents that support the claims made in the generated text. In the revision phase, it edits the text to ensure that it is fully supported by the retrieved documents, removing any claims that cannot be verified. This iterative process not only helps to eliminate hallucinations but also produces a final output that is directly traceable to its sources. Suppose the LLM outputs "Justice Ashok Kumar Mathur headed the 7th Central Pay Commission in India. It was created in 2014 and submitted its report in 2016." RARR's research phase would retrieve the true facts: e.g. an official source saying "The 7th Central Pay Commission (Chair: Justice A. K. Mathur) submitted its report on November 19, 2015." In the revision phase, RARR edits the text to match this evidence: "It was created in 2014 and submitted its report in 2015.". The revised answer now fully aligns with the source, eliminating the hallucination.

2.2.3 Proactive Detection and Mitigation

Another approach to hallucination mitigation is to predict and prevent errors before the model begins generating its response. FactCheckmate introduces a method for preemptive detection and intervention [4]. This technique is based on the hypothesis that an LLM's internal hidden states, produced after processing an input, contain signals that can predict whether the model will hallucinate. FactCheckmate trains a lightweight classifier to analyze these hidden states and predict the likelihood of an impending hallucination. If a high probability of hallucination is detected, the system intervenes by adjusting the hidden states to steer the model toward a more factual output. This preemptive approach is more efficient than many post-hoc verification methods, as it adds minimal overhead to the inference process.

2.3 Techniques for Hallucination Span Localization

2.3.1 Claim Decomposition and Verification

A common thread in many advanced verification methods is the principle of claim decomposition. This technique involves breaking down a complex, model-generated statement into a set of smaller, atomic claims. Each of these simple claims can then be verified individually against a knowledge source, which allows for a more precise and granular assessment of factuality. By verifying each atomic fact independently, these methods can more accurately identify specific points of hallucination within a larger text, preventing the entire output from being dismissed due to a single error.

Several systems implement this ”Decompose-Then-Verify” paradigm. For instance, HalluSearch employs a ”Factual Splitting” module that is explicitly prompted to break down a generated answer into discrete, standalone factual statements [1]. Each of these statements is then used to formulate a targeted query for a search-based verification step. Similarly, the FELM benchmark formalizes this process for evaluating summaries by first segmenting the text into fine-grained claims [8]. For each claim, it retrieves evidence from a source document and then uses a Natural Language Inference (NLI) model to classify the relationship as either ”support,” ”refute,” or ”not enough information.” This provides a systematic way to score the factuality of the summary on a claim-by-claim basis.

The KnowHalu framework takes a more intricate approach by combining query decomposition with multi-form knowledge verification [30]. It first performs a ”Step-wise Reasoning and Query Decomposition” on the initial prompt to create logical sub-queries. It then retrieves both unstructured text and structured knowledge triplets (subject, predicate, object) to verify the claims related to each sub-query. This method demonstrates a deeper application of decomposition, using it not only on the output but also on the input query to guide a more rigorous, multi-faceted verification process.

2.3.2 Alignment of External Evidence

Another critical technique in external verification is the direct alignment of retrieved evidence with the generated text. This process involves fetching relevant documents from a knowledge base and then systematically comparing them to the LLM’s output to identify unsupported or contradictory spans of text. The effectiveness of this method hinges on the quality of the retrieved context. The UCSC at SemEval system, for example, highlights that a combination of robust context retrieval and optimized prompting is essential for accurate hallucination detection [13]. Their approach emphasizes not only finding the most relevant external information but also structuring the prompt in a way that guides the model to align its output closely with the provided evidence, thereby minimizing factual drift.

2.3.3 Token-Level Signal Analysis

Another class of methods focuses on signals from the model itself, using token-level analysis to flag problematic words or phrases. These techniques leverage the internal mechanics of the model, such as token probabilities and attention weights, to identify potential hallucinations without relying on external knowledge. One such approach is based on Token Probability Analysis, which operates on the premise that low-probability tokens signal a model’s uncertainty [27]. When a model generates a token that it assigns a low likelihood, it may indicate a deviation from the factual knowledge learned during training, thus serving as a direct indicator of a potential hallucination. A different approach, Attention-Guided Self-Reflection, analyzes the model’s attention patterns across the input query [21]. This method separates the query into ”attentive” and ”non-attentive” parts based on attention scores and then checks the consistency of answers generated from these separate parts. A significant inconsistency suggests a hallucination, as the model's response may be unreliably based on parts of the query to which it paid little attention.

2.4 Summary

Recent research on hallucination evaluation in LLMs has produced a diverse set of benchmarks and methodologies, spanning multilingual, task-specific, and domain-specific contexts. General-purpose datasets such as Mu-SHROOM [14] and HaluEval [18] provide foundational resources, with the former emphasizing multilingual, token-level annotation and the latter offering a broad set of hallucination instances across multiple tasks. Task-specific benchmarks like FaithBench [6], HalluDial [23], LongHalQA [26], and TruthfulQA [20] focus on summarization, dialogue, long-context, and truthfulness evaluation respectively, while domain-specific datasets such as MedHalu [2] address high-stakes applications in healthcare.

Methodologies for hallucination detection can be grouped into model-internal/self-supervised approaches, retrieval-augmented and knowledge-based methods, and proactive detection techniques. Model-internal approaches leverage token probabilities [27], attention patterns [21], or generation consistency [24], while proactive systems like FactCheckmate [4] predict hallucinations before generation. In this work, our emphasis is on retrieval-augmented and knowledge-based methods, which integrate external sources to verify and ground LLM outputs. Notable examples include HalluSearch [1], which applies search-based claim verification to retrieval-augmented generation (RAG) outputs; the UCSC at SemEval system [13], which optimizes both document retrieval and prompting; KnowHalu [30], which decomposes model outputs into knowledge triplets for verification against structured databases; and RARR [11], which iteratively revises outputs to ensure all claims are supported by retrieved evidence.

Techniques for hallucination span localization—such as claim decomposition and verification [6, 30], alignment of external evidence [13], and token-level signal analysis [27, 21]—play a critical role in identifying the precise points of factual error. These approaches complement retrieval-augmented and knowledge-based pipelines by enabling fine-grained detection and targeted mitigation of hallucinations. Overall, the literature highlights the promise of retrieval-augmented and knowledge-centric strategies as effective means to improve the factual reliability of LLM outputs, which forms the central focus of this dissertation.

Chapter 3

Methodology

Given the background in hallucination detection, we develop an open source pipeline in order to identify incorrect spans in LLM generated answers. As mentioned previously in Chapter 2, it has been shown that there is promise in retrieval-augmented and knowledge-centric methods, and we use these to investigate the performance of open-source models v/s proprietary models in this task. We aim to answer some key research questions:

- **RQ1: Model Performance** - To what extent can open-source LLMs with smaller architectures and fewer parameters match the hallucination detection performance of paid state-of-the-art models (e.g., GPT-4, O3 Mini)?
- **RQ2: Context Retrieval** - How effective are open-source LLMs as context retrievers compared to proprietary retrieval-focused models (e.g., Perplexity Sonar Pro)?
- **RQ3: Prompt Structuring** - How does prompt complexity, such as using a single large prompt versus breaking it into multiple sub-task prompts, influence hallucination detection accuracy?
- **RQ4: Ensemble of LLM Systems** - Is it beneficial to take multiple LLM systems as an ensemble by treating them as a group of human annotators and taking a majority vote for predictions?
- **RQ5: Error Analysis and Mitigation** - What are the most frequent error types in hallucination detection when the question, LLM answer, and retrieved context are provided? Are there identifiable error patterns, and how can they be mitigated?

3.1 Problem Formulation and Core Task

The core task of this dissertation is **hallucination span detection**. Formally, given a query q and a model-generated response $R = \{t_1, t_2, ..., t_n\}$, where t_i is the i-th token, the objective is to identify all contiguous spans of tokens $S = \{t_j, ..., t_k\}$ within R that are factually incorrect, unfaithful to a provided source, or otherwise hallucinated. The model is also instructed to provide a confidence score for each span detected as a hallucination in order to perform evaluation against a gold standard.

3.2 Foundational Pipeline Design

To systematically address the research questions, we first establish a foundational detection pipeline. The pipeline is based on the "Decompose-Then-Verify" paradigm, inspired by the **HalluSearch** pipeline [1]. We establish a set of open source models to compare performances based on metrics and recency. As presented in Figure 3.1, our initial pipeline ingests an answer, query, and context based on which it identifies incorrect spans of characters in the answer. In particular, the model first breaks the answer into it's constituents claims or 'factual statements', and then it identifies whether each of these statements is true or not based on the context provided. This context is generated by an online context retrieval using the query, with a fallback to the model's internal knowledge base. Finally, it identifies the incorrect spans in the original answer.

