University of Milano-Bicocca
Department of Informatics, Systems and Communication
Master's Degree in Data Science

Detecting Factuality Hallucinations in Large Language Models: Challenges and Solutions

Supervisor: Prof. Marco Viviani

Master's Degree Thesis by:
Cristian Ceccarelli
ID Number 902254

Academic Year 2023-2024

Abstract

In recent years, the emergence of Large Language Models (LLMs) has revolutionized the field of Natural Language Processing (NLP), driving significant advancements in several applications, including text generation, machine translation, and conversational AI. Despite their remarkable capabilities, these models exhibit a critical limitation: their tendency to generate inaccurate or misleading content, a phenomenon known as hallucination. Hallucinations in LLMs can be categorized into two primary types: factuality hallucinations, where the model generates information that is contradictory, unverifiable, or misaligned with real-world knowledge, and faithfulness hallucinations, when the generated content is inconsistent with the input or the context provided by the user.

This study aims to conduct an in-depth investigation into the phenomenon of hallucinations in LLMs, examining their types, underlying causes, and the principal methods developed for their detection, with a focus on factuality hallucinations. To provide a comprehensive overview, this work presents a systematic review of the existing literature, categorizing hallucination detection strategies into three main approaches: methods based on knowledge retrieval from external sources, uncertainty estimation approaches through the analysis of the internal states of the LLM, and zero-resources and black-box methods, based on strategies to assess the consistency of the generated output. Beyond the theoretical analysis, this work also presents an experimental study comparing different hallucination detection methods and assessing their effectiveness on different benchmark datasets, highlighting the strengths and limitations of each approach.

The findings show that no approach consistently outperforms the others across all contexts. Supervised classification methods demonstrate superior performance in assessing the factual accuracy of long texts, yet their effectiveness diminishes when applied to shorter texts. Conversely, zero-resources and black-box approaches exhibit more stable and adaptable performance across several contexts. Furthermore, the integration of external knowledge within the approaches generally leads to an overall improvement in performance. However, the observed performance gains are not always substantial. Notably, further refining the knowledge retrieval process has the potential to yield even more significant improvements, emphasizing the importance of enhancing retrieval mechanisms to optimize hallucination detection strategies.

In conclusion, this study provides a significant contribution to the field of hallucination detection by presenting a comprehensive comparative analysis of existing methodologies. Additionally, it highlights potential future research directions aimed at enhancing the transparency and reliability of AI models.[1]

[1]Code developed during the work, data used, and results obtained can be found on GitHub at the following link: Link.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the advent of the digital era has profoundly transformed societal paradigms. In particular, rapid technological advancements and the increasing availability of data have led to the development of Large Language Models (LLMs). LLMs represent a class of advanced artificial intelligence models built on Transformer architectures, which leverage the concept of attention to analyze relationships between textual elements and capture contextual meaning effectively [1]. This mechanism enables LLMs to excel in natural language generation and a wide range of Natural Language Processing (NLP) tasks, including text summarization, machine translation, and conversational AI. Their remarkable ability to understand, interpret, and generate human-like language has established LLMs as indispensable tools in various fields, such as education, research, and healthcare, among others [2] [3] [4] [5].

Despite their remarkable capabilities and the significant technological advancements they represent, LLMs remain imperfect: one of their major limitations is their tendency to generate hallucinations. The term "hallucination" refers to instances in which a model generates a plausible but incorrect output. For example, consider the following dialogue:

```
User: "Who invented the airplane?"

LLM: "The airplane was invented by Leonardo da Vinci in 1505 when he
designed the first functional flying machine."
```

In this case, the LLM produced an inaccurate response. Although Leonardo da Vinci conceptualized and sketched various flying machines, the first functional airplane was constructed by the Wright brothers in 1903. Here, the model erroneously attributed the invention of the airplane to Leonardo da Vinci, likely due to his frequent association with aerodynamics and early aviation concepts. This example shows how models can generate content that appears credible but lacks verification from reality. In the literature, hallucinations can be classified into distinct categories: specifically, factuality hallucinations occur when an LLM generates an output that contradicts real-world knowledge. In contrast, faithfulness hallucinations arise when the model's output is inconsistent with the given input or the contextual information provided by the user.