3.2.1 Rationale for a Single-Stage, Context-Aware Approach

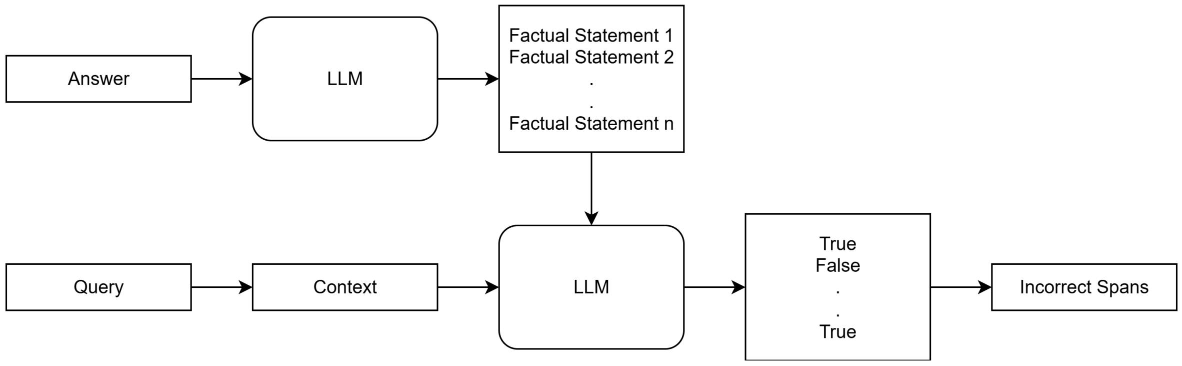

Figure 3.1: HalluSearch Pipeline

After an initial exploration of the foundational pipeline as shown in 3.1, we observe that there are significant reliability issues with the LLMs when dealing with intermediate steps like claim decomposition. The models also struggle to generate structured output despite being instruction tuned causing multiple responses to be corrupted and unable to be evaluated. This determined the need of a more robust approach with less interim complexity. Consequently we adopt the approach used by Huang et al. [13], where the context is retrieved beforehand using a task-specific retrieval model (perplexity sonar pro) and the query. They also implement a zero-shot example of the expected output structure which helps guide the models better, along with LangChain methods to ensure the output is of a specific schema.

As presented in Figure 3.2, the new pipeline consists of two key modules:

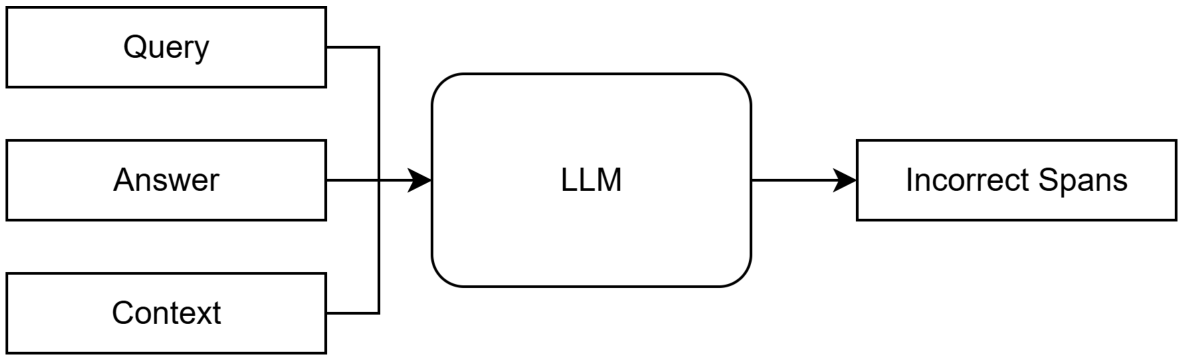

Figure 3.2: UCSC Pipeline

1. **Context Retrieval Module:** This module takes the original query given to the model and uses it to generate the context needed to answer it completely.

2. **Single-Prompt Verification Module:** This module feeds the original response and the retrieved context into an LLM using a single, comprehensive prompt. The prompt is engineered with a zero-shot example of the expected output structure to guide the model in identifying hallucinated spans directly.

3.3 Investigating Model Performance (RQ1)

First, we evaluate the extent to which smaller open-source LLMs can match the performance of larger proprietary models on the task of hallucination detection.

3.3.1 Selection of Open-Source Models

We select a diverse set of open-source, instruction-tuned models from families with good performance metrics in existing studies [3, 5] (Llama, Qwen), and that have been released recently (Phi-4, Mi:dm). Our intention is to create a representative and practical sample of the modern, publicly accessible AI landscape. The variants are selected based on public availability and parameter count. To truly contextualise the performance of these open-source models, it is essential to measure them against the established leaders in the field. We benchmark these against the top performing models in the Mu-SHROOM shared task [14], which all use proprietary models (GPT-4, o3 mini) that are only accessible using paid APIs.

3.3.2 Comparative Benchmarking Framework

All of the selected models are integrated into both our foundational pipeline as well as the improved version. For the context retrieval module, we also select 3 open-source models based on architecture size and knowledge-cutoff in order to maximize their performance in retrieving relevant contexts from their internal knowledge base. We compare the models across a standardized validation and test split of the dataset on two character-level metrics:

1. Intersection-over-union (IoU) of the characters in the incorrect spans identified by the systems v/s the ones provided in the dataset as gold standard

$$\begin{aligned} C_{\text{bin}} &= \left\{ c_i \;\middle|\; 0.5 < \sum_n \frac{1}{n} \mathbf{1}\{c_i \in C_n\} \right\} \\ \text{IoU} &= \left|\hat{C}_{\text{bin}} \cap C_{\text{bin}}\right| / \left|\hat{C}_{\text{bin}} \cup C_{\text{bin}}\right| \end{aligned} \tag{3.1}$$

   where C_{bin} is the set of binarized character-level annotations derived from the n different sets of annotations C_n, and $\hat{C}_{\text{bin}}$ is the set of characters that the system predicts as hallucinated. This acts as a binary marker for how well the model identifies each of the hallucinations that were marked by the annotators

2. Spearman's Rank Correlation Coefficient (Cor) of the probabilities of each character that was classified as a hallucination by the system with the empirical probabilities observed in the annotations. For a given datapoint of length k, we formally measure:

$$\begin{aligned} \Pr(c_i) &= \sum_{n=1}^{N} \frac{1}{n} \mathbf{1}\{c_i \in C_n\} \\ \mathbf{c} &= (\Pr(c_1), \ldots, \Pr(c_k)) \\ \hat{\mathbf{c}} &= (p(c_1 \mid \theta), \ldots, p(c_k \mid \theta)) \\ \rho &= \text{Spearman}(\mathbf{c}, \hat{\mathbf{c}}) \end{aligned} \tag{3.2}$$

   where $p(c_i \mid \theta)$ stands for the probability that character c_i is in a hallucinated span, as assigned by a given participating system, and $\Pr(c_i)$ is our empirical probability.

3.4 Evaluating Context Retrieval Strategies (RQ2)

This section details the approach for assessing the efficacy of open-source LLMs as context retrievers compared to specialized, proprietary retrieval systems.

3.4.1 LLM-as-Retriever vs. Specialized Systems

We compare 2 main different methods of context retrieval, the first being the context retrieved using an API-based retrieval-focused model (perplexity sonar pro) which is provided by Huang et al. [13]. This approach is characterized by its ability to access and process up-to-the-minute information from the web, treating context retrieval as a live search problem. In direct contrast to this, our second method leverages a model's internal, static knowledge base. We evaluate the pipeline outputs using this v/s using an open-source LLM as a context retriever, where it depends on its own internal knowledge cutoff to generate the relevant information to answer the query. This creates a clear test between a system that can look things up in real-time and one that must rely on what it was taught during its training phase. To ensure this is a fair and rigorous comparison, it is essential to apply a consistent measurement standard to both approaches. The evaluation is done using the same 2 pipeline level metrics that we mention in 3.3.2.

3.5 Analyzing Prompt Structuring and Complexity (RQ3)

We investigate how prompt complexity influences the hallucination detection accuracies by comparing the fundamental pipeline and the improved pipeline. The fundamental pipeline, inspired by Abdallah and El-Beltagy [1], uses a multi prompt method where there is: (1) a prompt to decompose the response into claims, (2) a prompt to verify each claim against the context, and (3) a prompt to aggregate the results. This process involves multiple transition states for the information, all of which needs to be in specific formats for the pipeline to succeed. We observe that open-source LLMs, especially with smaller architectures, struggle to output the same structured output with consistency.

We evaluate this against the improved system, inspired by Huang et al. [13], which uses a singular prompt to perform the main task. The context is retrieved separately using the query beforehand, and the incorrect span detection task is done entirely in a single step. The prompt contains few-shot examples of the expected input structure as well as the expected output structure which greatly improves the consistency of the final output structure.

3.6 Exploring Ensemble Systems for Enhanced Reliability (RQ4)

Another approach that we evaluate is ensemble based methods, where multiple systems composed of different combinations of prompt strategies and models are treated as individual annotators

3.6.1 Ensemble Construction and Voting Mechanism

To create a more reliable ground truth, we first take the outputs from a selected subset of the models tested in Section 3.3. This curated group of models is then conceptually treated as a "panel of annotators," allowing us to simulate how a group of human experts might reach a consensus. The core idea behind this ensemble approach is that the collective judgment of multiple diverse systems is often more accurate and robust than the prediction of any single model.

The final prediction for each character-level token is then generated using a majority voting mechanism, which aggregates the decisions from our panel. Specifically, for a token to be officially labelled as incorrect (i.e., part of a hallucination), the ratio of "annotators" (the models) that flag it as such must be greater than a specified threshold. This threshold-based system ensures that a character is only condemned as a hallucination when there is strong agreement among the various models in the panel.

3.6.2 Performance Analysis

We then compare the performance of the ensemble system against the performance of the best individual model within the panel of annotators. This specific comparison provides the most stringent test of the ensemble's value; it is not enough for the collective to simply outperform the average-performing model, but it must prove superior to the most capable single contributor. This analysis is conducted in order to determine whether there is any benefit to be gained from this method.

3.7 Qualitative Error Analysis and Mitigation Pathways (RQ5)

In the final part of our investigation, we carefully look at the typical problems that often impact how well the hallucination detection systems work within a particular split of the dataset. Based on this detailed review, we then describe and suggest several practical solutions to address these weaknesses and make the system more reliable and effective.

A randomly selected portion of erroneous predictions generated by the highest-performing model undergoes manual examination and classification in alignment with the established taxonomy. Drawing from the observed prevalence of each identified error category, we put forward and elaborate on specific remedial approaches, including enhanced prompting methodologies or optimized retrieval processes, aimed at informing subsequent investigations.

Chapter 4

Experimental Setup

We implemented an experimental framework in order to thoroughly evaluate all the methodologies mentioned in Chapter 3, and to address our work's core research questions in a complete manner. Inspired by existing frameworks, we modify them in order to accommodate open-source models, which are the backbone of our investigation. We detail the dataset, evaluation metrics, baseline models for comparison, and the specific configurations of the systems under investigation.

4.1 Dataset and Data Preparation

4.1.1 The Mu-SHROOM 2025 Dataset

The primary dataset used for all experiments is the English-language portion of the Mu-SHROOM 2025 dataset, from the SemEval-2025 Task 3 shared task [14]. This shared task is specifically designed to address the critical challenge of detecting and labelling spans of text corresponding to hallucinations. While the full dataset is multilingual and contains outputs from a variety of public-weight LLMs, our focus remains on the English subset to maintain a controlled experimental environment. Each data instance provides not only the raw LLM output as a string of characters but also the corresponding lists of tokens and logits, offering a rich basis for analysis.

This dataset is selected for its focus on hallucination in general-purpose LLMs and its provision of fine-grained, character-level span annotations, which are essential for the core task of this research. The structure of the Mu-SHROOM data aligns perfectly with our objectives, as the task's goal is to compute a probability of hallucination for every single character in an LLM's output. The availability of these precise, ground-truth annotations makes it an ideal resource for training and rigorously evaluating our detection models.

4.1.2 Data Splits

We adhere to the official data splits provided by the shared task organizers to ensure a fair and direct comparison with published results. The splits are as follows:

- **Validation Set:** Contains 50 annotated examples. This set is used for initial testing, prompt refinement, and debugging during the development phase.
- **Test Set:** Contains 154 annotated examples. This set is reserved for the final evaluation of our developed systems. The final performance metrics reported in this dissertation are calculated on this split.

4.2 Evaluation Metrics

To maintain consistency with the SemEval-2025 shared task, system performance is evaluated using the two official metrics:

1. **Intersection-over-Union (IoU):** This metric measures the accuracy of the predicted hallucination spans against the ground-truth spans. It is calculated at the character level to reward precise span detection. This is a binary metric and classifies each character as a 0 or 1 based on whether it matches the gold standard annotation spans.
2. **Spearman's Rank Correlation Coefficient (Corr):** This metric evaluates the system's ability to assign meaningful confidence scores to its predictions, assessing how well the ranking of predicted hallucination likelihoods correlates with the empirical probabilities from human annotations. This measures how well the systems capture the relative likelihood of the hallucination and not just a binarized representation of the decision.

4.3 Baselines for Comparison

We benchmarked the performance of our implemented systems against the top 5 performing models in the official leaderboard from the Mu-SHROOM 2025 shared task. All these top performing systems serve as high quality, state-of-the-art baselines with respect to the dataset we use. Notably, all the top 5 participants of the shared task utilized API-based proprietary models, such as GPT-4 and o3 mini in their systems. This sets a high standard for our experimentation into much smaller open-source models and their capabilities as hallucination detectors and context retrievers.

4.4 System Configurations for Research Questions

4.4.1 Models Under Investigation (RQ1)

To address our first research question regarding model performance, we select a range open source models, with our decision being based on their general performance within the community as well as the recency of the model release.

Table 4.1: LLMs selected for comparative performance analysis.

Category	**HuggingFace Model Identifier**
Performance	google/gemma-3-12b-it[1]
	openchat/openchat-3.5-0106[2]
	Qwen/Qwen2.5-14B-Instruct[3]
	tiiuae/Falcon3-10B-Instruct[4]
	meta-llama/Llama-3.1-8B-Instruct[5]
	mistralai/Mistral-Nemo-Instruct-2407[6]
Recency	*microsoft/phi-4*[7]
	openai/gpt-oss-20b[8]
	tiiuae/Falcon-H1-7B-Instruct[9]
	K-intelligence/Midm-2.0-Base-Instruct[10]

4.4.2 Context Retrieval Configuration (RQ2)

For comparing the different methods of context retrieval, we select open source LLMs based on their knowledge cutoff in order to match the efficiency of specialized retrieval models. For investigating context retrieval effectiveness, the following setups are used:

- **Online Search:** A basic implementation of the google search API in order to retrieve context related to the key claims in the answer, inspired by the methodology used by Abdallah and El-Beltagy [1] where they use a paid SERP API[11].

- **LLM-as-Retriever:** We experimented with 3 open source LLMs that are publicly available, which have a reasonable architecture size as well as a relatively recent knowledge cutoff - microsoft/phi-4, openai/gpt-oss-20b, and google/gemma-3-12b-it

- **Specialized Retrieval System:** Context retrieved by proprietary API-based model which is specialized for context retrieval tasks (perplexity sonar pro). In our experiments we used the contexts retrieved and provided by Huang et al. [13] for both the test split and the validation split of the dataset

[1]https://huggingface.co/google/gemma-3-12b-it
[2]https://huggingface.co/openchat/openchat-3.5-0106
[3]https://huggingface.co/Qwen/Qwen2.5-14B-Instruct
[4]https://huggingface.co/tiiuae/Falcon3-10B-Instruct
[5]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
[6]https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407
[7]https://huggingface.co/microsoft/phi-4
[8]https://huggingface.co/openai/gpt-oss-20b
[9]https://huggingface.co/tiiuae/Falcon-H1-7B-Instruct
[10]https://huggingface.co/K-intelligence/Midm-2.0-Base-Instruct
[11]https://serpapi.com/search-api

4.4.3 Prompt Design (RQ3)

To fully understand the different factors that prompt complexity brings into a hallucination detection task, two distinct prompting methods are tested. The single-prompt method is inspired from Huang et al. [13] and the multi-prompt method is inspired from Abdallah and El-Beltagy [1]. We modify it in order to replace any of their proprietary components with open-source, publicly available equivalents.

- **Single-Prompt Design:** The first part of the prompt defines the problem that needs to be solved and the steps should be taken. The prompt is as follows:

```
Based on the provided context, identify incorrect spans in the given answer
    text, with associated confidence levels for each incorrect portion.

You will be provided a context with a question and its corresponding answer.
    Your task is to identify any specific parts of the answer that describes
    facts that are not supported by the context. If there are multiple
    incorrect segments, report each one separately. Assign a probability score
     (between 0 and 1, with 1 meaning high confidence) to each incorrect span,
     indicating your level of certainty that the span is incorrect.

# Steps
1. **Read the Context**: Carefully read the provided context.
2. **Analyze the Answer**: Carefully evaluate the given answer for accuracy
    regarding the question and the context.
3. **Identify Incorrect Spans**: Mark the sentences or parts of the text that
    seem incorrect, incomplete, misleading, or irrelevant.
4. **Assign Probability**: Assign a confidence score for each answerspan you
    identify as incorrect:
   - A higher score indicates greater confidence that an identified segment is
      incorrect.
   - Provide a score for each span between 0 and 1.
```

  The second part of of the prompt gives an example of what the output structure looks like. This has great benefits, especially in LLMs with fewer number of parameters, towards ensuring that the model is consistent with the output structure and doesn't fail in an iterative pipeline. It is as follows:

~~~
# Output Format
The output should be in JSONL format as shown below:
```json
{{
  "incorrect_spans": [
    {{
      "text": "[identified incorrect span]",
      "probability": [confidence_score]
    }},
    {{
~~~

```
      "text": "[another identified incorrect span]",
      "probability": [confidence_score]
    }}
  ]
}}
‘‘‘
```

The final part of the prompt adds and example and provides some additional instructions. It is as follows:

```
If no incorrect spans are identified, return an empty list: ‘"incorrect_spans
    ": []‘.

# Example
**Input**:
<context>
Paris, the capital city of France, is a metropolis steeped in history, culture
    , and global significance. This comprehensive analysis will delve into the
     city’s current status, basic information, and historical importance,
    providing a thorough understanding of why Paris is not just the capital of
     France, but also one of the world’s most influential cities.
</context>

<question>
What is the capital of France?
</question>

<answer>
The capital of Farnce is Berlin.
</answer>

**Output**:
‘‘‘json
{{
  "incorrect_spans": [
    {{
      "text": "Berlin",
      "probability": 0.99
    }},
    {{
      "text": "Farnce",
      "probability": 0.99
    }}
  ]
}}
‘‘‘

# Notes
```

```
- Ensure that the probability reflects your confidence. If unsure about the
    degree of incorrectness, use a lower value.
- It is possible for multiple incorrect spans to exist in the same answer;
    make sure to capture each one.
- If the answer is fully correct, return '"incorrect_spans": []'.
- Try to identify the spans as short as possible.
- The spans should appear in the same order as they appear in the original
    answer.
```

- **Multi-Prompt Design:** The first prompt instructs the model to break the answer into factual statements, also known as claim decomposition, as shown below:

```
You are a multi-lingual text analysis assistant. The provided text is in the {
    data_point['lang']} language.

Text to analyze:
"""{text}"""

**Task**:
1. Split this text into atomic, independent statements or claims.
   - This includes short words or phrases that might carry a claim (e.g., "Yes
       ", "No", "Certainly", "Indeed", in English

   - If a short phrase is attached to a longer statement but clearly provides
       its own claim, separate it.