This issue is particularly concerning given the widespread adoption of LLMs across various domains. The generation of non-factual information poses a significant risk, as it may undermine the trust in AI-generated content, limit the practical applicability of LLMs, and contribute to the spread of misinformation, especially in critical fields such

as journalism, healthcare, and scientific research, where incorrect information can have serious and potentially harmful consequences. Given the significant impact and potential negative outcomes that the generation of hallucinations can have on the lives of individuals, this issue represents a critical challenge in the context of LLMs. It is therefore essential to fully understand the phenomenon, and develop advanced techniques for hallucination detection and mitigation to ensure the reliability and safety of these technologies in real-world applications.

1.1 Main Challenges in Hallucination Detection

Despite the significant advancements achieved in recent years, hallucination detection remains a contemporary and highly relevant challenge, presenting numerous complexities. The primary difficulties and challenges associated with hallucination detection include:

- **Lack of clear definitions and standardized metrics:** One of the main obstacles in hallucination detection is the lack of clear, precise, and universally accepted definitions of the phenomenon by the scientific community. Often, terms such as "hallucination", "misinformation", and "confabulation" are used interchangeably without precise distinction, complicating the understanding of the phenomenon, hindering the development of effective strategies, and making it difficult to compare solutions. A further obstacle is the absence of universally recognized standardized metrics for quantifying hallucinations in language models. As a consequence, some studies rely on qualitative metrics while others rely on quantitative metrics, making comparisons between different approaches and methods even more complicated.
- **Limited number of manually annotated hallucination benchmarks:** Most of the existing benchmarks for the detection of hallucinations are based on short and factual answers to questions, where it is easy to verify the correctness of the information provided. However, hallucinations can take more subtle forms that are difficult to detect, and the absence of specific benchmarks hinders the study of these phenomena, limiting the possibility of developing targeted methods for their detection. Moreover, manual annotation is costly and time-consuming, thus limiting the development of effective benchmark datasets.
- **Invisibility of errors:** One of the most complex issues to consider in hallucination detection is their surface credibility: LLMs generate grammatically correct, consistent, and convincing responses even when they contain errors. This makes a factual text and a hallucinated text virtually indistinguishable by both humans and machines, making it complicated to detect hallucinations using traditional methods such as human evaluation and text classification.
- **Generalization across different domains:** LLMs are distinguished by their versatility and operability in multi-task, multi-language, and multi-domain contexts. However, this adaptability introduces difficulties in hallucination detection such as the need for broad and robust benchmarks covering various types of tasks and domains, and encompassing multiple languages.

1.2 Research Objectives

Given the challenges presented above, this study aims to investigate and compare the principal methods for detecting hallucinations in LLMs, with a particular focus on factuality

hallucinations. Specifically, the main objectives of this work are:

- **Analysis of the phenomenon and its causes**: Examine in detail the various types of hallucinations that can occur in LLMs, and investigate the underlying factors that lead models to generate hallucinated content.
- **Review of existing approaches:** Provide a comprehensive overview of the main hallucination detection methods proposed in the literature, including techniques based on supervised classification, knowledge retrieval, and uncertainty estimation of the LLM through the analysis of its internal states.
- **Development and experimental evaluation:** Design and implement a series of experiments to assess the effectiveness of some of the hallucination detection methods in the literature, across different application contexts. Additionally, implement modifications to the selected approaches, and assess the impact of these changes on the performance.
- **Results analysis and future improvements:** Analyze the results obtained to identify areas of improvements, to increase the robustness of the models. Additionally, propose future research directions aimed at improving the detection of hallucinations, and enhancing the overall reliability of LLMs.

The main contribution of this study lies in the comparative analysis and evaluation of hallucination detection approaches. Furthermore, it introduces specific modifications to existing methods to assess their impact on performance. By offering a comprehensive overview of the current state-of-the-art, this work provides researchers with valuable insights to develop more innovative and reliable solutions in the field of hallucination detection.

1.3 Methodology Overview

To achieve the goals set, the process is divided into stages. In the first stage, a detailed review of the literature is conducted to identify the main types of hallucinations in LLMs, and the main causes of this phenomenon. Subsequently, the main hallucination detection approaches in the literature are reviewed, categorizing them, analyzing their methodology and results, and highlighting their advantages and disadvantages.

Next, we proceed with the definition of the experiments, which include:

- **Supervised text classification:** Implementation of text classification models to distinguish between truthful and hallucinated outputs through the use of annotated datasets.
- **Uncertainty estimation:** Extraction and analysis of the internal states of the LLM to check whether they contain information useful for distinguishing between factual and hallucinated outputs.
- **Zero-resources and black-box:** Implementation of approaches that do not require access to knowledge or internal model states, but are based on strategies that assess the consistency of the output generated by the LLM.

- **Knowledge retrieval:** Development of an automatic knowledge retrieval system using the Google Search API, and integration of the retrieved knowledge into the framework of various approaches, to evaluate the impact of knowledge on performance.

The experimental approach uses benchmark datasets for hallucination detection, prepared specifically for each method. This pathway makes it possible to highlight critical issues and strengths of the different strategies, providing a solid basis for future implementations and research in the field.

1.4 Structure of the Manuscript

The work described in this Master's Thesis is structured as follows:

- **Chapter 1 - Introduction:** Provides the research context, outlining the problem of hallucinations in LLMs, the main challenges, research objectives, and contributions. Additionally, it offers a general overview of the study.
- **Chapter 2 - Background and Related Work:** Conducts a detailed review of the existing literature, presenting an overview of the different types of hallucinations, their underlying causes, and the methods developed for their detection.
- **Chapter 3 - Methodology:** Describes the experimental approaches adopted, the models utilized, and the experiments conducted to evaluate hallucination detection techniques.
- **Chapter 4 - Results:** Introduces the datasets and evaluation metrics employed in the study, followed by a discussion of the results obtained, highlighting the performance of the proposed approaches, and analyzing their strengths and limitations.
- **Chapter 5 - Conclusions and Future Work:** Summarizes the main results of the study, and proposes potential directions for future research aimed at improving hallucination detection in LLMs.

Chapter 2

Background and Related Work

For a complete understanding of hallucinations in the context of LLMs, this chapter will discuss the various types of hallucinations that can occur. Subsequently, the main causes of this phenomenon and the most common methods in the literature for detecting hallucinations will be explored. This chapter will be based on the following works, which explain in detail the concept of hallucination and its underlying causes, as well as the benchmarks and methods of evaluation and detection most widely used in the literature:

- **Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models:** Research that explores recent efforts related to hallucinations in the context of LLMs, presenting taxonomies and approaches to detect and mitigate hallucinations, with particular emphasis on the challenges posed by LLMs and future research directions [6].
- **An Audit on the Perspectives and Challenges of Hallucinations in NLP:** This survey examines the characterization of the concept of hallucination in the peer-reviewed literature by analyzing 103 publications in the NLP context. The paper highlights various shortcomings (such as that of an unambiguous and universally accepted definition), and potential challenges, including from an ethical point of view, by analyzing hallucinations also from a societal perspective [7].
- **Survey of Hallucination in Natural Language Generation:** Survey exploring the concept of hallucination not only limited to LLMs but more broadly to the whole field of Natural Language Generation (NLG). The paper is divided into two parts: one exploring the main mitigation methods, problems, and future research directions while the other explores the phenomenon within specific tasks such as dialogue generation, generative question-answering, and text summarization [8].
- **A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions:** Survey on which the taxonomy explained in this chapter is based. The particularity of this paper is that it provides an innovative taxonomy related to hallucinations compared to most works, which focus more on LLM applications. It also provides a comprehensive overview of the most popular methods for the mitigation and detection of hallucinations, as well as challenges due to current limitations, and formulates open research questions to outline possible future developments on hallucinations in LLM [9].

2.1 Hallucinations in LLMs

The term *hallucination* originated long before the concept of large language model. Indeed, it was first introduced into the English language in the 17th century by Sir Thomas Browne in his essay *Pseudodoxia Epidemica*, from the Latin word *alucinari* [10]. For Browne, hallucination meant a sort of vision, and he defined the term as *"depraved and receive[s] its objects erroneously"*. Since then, the term hallucination was always associated with the fields of psychology and psychiatry until it was applied to the context of image resolution enhancement, referring to the generation of new pixel values [11]. Later, the term was often used in computer vision studies [12] and in Recurrent Neural Networks researches, illustrating how an LSTM could generate non-existent URLs, thus hallucinating the data [13]. Within the NLP context, the most standard definition attributed to the term hallucination is *"the generated content that is nonsensical or unfaithful to the provided source content"*, and similarly to hallucinations in the psychiatric context that are difficult to tell apart, hallucinated text is fluent and natural despite being nonsensical or non-factual, making it difficult to distinguish from non-hallucinated text. In the literature, the concept of hallucinations in NLG tasks is commonly divided into two primary types: intrinsic hallucination, where the output of the language model conflicts with the source content, and extrinsic hallucinations, where the generated output cannot be verified by the source content. However, according to [9], the spread of these models in various fields has highlighted the limitations of this task-specific categorization, thus necessitating a more detailed taxonomy oriented towards representing the unique complexities of hallucinations in LLMs. Thus, following this new line of thought, hallucinations in LLMs can be categorized into factuality hallucinations and faithfulness hallucinations.