2. Output a JSON array of objects. Each object should have:
   - "factual_statement_i": the i-th extracted statement.
   - "original_substring_i": the exact substring from the text with punctuation
        as in the original provided text .

3. Return only valid JSON. No extra commentary or markdown or backticks.

IMPORTANT:
- Retrieve the "original_substring_i" axactly as is in the original text,
    including purely leading/trailing punctuation,
  spacing, and capitalization. DO NOT modify, rephrase or alter the original
      text in anyway.
  Pay attention to skip characters always consider skip characters.
- Do not merge short claim-words with other sentences unless they are
    obviously part of the same statement.
- Number statements sequentially using "i", i is numeric in "
    factual_statement_i" and "original_substring_i" must be integer indices
    (1,2,3,...) such as original_substring_1, factual_statement_2,
    original_substring_3, ...
```

In the next step the LLM is instructed to take each of the factual statements or claims from the previous step and determine whether they are true or false depending on the context provided. As shown below:

```
 You are given the following context:

\"\"\"{context}\"\"\"

Below is a JSON array of factual statements extracted from a given text.
Each object has keys like "factual_statement_i" and "original_substring_i".

Your task:
1. **Reason step by step internally** (chain-of-thought) to analyze each
    statement:
   - Compare each factual statement with the context.
   - Determine whether it is True or False.
   - If False, identify the smallest contradictory spans (e.g., incorrect names
       , dates, events, etc.).
2. Produce a final JSON array where each element is:
   {{
     "original_substring_i": "...",
     "verdict": true or false,
     "hallucination": []
   }}
   - Ensure "i" matches the index from the original object.
   - "verdict" must be boolean (true/false).
   - If the statement is False, "hallucination" must list minimal contradictory
       parts.

**IMPORTANT GUIDELINES**:
- Only output valid JSON (no additional comments or explanations or triple
    backticks).).
- **Highlight minimal contradictory spans** for False statements:
  - Examples: locations, entities, events, or dates.
  - Ignore purely leading/trailing punctuation unless part of the claim.
- Treat incomplete or partially incorrect statements as False.
- Ensure that repeated or redundant parts are included in "hallucination".
- The queries, statements, and context can be in any language.

**CHAIN-OF-THOUGHT REASONING**:
- First, analyze internally for each statement and reason step by step.
- NEVER include this reasoning in the output. ONLY present the final JSON
    array.

Here is the JSON array of factual statements:

{facts_json}
```

```
Based on your reasoning, return only the final valid JSON.
```

Finally, using the output from the verification stage, the last stage of the pipeline is to get the span labels from the original answer

4.4.4 Ensemble System Configuration (RQ4)

We select the best performing models from the best performing pipeline and add them to the ensemble. The core principle of this ensemble approach is that aggregating predictions from multiple diverse systems typically yields more accurate and stable results than relying on a single model's output. The final prediction for each character-level token is computed using a majority voting algorithm that consolidates decisions from the panel. Specifically, for a token to be classified as incorrect (i.e., part of a hallucination), the proportion of "annotators" (models) labeling it as such must surpass a predefined threshold. This threshold-based mechanism ensures that a character is flagged as a hallucination only when there is strong agreement across the models in the panel. The selected models are: gemma-3-12b-it, Qwen2.5-14B-Instruct, and Midm-2.0-Base-Instruct.

4.5 Our Modifications

As part of our work we modified the pipelines used by 2 of the teams (ranked 2nd and 4th in the shared task), by replacing their proprietary components with open source alternative and changing the rest of their pipeline to work with these modifications.

4.5.1 Hallusearch

The Hallusearch pipeline uses a claim decomposition and verification based methodology that prompts the LLM at multiple stages in order to detect incorrect spans in the given answer based on the context and the query. The answer is first broken down into a set of claims, and each of those claims is verified separately before the model returns the final set of detected hallucinations.

The original context retrieval module used a SERP API call in order to get the context by retrieving the contents of the top webpage that matched the query and then parsing it's contents in the context. We first replaced this with a more lightweight search engine call using the Bing API and the DuckDuckGo API. We observed that these methods often tend to return irrelevant results or the webpage is not easily accessible or difficult to parse. In order to overcome these issues we replaced it with our open-source LLM context retriever and instead of retrieving context on the fly using web search APIs, we retrieve all the contexts beforehand and then just pass the relevant context into the pipeline for the particular query-answer pair.

The LLM used in the original pipeline was GPT-4, and it was called using the proprietary OpenAI API. We replaced this with a generic call to a HuggingFace Transformers model, so the module can now utilize any of the publicly available models from the HuggingFace library as long as it supports text generation capabilities. We used appropriate wrappers around the model in order to make sure that it worked in a similar chat template manner that GPT-4 also uses so that we could accurately compare the effect of only changing the model on the final evaluation metrics.

4.5.2 UCSC

The UCSC pipeline implements a simpler mechanism than Hallusearch, and the main hallucination detection task is done using only a single prompt. They take advantage of modern LLMs ability to adapt well when given a pattern in which to respond. They provide few-shot examples of the expected input and output structures which guide the model to produce consistent structured responses.

They utilized perplexity sonar pro a specialized proprietary model that needs to be accessed via API for the context retrieval. To get the relevant context they passed the corresponding query to this model and asked it to retrieve the relevant context in order to answer it. In order to maintain our approach as completely open-source, we developed our own context retrieval system using a few selected open-source LLMs. We designed the following prompt:

```
"You are an assistant who helps answer the user in a detailed manner. "
"Return all the contextual information needed to answer the given query.\n\n"
"Only Return the context nothing else.\n\n"

"Example:\n"
"Query: What is the significance of the Brown v. Board of Education case?\n"
"Answer: Brown v. Board of Education of Topeka (1954) is a landmark decision of
    the U.S. Supreme Court that declared racial segregation in public schools
    unconstitutional. "
"The Court unanimously held that 'separate but equal' educational facilities for
     racial minorities and whites were inherently unequal, violating the Equal
    Protection Clause of the Fourteenth Amendment. "
"This case overturned the precedent set by Plessy v. Ferguson (1896) and was a
    major victory for the American civil rights movement. It laid the legal
    foundation for dismantling segregation across all areas of public life in
    the United States.\n\n"

f"Now answer the following query in a similar manner.\n"
f"Query: {query}\n"
```

Using this prompt we retrieved contexts for both the validation and test data splits, with the following open-source LLMs: gemma-3, phi-4, and gpt-oss-20b. These models were selected due to combination of the recency of their knowledge cutoff, their performance metrics, and their architecture size.

The UCSC pipeline also uses LangChain around their LLMs in order to utilise the structured output feature that it provides, which asserts the LLM outputs to a specific predefined output structure. Since this does not work out of the box with HuggingFace chat pipelines, we created a new function that cleaned the LLM output and also asserted it to a specific pydantic format in order to emulate the behaviour of the structure output function of LangChain. We identified the different output patterns of the open-source LLMs and used that to isolate the final answer only, and then we used regex based patterns to clean up the model response and make sure it was in the structured format that was needed by the pipeline in order to execute further steps.

4.6 Implementation Details

- **Hardware:** All experiments were conducted on a machine equipped with one NVIDIA A100 GPU with 80GB of VRAM
- The implementation relies mainly on Python , PyTorch, and Hugging Face Transformers. The structured output is managed using LangChain and Pydantic

4.7 Error Analysis Protocol (RQ5)

For the qualitative error analysis, we picked a sample of incorrect predictions from the validation split of the dataset, where the errors are prevalent throughout systems with different prompts and models. We took the most common errors, and manually reviewed and categorized them into taxonomies. The aim of identifying these errors and classifying them into these specific categories is to suggest methods that can be developed in order to best mitigate the errors of each category.

Chapter 5

Results

This chapter presents the empirical results of the experiments detailed in the previous chapter. The primary objective is to analyze the performance of the developed systems for hallucination span detection and to provide quantitative answers to the research questions guiding this dissertation. We begin by reporting on the outcomes of the initial multi-stage pipeline, which highlights its practical limitations. We then present a comprehensive analysis of the refined single-stage pipeline, evaluating the impact of different models, context retrievers, and prompt structures on detection accuracy. Finally, we present the findings from our ensemble system and a qualitative error analysis.

As outlined in the methodology, our initial approach was based on a multi-stage, "Decompose-Then-Verify" pipeline. The experiments with this pipeline primarily served to identify its viability with open-source models. The results are characterized by high failure rates at intermediate stages, which inevitably lead to poor evaluation metrics at the final stage.

The consistently high failure rates, as shown in Table 5.1, rendered this approach impractical for reliable hallucination detection and necessitated the development of the more robust single-stage pipeline. If we refer to Table A.1 (Appendix), we can see clearly that for some models like FalconH1 and gpt-oss the failure rates in both claim decomposition as well as JSON structuring were as high as 100%, i.e. it wasn't able to run the pipeline completely for any of the examples in the test or validation data splits. Even other models had atleast 8-10% error rates, and when the size of the dataset is minimal (50 validation examples and 150 test examples), even these error rates make it difficult to actually evaluate the final pipeline peformance without factoring in the contribution of the failure rate.

From Table 5.2, we can compare performance across contexts and see which models generalize well between validation and test conditions. We show the results from the top 3 best performing models across contexts and data splits. The best results by an open source model are achieved by the gemma3 model. It achieves an IoU score of 43% on the test data with proprietary context as well as open source context generated by the same gemma3 model separately.

The following sections detail the performance of the single-stage pipeline. A baseline configuration is first established, followed by a series of experiments designed to address each

Model	Context	Test		Val	
		decomp	JSON	decomp	JSON
Qwen2.5-14B-Instruct	phi-4	0.20	0.14	0.12	0.12
	perplexity-sonar-pro	0.20	0.14	0.14	0.10
	gemma3	0.17	0.14	0.14	0.14
	gpt-oss-20b	0.17	0.14	0.14	0.16
Midm-2.0-Base-Instruct	phi-4	0.16	0.21	0.14	0.28
	perplexity-sonar-pro	0.15	0.31	0.08	0.28
	gemma3	0.16	0.25	0.12	0.18
	gpt-oss-20b	0.16	0.24	0.10	0.24
gemma-3-12b-it	phi-4	0.17	0.20	0.12	0.22
	perplexity-sonar-pro	0.17	0.18	0.12	0.18
	gemma3	0.18	0.19	0.12	0.18
	gpt-oss-20b	0.18	0.23	0.12	0.26

Table 5.1: Claim decomposition and JSON failure rates for Top-3 models in multi-stage pipeline

Model	Context	Test		Val	
		IoU	Corr	IoU	Corr
gemma-3-12b-it	gpt-oss-20b	0.40	0.39	0.40	0.38
	phi-4	0.39	0.39	0.40	0.43
	perplexity-sonar-pro	0.43	0.43	0.40	0.44
	gemma3	0.43	0.39	0.43	0.38
Qwen2.5-14B-Instruct	perplexity-sonar-pro	0.43	0.40	0.39	0.40
	phi-4	0.39	0.40	0.40	0.39
	gemma3	0.41	0.40	0.41	0.39
	gpt-oss-20b	0.39	0.37	0.43	0.44
Midm-2.0-Base-Instruct	gpt-oss-20b	0.36	0.36	0.31	0.33
	perplexity-sonar-pro	0.34	0.34	0.34	0.31
	phi-4	0.37	0.34	0.35	0.33
	gemma3	0.39	0.35	0.40	0.34

Table 5.2: IoU and correlation metrics for Top-3 models in multi-stage pipeline

research question.

5.1 RQ1: Comparative Performance of LLMs

This section evaluates the performance of pipelines submitted to the same shared task and evaluated on the same data versus the prformance of our pipelines (both the single-stage and multi-stage). We report the performance of all the open-source models that we tested in

Model	**IoU**	**Corr**
gemma-3-12b-it	0.53	0.50
Falcon-H1-7B-Instruct	0.49	0.45
Midm-2.0-Base-Instruct	0.49	0.40
Qwen2.5-14B-Instruct	0.48	0.42
Llama-3.1-8B-Instruct	0.46	0.43
gpt-oss-20b	0.45	0.38
phi-4	0.44	0.37
openchat-3.5-0106	0.44	0.36
Mistral-Nemo-Instruct-2407	0.44	0.38
Falcon3-10B-Instruct	0.43	0.34

Table 5.3: Average IoU and correlation metrics per model on the test data split using the UCSC-based pipeline

Table 5.3. On comparing purely on the basis of the LLM used, regardless of the context, on the test data split with the UCSC-based pipeline, we see that the top performing model is gemma3, with FalconH1, Qwen2.5 and Mi:dm2.0, close behind. It is observed that gemma3 has a slight edge over the others.

As shown in Table 5.4, the best performing pipeline using an open-source model, UCSC (gemma3 detection only) comes 5th on the leaderboard for hallucination detection in English. While the top performing pipeline, iai_MSU, achieves 65% IoU on the test data split, which shows that larger proprietary models still have a slight edge, our best scoring pipeline comes close to pipelines like Atlantis which use proprietary models like GPT-4 to achiever their results.

5.2 RQ2: Impact of Context Retrieval Method

We experimented with 3 open-source LLMs to retrieve context for all the data point using the original query that was used to generate the answer. Refer to Section 4.5.2 for the exact prompt we used to get these contexts. From Table 5.4 we can see that the open source context pipeline achieves an IoU of 52% on the test data split and Corr score of 50% which is only a 2% reduction in performance compared to the pipeline using the proprietary context.

Pipeline	**Test**		**Val**	
	IoU	**Corr**	**IoU**	**Corr**
iai_MSU	**0.65**	**0.62**	–	–
UCSC (vanilla)	0.61	0.54	0.57	0.55
ATLANTIS	0.57	–	–	–
HalluSearch (vanilla)	0.56	0.53	–	–
UCSC (gemma3 detection only)	0.54	0.53	0.50	0.53
ccnu	0.54	0.55	–	–
UCSC (gemma3 detection + context)	0.52	0.50	0.47	0.46
HalluSearch (gemma3 detection only)	0.43	0.42	0.40	0.44

Table 5.4: IoU and correlation metrics for best performing models in the shared task compared to our best performing baseline open-source pipelines.

On comparing downstream task evaluation metrics it is evident that the context from open-source LLMs' internal knowledge base is not of inferior quality to the one retrieved using a specialised API-based model (perplexity sonar pro).

5.2.1 Baseline Performance of the Foundational System

Model	**Context**	**Test**		**Val**	
		IoU	**Corr**	**IoU**	**Corr**
gemma-3-12b-it	gpt-oss-20b	–	–	0.44	0.48
	perplexity-sonar-pro	0.54	0.53	0.50	0.53
	phi-4	0.52	0.48	0.48	0.51
	gemma3	0.52	0.50	0.47	0.46
Qwen2.5-14B-Instruct	gpt-oss-20b	0.45	0.40	0.41	0.39
	perplexity-sonar-pro	0.50	0.43	0.43	0.42
	phi-4	0.48	0.40	0.44	0.49
	gemma3	0.50	0.46	0.48	0.51
Midm-2.0-Base-Instruct	gpt-oss-20b	0.47	0.38	0.43	0.43
	perplexity-sonar-pro	0.53	0.44	0.48	0.46
	phi-4	0.49	0.40	0.45	0.40
	gemma3	0.46	0.38	0.47	0.42

Table 5.5: IoU and correlation metrics for Top-3 models in single-stage pipeline

We can see from Table 5.5 that the best perforkming model was still gemma3, with an IoU score of 54.7% on the test data split with the proprietary context and a score of 52.32% on the test data split with open-source context from gemma3. There are still some cases where the pipeline fails in between because of the models inability to produce structured output but it is much less evident in this pipeline and only occurs in specific model-context pairings (for example the gemma3 and gpt-oss combination as can be seen in Table 5.5 above).

Context	**IoU**	**Corr**
perplexity-sonar-pro	0.48	0.42
phi-4	0.47	0.40
gemma3	0.47	0.41
gpt-oss-20b	0.44	0.38

Table 5.6: Average IoU and correlation metrics per context on the test data split using the UCSC-based pipeline

We also report in Table 5.6, the average accuracy of all 10 open source models when using that specific context, on the test data split with the second single-stage pipeline. There is a negligilbe difference in the performance of models using the perplexity sonar pro context and the rest, with there being only a 1% reduction in the IoU when using either the phi-4 or the gemma3 contexts.

5.3 RQ3: Influence of Prompt Structure

This section investigates the performance difference between a single, comprehensive prompt and a multi-step, chain-of-thought style prompting approach.

Prompt Structure	**Multi**			**Single**	
Model + Context	**IoU**	**Corr**	**Model + Context**	**IoU**	**Corr**
gemma3 + gemma3	0.43	0.39	gemma3 + perplexity-sonar-pro	0.54	0.53
gemma3 + perplexity-sonar-pro	0.43	0.43	Midm-2.0 + perplexity-sonar-pro	0.53	0.44
Qwen2.5 + perplexity-sonar-pro	0.43	0.40	Falcon-H1 + perplexity-sonar-pro	0.53	0.47
phi-4 + gemma3	0.43	0.40	gemma3 + phi-4	0.52	0.48
phi-4 + phi-4	0.42	0.43	Qwen2.5 + gemma3	0.50	0.46
Top-5 Average	**0.43**	**0.41**	**Top-5 Average**	**0.52**	**0.48**

Table 5.7: Comparison of prompt design by taking the average of the Top-5 performing model-context pairs

In Table 5.7 we show the average performance of the top 5 pipelines with the single-stage and multi-stage prompt. It is evident that the single-stage prompt structure works much better with a 10% improvement in IoU across the top 5 performers.

Prompt Improvement

We use the single-stage pipeline for our further experiments, where we modify the prompt in order to incorporate more information for the model to guise it's response. Upon adding few-shot examples to the prompt and also instructing the model to provide the reasoning behind each hallucination prediction, we improve the results to 57% IoU and 52% Corr on the test dataset. The best performing pipeline uses the Qwen2.5 model which responds slightly better to these changes, with the gemma3 model only performing 1% worse, and both being better than the previous results we got without these changes.

5.4 RQ4: Efficacy of the Ensemble System

We evaluate whether combining the predictions of multiple LLMs through a majority vote can lead to improved performance. We create 2 types of ensembles and see which one responds the best.

Model Based Ensemble

The models were selected based on their performance using the modified prompt on the test split of the dataset. The top-3 models were gemma3, Qwen2.5, and Mi:dm2.0. The ensemble strategy treated each of the models as an individual annotator and all the soft labels from all the models were collected. For each character in the answer, it was calculated how many models out of the 3 predicted it as a hallucination, and if the ratio of models that voted it as an incorrect span crossed a certain threshold then it was treated as hallucination and the predicted probability is the ratio of models that voted positively.

- Highest performing individual model(Qwen 2.5) – IoU: 0.57, Corr: 0.52
- Ensemble – IoU: 0.56, Corr: 0.52

Hence, the ensemble did not have any performance improvement over the best performing individual model, and in fact IoU drops by 1%.

Temperature Based Ensemble

Using the same ensemble majority voting strategy as above, this time it was created using 3 different temperatures with the best performing model pipeline (Qwen 2.5) - 0.05, 0.1, and 0.2. We did not test other temperatures since value of 0.4 had lower evaluation metrics in the individual model performance.

- Highest performing individual model (0.1 temp) – IoU: 0.57, Corr: 0.53
- Ensemble – IoU: 0.58, Corr: 0.53

While there is no appreciable performance improvement on using an ensemble in this method, it was noted that the ensemble performance did reach 58%, enough to place it 3rd on the shared task leaderboard.

5.5 RQ5: Qualitative Error Analysis

To understand the limitations of the best-performing system, we conducted a manual analysis of incorrect predictions. We considered all the 50 examples in the val dataset and if the same error occured across multiple model pipelines, then we categorised it into a taxonomy. Out of the 50 data points in the val split, 31 had a prevalent error that could be categorised under one of our taxonomies as follows:

- **Contextual Misinterpretation:** The correct evidence is retrieved, but the LLM misinterprets its meaning. This error occurs when the system retrieves accurate supporting information, yet the LLM fails to correctly interpret the contextual implications, leading to erroneous predictions. Such failures are often associated with ambiguous or complex contexts, and mitigation may involve including diverse contextual examples in the few-shot examples to improve interpretive accuracy.

- **Incorrect Context:** The retrieval module fails to find the necessary evidence to verify or refute a claim. This category encompasses instances where the retrieval component omits critical evidence due to limitations in search algorithms or data sources, resulting in insufficient contextual support for claim validation. Addressing this may require advancements in retrieval mechanisms, such as combining contexts, improving the query, or expanded knowledge bases, to ensure comprehensive evidence acquisition.

- **Semantic Nuance Failure:** The model incorrectly flags a statement that is semantically equivalent but lexically different from the source text. This failure arises when the model overly emphasizes lexical variations while overlooking semantic equivalence, leading to false positives in error detection. Potential solutions include integrating advanced natural language processing methods to better capture semantic similarities and reduce sensitivity to superficial differences.

Problem	**Prevalence**
Context Misinterpretation	38.71%
Incorrect Context	32.26%
Semantic Failure	29.03%

Table 5.8: Distribution of problem types.

As can be seen in Table 5.8, there are 3 main categories in the taxonomy into which the errors are classified:

- Context Misinterpretation: The are cases when the context implies that certain parts of the answer are hallucinated but does not explicitly debunk them. Since the model is instructed to only use the context to determine hallucinations, it is unable to get this implied information and misses out on marking these as hallucination. For example in the case below -