2.1.1 Factuality Hallucinations

This type of hallucination encompasses all contents generated by the language model that conflict with real-world knowledge, and will be the category of hallucinations on which this work will focus most, being the one in which misinformation is present, and can be spread. Depending on whether the content generated by the model can be verified against reliable knowledge or source, factual hallucinations can be further divided into:

- **Factual inconsistency:** This type of hallucination occurs when the generated content can be compared with real-world knowledge, but contradicts it, thus generating incorrect information. For example, as can be seen in Table 2.1, if a user asked a language model *"Who was the first president of the United States?"*, and the model answered *"The first United States president was Abraham Lincoln"*, this would contradict the real-world facts, generating a hallucination of factual inconsistency.

- **Factual fabrication:** This type of factual hallucination refers to when the content generated by the model contains facts that cannot be verified against a reliable source or knowledge. For example, as shown in Table 2.1, if a user asked the model *"Can you tell me about the first documented sighting of dragons in history?"*, and the model replied *"The first documented sighting of dragons occurred in 1224 AD in what is now modern-day Switzerland"*, although the model has fabricated a plausible story, it cannot be verified against any real-world facts, thus generating a factual fabrication hallucination.

Sub-Type	User Input	Model Output	Explanation
Factual Inconsistency	Who was the first United States president?	The first United States president was Abraham Lincoln	The LLM's response is factually inconsistent as George Washington was the first United States president. Abraham Lincoln, despite being one of the most famous USA presidents, was not the first but the 16th.
Factual Fabrication	Can you tell me about the first documented sighting of dragons in history?	The first documented sighting of dragons occurred in 1224 AD in what is now modern-day Switzerland.	The LLM's response is a fabricated claim since there's no verified, real-world evidence supporting that dragons were seen in 1224 AD in Switzerland.

Table 2.1: Examples of factuality hallucinations. In red, there is the hallucinated portion of text generated by the model.

2.1.2 Faithfulness Hallucinations

These hallucinations occur when the generated content is inconsistent with the input or context provided by the user. Following this idea, we can further categorize faithfulness hallucinations into the following types:

- **Instruction inconsistency:** These types of hallucinations occur when the output generated by the model is misaligned with the input instructions provided by the user. For example, as shown in Table 2.2, if a user asked the model to translate a sentence from English to French, and the model translated it into Italian, then it would be a case of instruction inconsistency hallucination.

- **Context inconsistency:** These types of hallucinations occur when the output generated by the model is misaligned with the context provided as input by the user. For example, as can be seen in Table 2.2 if a user described a company founded in San Francisco in 2010, asking the model to provide it with a detailed overview, and the model produced an output in which it incorrectly described the company as being built in 2005 in New York, this would be inconsistent with the contextual information provided as input, and would be a context inconsistency hallucination.

- **Logical inconsistency:** These types of hallucinations occur mainly during reasoning tasks, that is when the model is led to perform reasoning steps. These hallucinations are characterized by inconsistency between the steps or inconsistency between the steps and the final response. For example, as shown in Table 2.2 if the user asked the model to solve the equation $6x - 1 = 11$, and the model returned $x = 5$ as output instead of $x = 2$, then this would be a logical inconsistency hallucination.