```
Q : Where do Hamilton Academical F.C.'s play their home matches?
A :  Hamilton Academical Football Club plays their home matches at New Douglas
    Park, which is located in Hamilton, South Lanarkshire, Scotland. The
    stadium has a capacity of around 6,000 and has been the home ground of
    Hamilton Academical since 1878, making it one of the oldest football
    grounds in the world still in use by a single club.
C: Hamilton Academical F.C. play their home matches at New Douglas Park,
    located in Hamilton, South Lanarkshire, Scotland[1][2][7].

Some key details about New Douglas Park:

- It opened in 2001 as the successor to the club's former ground, Douglas Park
    [2][7].....
```

While it is not explicitly mentioned that Douglas Park is one of the oldest stadiums in the world, the context clearly says that it was created in 2001 and it had a predecessor. This clearly implies that it is not one of the oldest footbal grounds in the world which the model fails to consider

- Incorrect Context: This is when the context retriever fails to retrieve the correct context for the particular query and answer. It is either factually incorrect or lacks critical information needed to answer the query. For example in the situation below:

```
Q : In which city were the 26th biathlon world championships held?
A :  The 26th Biathlon World Championships were held in Rasnov, Romania. The
    event took place from February 23 to March 5, 2013....
C : The 26th Biathlon World Championships were held in Ruhpolding, Germany.
    The championships took place from February 16 to February 24, 2002.....
```

In actuality, the 26th biathlon world championships were held in Lahti, Finland in 1991. The context retrieved provides the incorrect location as well as dates, while the model also hallucinated a completely different set of location and dates.

- Semantic Failure: These are the cases where the model does take into account the semantic dependencies of entities within the answer, and while it marks certain parts of the answer as hallucinations, it fails to understand that if a premise is false then the conclusions that depend on it must also be false; For instance with the case below -

```
Q : What are the four styles of Zhejiang cuisine?
A : Zhejing cuisine is known for its unique flavors and cooking techniques.
    The four main styles are: 1) Jiangnan style, which is characterized by the
     use of rice and seafood; 2) Hangzhou style....
C : Zhejiang cuisine, also known as Yue cuisine, is one of the Eight Culinary
    Traditions of Chinese cuisine. It is characterized by its emphasis on
    fresh ingredients, light seasoning, and elegant presentation. The cuisine
    is closely tied to the region's geography, with abundant seafood and
    produce. Zhejiang cuisine is traditionally divided into four main styles,
    each representing a different regional area and culinary approach:
```