Sub-Type	User Input	Model Output	Explanation
Instruction Inconsistency	Translate the following sentence from English to French: Roses are red.	Le rose sono rosse	The user explicitly asked to translate the sentence from English to French. However, the model translated the sentence from English to Italian.
Context Inconsistency	...this company was founded in San Francisco in 2010 ... Provide a complete overview of this company.	Sure, here is a complete and detailed overview of the company. The company was founded in New York in 2005 ...	The user clarified that the company was founded in San Francisco in 2010. However, the model, when providing the overview, states that it was founded in New York in 2005.
Logical Inconsistency	Solve the following equation by performing reasoning steps: $6x - 1 = 11$.	Step 1: Add 1 to both sizes and the equation becomes $6x = 12$ Step 2: Divide both terms by 6 to get $x = 5$, so the result is $x = 5$.	During the first step, the model correctly sums 1 to both terms. However, in the second step, the model incorrectly divides 12 by 2, yielding a result of $x = 5$ which is inconsistent with the previous steps.

Table 2.2: Examples of faithfulness hallucinations. In red, there is the hallucinated portion of text generated by the model, while in blue there is the context or the instruction provided by the user which contradicts the output of the model.

2.2 Why Do LLMs Hallucinate?

The causes that lead an LLM to generate hallucinated content can be diverse, and in the literature are summarized in the following three categories:

- **Hallucinations from data:** Hallucinations due to data provided to the model during the pre-training phase.
- **Hallucinations from training:** Hallucinations due to mistakes during pre-training or alignment phases.
- **Hallucinations from inference:** Hallucinations due to decoding strategies during the inference phase.

In this section, the underlying causes of LLM hallucinations will be studied in detail.

2.2.1 Hallucinations From Data

Data are crucial for the model to acquire generalization capabilities and knowledge; however if the quality of the data during the pre-training process is poor, data could be a cause of hallucinations. Data-related hallucinations can result from two aspects: the source of data or certain characteristics of the knowledge itself.

Problematic Training Data

One of the main challenges when building an LLM from scratch is to maintain a high and consistent data quality; however, this is very difficult due to the enormous scale of data

needed to train an LLM. During data collection for the training, heuristic methods are implemented to effectively collect large amounts of data, and this could increase the risk of introducing incorrect information. [9] defined this type of hallucinations as *misinformation and biases*, and further categorize them into:

- **Imitative falsehoods:** These hallucinations can occur when the language model, trained on incorrect data, mimics the distribution of training data, amplifying inaccuracies, and generating non-factual content. For example, as shown in Table 2.3, if the user asked the model *"Who invented the telephone?"*, and it answered *"Graham Bell"*, the model would oversimplify a complex historical reality, as Antonio Meucci, an Italian inventor, developed a prototype of the telephone years before Bell, but was unable to patent it due to financial difficulties. Because of this common belief, the model has seen this data many times, and provided the wrong output.

- **Duplication bias:** Studies show that LLMs tend to store data viewed multiple times, and this habit increases as the size of the model increases [14] [15]. In this situation, the LLM may prioritize duplicate data, thus leading to hallucinations. For example, as can be seen from Table 2.3, if the user asked the model *"Name some famous scientists, excluding Albert Einstein"*, and it answered *"Isaac Newton, Albert Einstein, and Nikola Tesla are some of the most famous scientists in history"*, although the user explicitly asked to exclude Albert Einstein from the list of scientists, the combination of Einstein together with other scientists is widely repeated within the pre-training data. Therefore, the model memorizes and prioritizes this frequent association, generating outputs that deviate from the user's requests.

- **Social biases:** The data collected from the Internet to train the LLM might contain diverse and biased viewpoints, and this could be reflected in the output generated by the model. For example, as can be seen in Table 2.3, the LLM might associate the profession of a nurse with a woman, even if the gender is not explicitly specified in the context provided by the user. This error reflects a gender bias learned from pre-training data, which often associates nursing professions with women, perpetuating social stereotypes.

Within this category, we can also find hallucinations due to limitations of the acquired knowledge. Indeed, some of the data provided to the model may be out-of-date or knowledge related to specific domains, such as medicine and law, may be missing, introducing inaccuracies and misinformation into the model. [9] defines this type of hallucinations as *knowledge boundary* hallucinations, categorizing them into:

- **Domain knowledge deficiency:** LLMs possess a broad general knowledge, having been trained on a large number of publicly available datasets. Therefore, due to the lack of proprietary, domain-specific data (such as data about medicine) in the training phase, the model may exhibit hallucinatory behaviour, and could fabricate answers. For example, as shown in Table 2.4, if the user asked the model *"What is the recommended treatment for Lyme disease in its advanced stage?"*, and the model answered *"In advanced stages of Lyme disease, the recommended treatment is a daily dose of ibuprofen, as it completely eliminates the bacterial infection, and restores health within a week"*, this would be a false and dangerous statement. In fact, ibuprofen does not eliminate the bacterial infection caused by Lyme disease, which can only be eliminated by taking antibiotics such as doxycycline (in adults) and amoxicillin (in children) [16]. Therefore, this incorrect answer highlights the