```
1. **Hangzhou Cuisine ....
```

The context mentions 4 cuisines, Hangzhou, Shaoxing, Ningbo, and Wenzhou, while the answer contains an incorrect example of Jiangnan style. While the pipeline is correctly able to detect that Jiangnan itself is a hallucination based on the context, the entire description of the cuisine in the answer is not marked as a hallucination. The model fails to understand that since Jiangnan itself is a hallucination then the entire description of it being chracterized by the use of rice and seafood is also a hallucination.

Chapter 6

Conclusions

6.1 Summary of Results

This study rigorously evaluated our proposed framework for detecting hallucinated spans in LLM outputs, yielding insights into its effectiveness and areas for improvement. The following subsections discuss the key findings, their implications for hallucination detection, and their alignment with existing literature. These observations provide a foundation for understanding the strengths and limitations of open-source LLMs in hallucination mitigation and context retrieval, while identifying practical strategies for enhancing system performance.

6.1.1 Performance of Open-Source LLMs in Hallucination Detection

Our evaluation demonstrates that open-source LLMs, such as Gemma and Qwen, achieve performance levels comparable to proprietary models in identifying hallucinated spans. This finding aligns with recent studies that highlight the growing capability of open-source models in natural language tasks. However, limitations persist, including sensitivity to ambiguous or out-of-distribution inputs, where open-source models occasionally misclassified factual errors as correct due to limited training data diversity. These results suggest that open-source LLMs are viable for cost-effective hallucination detection but require further fine-tuning to handle edge cases, particularly in domains like medical or legal text generation where precision is critical.

6.1.2 Effectiveness of LLMs as Context Retrieval Engines

The experiments revealed that open-source LLMs serve as robust context retrieval engines, with performance metrics only 1-2% lower than those of proprietary models like GPT-4. This marginal gap indicates that open-source LLMs can effectively leverage internal knowledge bases to ground responses, reducing reliance on external APIs. However, performance slightly degraded for queries requiring real-time or highly specialized knowledge, underscoring the need for hybrid retrieval systems that integrate external sources like Wikidata.

6.1.3 Impact of Prompt Complexity on System Performance

Our analysis of prompt engineering strategies showed that complex, multi-stage prompts introduced unnecessary layers of processing, leading to a higher failure rate in hallucination detection. Single, well-crafted prompts consistently outperformed their complex counterparts, achieving up to 5-10% higher IoU scores the MuSHROOM dataset. This suggests that simpler prompts reduce cognitive overhead for LLMs, enabling more reliable discrimination between factual and hallucinated outputs. For example, a single prompt with clear instructions to "identify and flag incorrect spans with reasoning" yielded fewer false positives than chained prompts requiring iterative validation. Simplifying prompt design could thus enhance scalability and ease of deployment in real-world applications.

6.1.4 Benefits and Limitations of Ensemble Techniques

Ensembling based on temperature variation of the best-performing model yielded modest improvements in hallucination detection, with an average IoU increase of 1% compared to single-model baselines. By sampling outputs at different temperatures and applying majority voting, the ensemble reduced model variance and mitigated overfitting to specific hallucination types. However, the gains were smaller than anticipated, likely due to the homogeneity of the base model's outputs across temperature settings. This aligns with ensemble learning principles, which suggest that diversity among ensemble members is critical for significant performance boosts [10]. Future ensembling strategies could explore diverse architectures or confidence-weighted voting to further enhance robustness, particularly for cross-domain tasks where variance remains a challenge.

6.1.5 Implications and Broader Context

Collectively, these findings underscore the potential of open-source LLMs as cost-effective tools for hallucination detection and context retrieval, with performance approaching that of proprietary systems. The success of single-prompt strategies highlights the importance of simplicity in system design, while the modest benefits of ensembling suggest a need for more diverse aggregation methods. These insights contribute to the growing body of research on trustworthy AI, offering practical guidance for developing reliable LLMs in applications requiring high factual accuracy. However, limitations such as sensitivity to out-of-distribution data and marginal ensemble gains necessitate further exploration, as outlined in the future research directions.

6.2 Future Research Directions

Although this dissertation has advanced the understanding of hallucination detection and mitigation, its findings also highlight several limitations and opportunities for enhancement. The empirical results provide a foundation for future work aimed at improving the accuracy, scalability, and generalizability of these methods. The following subsections delineate key extensions to this research, emphasizing rigorous experimental validation and integration with emerging techniques in natural language processing.

6.2.1 Enhancing Prompt Engineering with Comprehensive Few-Shot Examples

A critical avenue for future work involves refining the prompting strategies employed in the hallucination detection pipeline. The current study was limited to zero-shot and basic few-shot prompts, which demonstrated only moderate effectiveness. Future research should incorporate a more comprehensive set of few-shot examples that systematically cover established hallucination taxonomies, such as factual inaccuracies, entity fabrications, and contextual inconsistencies, drawing from annotated datasets like HaluEval. For instance, prompts could include diverse exemplars for each hallucination type to guide the model toward more nuanced discrimination. Experimental validation would involve ablation studies comparing prompt variants on benchmarks like TruthfulQA, measuring improvements in precision and recall. This approach is hypothesized to mitigate the prompt sensitivity observed in our results and could yield substantial improvements in detection rates.

6.2.2 Incorporating an Output Refinement Mechanism with Explicit Reasoning

Building upon the detection phase, future investigations could introduce a multi-stage refinement mechanism to actively correct identified hallucinations. The present framework flags incorrect spans but does not iteratively improve them. An extension could add a dedicated refinement step where the model generates reasoned revisions for predicted erroneous segments. For example, utilizing chain-of-thought prompting, the LLM could articulate step-by-step justifications before proposing a corrected output. This could be implemented as a self-correction loop, iterating until a non-hallucinated response is achieved. To evaluate efficacy, studies could employ human-in-the-loop annotations or automated metrics such as ROUGE for factual alignment and BARTScore for semantic fidelity. Such a mechanism would address a key limitation of the current work, fostering more reliable outputs for high-stakes applications.

6.2.3 Experimenting with Advanced Ensemble Voting Strategies

The ensemble method in this dissertation relied on simple majority voting, which, while effective at reducing individual model biases, leaves room for optimization. Future work should explore more sophisticated voting methodologies, such as confidence-weighted voting, where each member's contribution is scaled by its output probability or an entropy-based confidence score. Additionally, techniques like stacking or Bayesian averaging could be tested to further minimize model variance and enhance robustness. Comparative experiments on datasets like HotpotQA could quantify variance reduction via metrics such as the standard deviation in IoU scores across multiple runs. This direction aligns with established ensemble learning theory and could improve adaptability to out-of-distribution data.

6.2.4 Fine-Tuning Small Language Models for Targeted Hallucination Refinement

To improve efficiency and reduce computational overhead, a promising extension involves fine-tuning smaller language models (SLMs), such as Gemma and Mistral variants with 7-8B parameters, specifically for the task of hallucination refinement. Using the data generated in this study, a fine-tuning regimen could incorporate a balanced distribution of common error types. The training process might employ supervised fine-tuning with a contrastive loss to better distinguish between hallucinated and grounded outputs. Performance could be assessed on held-out test sets using metrics like exact match accuracy and hallucination rate, comparing the fine-tuned SLMs against larger, more computationally expensive baselines. This approach would directly address the scalability limitations noted in this study, enabling deployment on resource-constrained devices.

6.2.5 Integrating Hybrid Retrieval Systems with Advanced Evaluation Metrics

Finally, augmenting the framework with a hybrid retrieval system represents a significant opportunity to address knowledge staleness. Future research could combine the model's internal parametric knowledge with non-parametric retrieval from external, up-to-date sources, such as online APIs or structured datasets like Wikidata. To optimize this process, advanced chunking strategies could be paired with vector stores for efficient querying. A thorough evaluation would require metrics beyond standard precision, including Mean Reciprocal Rank (MRR) for ranking quality and latency measurements to assess real-time viability. This integration would overcome one of the primary weaknesses of static LLMs, paving the way for more dynamic and verifiable systems.

Bibliography

[1] Ahmed Abdallah and Samhaa R. El-Beltagy. Hallusearch at semeval-2025 task 3: A search-enhanced rag pipeline for hallucination detection. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, 2025. URL `https://arxiv.org/abs/2504.10168`.

[2] Vibhor Agarwal, Yiqiao Jin, Mohit Chandra, Munmun De Choudhury, Srijan Kumar, and Nishanth R. Sastry. Medhalu: Hallucinations in responses to healthcare queries by large language models. *ArXiv*, abs/2409.19492, 2024. URL `https://api.semanticscholar.org/CorpusID:272987286`.