Hallucination Type	User Input	Model Output	Explanation
Imitative Falsehoods	Who invented the telephone?	Graham Bell	Although Bell patented the telephone, it was actually invented by Antonio Meucci.
Duplication Bias	Name some famous scientists, excluding Albert Einstein	Isaac Newton, Albert Einstein, and Nikola Tesla are some of the most famous scientists in history	The user explicitly made it clear to the model to exclude Albert Einstein from the list, but despite this, it still included him.
Social Bias	Describe the typical duties of a nurse	A nurse is responsible for ... and she ensures that patients ...	Although the gender of the nurse is not specified in the context, the model automatically assumed that was a woman.

Table 2.3: Examples of misinformation and biases. In red, there is the hallucinated portion of text generated by the model.

model's limited knowledge in the medical field, as it has not been trained to answer very specific medical questions correctly.

- **Outdated factual knowledge:** Another limitation that knowledge may have is that of being outdated. Indeed, once the models have been trained, the knowledge within them is never updated, so when the model is asked a query regarding a fact that transcends the temporal boundary of its knowledge, it can invent answers or provide responses that would once have been correct but are now outdated. For example, as can be seen in Table 2.4, if the user asked *"Where was the last football World Cup hosted?"*, and the model answered *"The last football World Cup was hosted in Russia in 2018"*, it would be an obsolete answer that would illustrate the temporal limitations of the model, since the last football World Cup was in 2022, and was hosted in Qatar.

Hallucination Type	User Input	Model Output	Explanation
Domain Knowledge Deficiency	What is the recommended treatment for Lyme disease in its advanced stage?	In advanced stages of Lyme disease, the recommended treatment is a daily dose of ibuprofen, as it completely eliminates the bacterial infection, and restores health within a week	Ibuprofen does not eliminate the bacterial infection, which can only be treated through the use of antibiotics.
Outdated Factual Knowledge	Where was the last football World Cup hosted?	The last football World Cup was hosted in Russia in 2018	The last World Cup was held in Qatar in 2022, and not in Russia in 2018.

Table 2.4: Examples of hallucinations due to knowledge boundaries. In red, there is the hallucinated portion of text generated by the model.

Suboptimal Knowledge Utilization

Although pre-training data provide extensive real-world knowledge, sometimes the model fails to use this knowledge effectively, thus generating hallucinations. [9] categorized these hallucinations into:

- **Knowledge shortcut:** Although the method by which LLMs learn from training data remains elusive, some studies show how, instead of actually understanding factual knowledge, they resort to some shortcuts, such as word proximity or statistical co-occurrence of words within the documents [17] [18]. These shortcuts may lead the model to learn spurious correlations, and generate hallucinations if these correlations contain misinformation. For example, as can be seen in Table 2.5, if a user asked *"What is the capital city of Brazil?"*, and the model answered *"Rio de Janeiro"*, this would be an inaccuracy since the capital city of Brazil is Brasilia. This could be due to the fact that, probably, the word "Rio de Janeiro" appears very often next to the word "Brazil" within the training data, leading the model to learn a relationship (in this case, state and capital city relationship) that does not actually exist, generating misinformation.
- **Knowledge recall failures:** Given the large knowledge it possesses, the LLM may struggle to recall it. This can be noticed in two situations: the first is when the model has to recall knowledge that is rare within the training data, while the second situation is when the model has to deal with complex scenarios that require reasoning. In the first case, the model may fabricate inaccurate answers, while in the second case, due to its reasoning limitations, it may generate incorrect results. For example, as can be seen in Table 2.5, if a user asked *"Who was Raden Adjeng Kartini, and what is she known for?"*, and the model answered *"Raden Adjeng Kartini was an Indonesian scientist best known for her contributions to physics"*, it would be an inaccuracy, since Raden Adjeng Kartini was an activist in the field of women's rights and education, and not a physicist [19]. However, it is possible that Kartini's name is underrepresented within the knowledge of the model, and that it provided an inaccurate answer based on incorrect or incomplete associations within the training data.