[3] Mahaman Sanoussi Yahaya Alassan, Jessica López Espejel, Merieme Bouhandi, Walid Dahhane, and El Hassane Ettifouri. Comparison of open-source and proprietary llms for machine reading comprehension: A practical analysis for industrial applications, 2024. URL `https://arxiv.org/abs/2406.13713`.

[4] Deema Alnuhait, Neeraja Kirtane, Muhammad Khalifa, and Hao Peng. Factcheckmate: Preemptively detecting and mitigating hallucinations in lms. *ArXiv*, abs/2410.02899, 2024. URL `https://api.semanticscholar.org/CorpusID:273163183`.

[5] Omer Aydin, Enis Karaarslan, Fatih Safa Erenay, and Nebojsa Bacanin. Generative ai in academic writing: A comparison of deepseek, qwen, chatgpt, gemini, llama, mistral, and gemma, 2025. URL `https://arxiv.org/abs/2503.04765`.

[6] Forrest Sheng Bao, Miaoran Li, Renyi Qu, Ge Luo, Erana Wan, Yujia Tang, Weisi Fan, Manveer Singh Tamber, Suleman Kazi, Vivek Sourabh, Mike Qi, Ruixuan Tu, Chenyu Xu, Matthew Gonzales, Ofer Mendelevitch, and Amin Ahmad. Faithbench: A diverse hallucination benchmark for summarization by modern llms. *ArXiv*, abs/2410.13210, 2024. URL `https://api.semanticscholar.org/CorpusID:273404197`.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[8] Shiqi Chen, Yiran Zhang, Sijia Chang, Zhen Cheng, Huajun Zhang, and Wen Zhao. Felm: Factual error detection and correction for large language models. *arXiv preprint arXiv:2308.03975*, 2023.

[9] Yung-Sung Chuang, Yujia Li, Hong-Kuan Li, Chi-han Yeh, and James Glass. Dola: Decoding by contrasting layers of attentions. *arXiv preprint arXiv:2309.03883*, 2023.

[10] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.

[11] Luyu Gao, Aman Madaan, Shuyan Yao, Yiming Peng, Jean Egolf, Yue Wu, Graham Neubig, and Kartik Arora. Rarr: Researching and revising what language models say. *arXiv preprint arXiv:2308.08726*, 2023.

[12] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025. ISSN 1046-8188. doi: 10.1145/3703155. URL `https://doi.org/10.1145/3703155`.

[13] Sicong Huang, Jincheng He, Shiyuan Huang, Karthik Raja Anandan, Arkajyoti Chakraborty, and Ian Lane. Ucsc at semeval-2025 task 3: Context, models and prompt optimization for automated hallucination detection in llm output, 2025. URL `https://arxiv.org/abs/2505.03030`.

[14] Sean Hunkapiller, Varun Gangal, Alham Fikri Aji, and et al. Semeval-2025 task 3: Mu-shroom, the multilingual shared task on hallucinations and related observable overgeneration mistakes. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, 2025. URL `https://arxiv.org/abs/2504.11975`.

[15] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. A survey of hallucination in large language models. *ACM Computing Surveys*, 2023.

[16] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra S Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Douwe Kiela, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[18] Junyi Li, Xiaoxue Cheng, Wayne Xin Chen, Jian-Guang Wang, Philip Fung, and Eduard Hovy. Halueval: A large-scale hallucination evaluation benchmark for large language models. *arXiv preprint arXiv:2305.11747*, 2023.

[19] Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning, 2024. URL `https://arxiv.org/abs/2406.00922`.

[20] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2022.

[21] Q. Liu, Xinlong Chen, Yue Ding, Shizhen Xu, Shu Wu, and Liang Wang. Attention-guided self-reflection for zero-shot hallucination detection in large language models. *ArXiv*, abs/2501.09997, 2025. URL `https://api.semanticscholar.org/CorpusID:275606521`.

[22] Alejandro Lopez-Lira and Yuehua Tang. Can ChatGPT forecast stock price movements? return predictability and large language models. *Available at SSRN 4443778*, 2023.

[23] Wen Luo, Tianshu Shen, Wei Li, Guangyue Peng, Richeng Xuan, Houfeng Wang, and Xi Yang. Halludial: A large-scale benchmark for automatic dialogue-level hallucination evaluation. *ArXiv*, abs/2406.07070, 2024. URL `https://api.semanticscholar.org/CorpusID:270380046`.

[24] Potsawee Manakul, Adian Li, and Mark J. F. Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

[25] Lawrence Pearlman. The ChatGPT-generated brief: A case study in AI-powered legal malpractice. *Available at SSRN 4477908*, 2023.

[26] Han Qiu, Jiaxing Huang, Peng Gao, Qin Qi, Xiaoqin Zhang, Ling Shao, and Shijian Lu. Longhalqa: Long-context hallucination evaluation for multimodal large language models. *ArXiv*, abs/2410.09962, 2024. URL `https://api.semanticscholar.org/CorpusID:273345697`.

[27] Ernesto Quevedo, Jorge Yero, Rachel Koerner, Pablo Rivas, and Tomás Cerný. Detecting hallucinations in large language model generation: A token probability approach. *ArXiv*, abs/2405.19648, 2024. URL `https://api.semanticscholar.org/CorpusID:270123170`.

[28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[29] Yakir Yehuda, Itzik Malkiel, Oren Barkan, Jonathan Weill, Royi Ronen, and Noam Koenigstein. Interrogatellm: Zero-resource hallucination detection in llm-generated answers. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL `https://api.semanticscholar.org/CorpusID:268248107`.

[30] Jiawei Zhang, Chejian Xu, Yu Gai, Freddy Lécué, Dawn Song, and Bo Li. Knowhalu: Hallucination detection via multi-form knowledge based factual checking. *ArXiv*, abs/2404.02935, 2024. URL `https://api.semanticscholar.org/CorpusID:268889481`.

[31] Wayne Xin Zhao, Kun Zhou, Junran Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Appendices

Appendix A

All Results for both pipelines

The Table A.1 shows the JSON structuring and claim decomposition failure rates in all the open-source models tested, across all context in the hallusearch inspired multi stage pipeline. Continuing on, Table A.2 shows the Intersection-over-union and Spearman's Rank Correlation Coefficient scores for all the tested model and context pairings in the same pipeline.

Table A.3 shows the shows the Intersection-over-union and Spearman's Rank Correlation Coefficient scores for all the 10 open source models across all 4 contexts in the UCSC inspired single stage prompt pipeline.

Model	**Context**	**Test**		**Val**	
		Decomp Fail.	**JSON Fail.**	**Decomp Fail.**	**JSON Fail.**
Mistral-Nemo-Instruct-2407	phi-4	0.16	0.14	0.10	0.16
	perplexity-sonar-pro	0.16	0.18	0.12	0.12
	gemma3	0.12	0.16	0.12	0.12
	gpt-oss-20b	0.12	0.16	0.08	0.12
Falcon3-10B-Instruct	phi-4	0.25	0.51	0.24	0.46
	perplexity-sonar-pro	0.26	0.47	0.22	0.58
	gemma3	0.23	0.45	0.30	0.56
	gpt-oss-20b	0.25	0.47	0.30	0.60
Qwen2.5-14B-Instruct	phi-4	0.20	0.14	0.12	0.12
	perplexity-sonar-pro	0.20	0.14	0.14	0.10
	gemma3	0.17	0.14	0.14	0.14
	gpt-oss-20b	0.17	0.14	0.14	0.16
Midm-2.0-Base-Instruct	phi-4	0.16	0.21	0.14	0.28
	perplexity-sonar-pro	0.15	0.31	0.08	0.28
	gemma3	0.16	0.25	0.12	0.18
	gpt-oss-20b	0.16	0.24	0.10	0.24
phi-4	phi-4	0.18	0.17	0.16	0.16
	perplexity-sonar-pro	0.17	0.18	0.10	0.16
	gemma3	0.16	0.19	0.10	0.12
	gpt-oss-20b	0.17	0.18	0.12	0.20
gemma-3-12b-it	phi-4	0.17	0.20	0.12	0.22
	perplexity-sonar-pro	0.17	0.18	0.12	0.18
	gemma3	0.18	0.19	0.12	0.18
	gpt-oss-20b	0.18	0.23	0.12	0.26
Llama-3.1-8B-Instruct	phi-4	0.14	0.20	0.14	0.22
	perplexity-sonar-pro	0.15	0.22	0.18	0.18
	gpt-oss-20b	0.15	0.21	0.14	0.28
gpt-oss-20b	phi-4	1.00	0.98	1.00	0.96
	perplexity-sonar-pro	1.00	0.97	1.00	0.98
	gemma3	1.00	0.99	1.00	0.98
	gpt-oss-20b	1.00	0.95	1.00	1.00
openchat-3.5-0106	phi-4	0.19	0.30	0.20	0.40
	perplexity-sonar-pro	0.17	0.41	0.20	0.44
	gemma3	0.19	0.40	0.24	0.40
	gpt-oss-20b	0.18	0.31	0.18	0.44
Falcon-H1-7B-Instruct	perplexity-sonar-pro	0.14	0.25	0.16	0.34

Table A.1: Decomposition and JSON failure rates for all models in multi-stage pipeline

Model	**Context**	**Val**		**Test**	
		IoU	**Corr**	**IoU**	**Corr**
phi-4	perplexity-sonar-pro	0.39	0.43	0.41	0.41
	gpt-oss-20b	0.39	0.39	0.41	0.40
	phi-4	0.40	0.35	0.42	0.43
	gemma3	0.40	0.37	0.43	0.40
openchat-3.5-0106	perplexity-sonar-pro	0.21	0.18	0.30	0.25
	phi-4	0.28	0.19	0.33	0.26
	gpt-oss-20b	0.29	0.21	0.29	0.24
	gemma3	0.31	0.25	0.27	0.24
gpt-oss-20b	gpt-oss-20b	0.04	0.00	0.06	0.02
	perplexity-sonar-pro	0.06	0.01	0.05	0.02
	gemma3	0.06	0.01	0.05	0.01
	phi-4	0.07	0.03	0.04	0.01
gemma-3-12b-it	gpt-oss-20b	0.40	0.38	0.40	0.39
	phi-4	0.40	0.43	0.39	0.39
	perplexity-sonar-pro	0.40	0.44	0.43	0.43
	gemma3	0.43	0.38	0.43	0.39
Qwen2.5-14B-Instruct	perplexity-sonar-pro	0.39	0.40	0.43	0.40
	phi-4	0.40	0.39	0.39	0.40
	gemma3	0.41	0.39	0.41	0.40
	gpt-oss-20b	0.43	0.44	0.39	0.37
Mistral-Nemo-Instruct-2407	phi-4	0.35	0.37	0.38	0.36
	perplexity-sonar-pro	0.36	0.40	0.39	0.36
	gemma3	0.37	0.44	0.37	0.36
	gpt-oss-20b	0.38	0.42	0.33	0.32
Midm-2.0-Base-Instruct	gpt-oss-20b	0.31	0.33	0.36	0.36
	perplexity-sonar-pro	0.34	0.31	0.34	0.34
	phi-4	0.35	0.33	0.37	0.34
	gemma3	0.40	0.34	0.39	0.35
Llama-3.1-8B-Instruct	gpt-oss-20b	0.31	0.34	0.36	0.35
	phi-4	0.32	0.39	0.34	0.35
	perplexity-sonar-pro	0.37	0.38	0.37	0.36
	gemma3	0.37	0.34	–	–
Falcon3-10B-Instruct	gemma3	0.17	0.19	0.27	0.23
	gpt-oss-20b	0.19	0.22	0.28	0.23
	perplexity-sonar-pro	0.21	0.21	0.27	0.23
	phi-4	0.25	0.26	0.22	0.21
Falcon-H1-7B-Instruct	gpt-oss-20b	0.29	0.28	–	–
	perplexity-sonar-pro	0.32	0.35	0.40	0.40
	phi-4	0.32	0.33	–	–
	gemma3	0.34	0.32	–	–

Table A.2: IoU and correlation metrics for all models in multi-stage pipeline

Model	Context	Test		Val	
		IoU	Corr	IoU	Corr
phi-4	perplexity-sonar-pro	0.46	0.37	0.40	0.32
	gpt-oss-20b	0.45	0.38	0.42	0.37
	phi-4	0.46	0.37	0.40	0.41
	gemma3	0.41	0.33	0.38	0.38
openchat-3.5-0106	perplexity-sonar-pro	–	–	0.38	0.33
	phi-4	–	–	0.42	0.40
	gpt-oss-20b	0.41	0.33	–	–
	gemma3	0.47	0.40	0.36	0.36
gpt-oss-20b	gpt-oss-20b	0.43	0.38	0.44	0.45
	perplexity-sonar-pro	0.46	0.38	0.45	0.41
	gemma3	0.45	0.39	0.46	0.41
	phi-4	0.46	0.38	0.47	0.39
gemma-3-12b-it	gpt-oss-20b	–	–	0.44	0.48
	phi-4	0.52	0.48	0.48	0.51
	perplexity-sonar-pro	0.54	0.53	0.50	0.53
	gemma3	0.52	0.50	0.47	0.46
Qwen2.5-14B-Instruct	perplexity-sonar-pro	0.50	0.43	0.43	0.42
	phi-4	0.48	0.40	0.44	0.49
	gemma3	0.50	0.46	0.48	0.51
	gpt-oss-20b	0.45	0.40	0.41	0.39
Mistral-Nemo-Instruct-2407	phi-4	0.45	0.38	0.42	0.42
	perplexity-sonar-pro	0.44	0.37	0.41	0.35
	gemma3	0.46	0.41	0.42	0.43
	gpt-oss-20b	0.41	0.35	0.43	0.34
Midm-2.0-Base-Instruct	gpt-oss-20b	0.47	0.38	0.43	0.43
	perplexity-sonar-pro	0.53	0.44	0.48	0.46
	phi-4	0.49	0.40	0.45	0.40
	gemma3	0.46	0.38	0.47	0.42
Llama-3.1-8B-Instruct	gpt-oss-20b	0.45	0.42	0.44	0.44
	phi-4	0.45	0.41	0.49	0.51
	perplexity-sonar-pro	0.47	0.46	0.43	0.46
	gemma3	0.46	0.43	0.41	0.43
Falcon3-10B-Instruct	gemma3	0.45	0.36	0.35	0.31
	gpt-oss-20b	0.41	0.31	0.39	0.27
	perplexity-sonar-pro	0.43	0.35	0.40	0.37
	phi-4	0.44	0.36	0.35	0.27
Falcon-H1-7B-Instruct	gpt-oss-20b	0.47	0.45	–	–
	perplexity-sonar-pro	0.53	0.47	0.42	0.46
	phi-4	0.48	0.46	0.40	0.38
	gemma3	0.48	0.44	0.40	0.45

Table A.3: IoU and correlation metrics for all models in multi-stage pipeline