2.2.2 Hallucinations from Training

The training of an LLM mainly comprises two phases:

1. **Pre-training phase:** Phase in which unstructured and unlabeled data are provided to the model from which it comprehends information and concepts, thus learning real-world knowledge.
2. **Alignment phase:** It is a process that ensures that the content generated by the LLM aligns with human ethics and moral values.

Both of these processes are crucial for the development and proper functioning of an LLM. However, an error within these phases can lead the model to generate hallucinations.

Hallucination Type	User Input	Model Output	Explanation
Knowledge Shortcut	What is the capital city of Brazil?	The capital city of Brazil is Rio de Janeiro	The model has seen the words "Rio de Janeiro" and "Brazil" next to each other many times, and learned a wrong relationship, as the capital of Brazil is Brasilia.
Knowledge Recall Failures	Who was Raden Adjeng Kartini, and what is she known for?	Raden Adjeng Kartini was an Indonesian scientist best known for her contributions to physics	There is few knowledge related to Raden Adjeng Kartini within the model, which fabricates its response by saying that she was a physicist, while Kartini was a women's rights activist.

Table 2.5: Examples of hallucinations due to suboptimal knowledge utilization. In red, there is the hallucinated portion of text generated by the model.

Hallucinations from Pre-training

Within the pre-training phase, hallucinations can be caused by flaws within the model architecture or by the training strategies adopted. [9] classifies these reasons into *architecture flaw* and *exposure bias*, depending on whether they originate from the internal architecture of the model or from the way the training is carried out.

- **Architecture flaw:** LLMs predict the next tokens based on the previous ones, following the context from left to right, facilitating training. However, it only uses context in one direction, limiting the model's ability to capture contextual dependencies, and increasing the risk of generating hallucinations. In addition, studies have shown how attention modules within LLMs can foster unpredictable errors during the algorithmic reasoning of the model, and favor the generation of hallucinated outputs [20].
- **Exposure bias:** The concept of exposure bias relates to a difference between how LLMs are trained, and how they work during inference. Indeed, during training, the model has access to the correct answers so the next token distribution is conditioned by the ground truth, whereas during inference the model does not know the correct answers, and has to rely only on the previously generated tokens. This can generate cascading errors where an error progressively worsens the generated text; in fact, if the model generates a wrong token it will use that error as the basis for the next predictions, further deviating from what it should have produced [21].

Hallucinations from Alignment

The alignment phase of the LLM is crucial to improve the quality of responses, and align them with human preferences. This process is mainly based on two techniques:

1. **Supervised Fine-Tuning (SFT):** It is the first step in the LLM alignment process, and consists of providing the model with high-quality outputs (aligned with ethical and moral values) so that it learns to replicate the style of these examples.

2. **Reinforcement Learning from Human Feedback (RLHF):** This procedure uses reinforcement learning techniques to align the model with human preferences, also including the human within the training cycle.

Despite its importance and benefits, the alignment process can introduce the risk of hallucinations, as it can push the model to behave beyond its capabilities. [9] divides these causes into two categories: *capability misalignment* and *belief misalignment*.

- **Capability misalignment:** The models have inherent limits defined during the pre-training phase, and SFT helps them unlock the acquired capabilities. However, sometimes the alignment may require the model to produce content that exceeds its knowledge limits, such as answering questions never encountered during the pre-training phase, and this can lead to the production of hallucinated content as the model attempts to meet demands it cannot handle.
- **Belief misalignment:** Some studies show how the model, during alignment, develops an internal belief of what is true, and what is false, based on training data [22]. However, it may happen that the answers given do not always reflect this internal belief. Indeed, the model may be driven to give answers that please the human evaluators while putting the truthfulness of the statement in the background: this behavior is called *sycophancy* [23]. As a result, the model may give false answers just to please the human evaluators, thus generating hallucinations.

2.2.3 Hallucinations from Inference

After training and optimization, the model will be used during inference, where commands will be given or questions will be asked to the model, which must generate the answer according to decoding strategies. However, during this process, errors may occur, leading the model to hallucinate. [9] divides the causes due to inference into *inherent sampling randomness* and *imperfect decoding representation*.

Inherent Sampling Randomness

During text generation, randomness is introduced within the decision-making process of LLMs so that the content is varied, and not repetitive. In fact, it has been noted that highly probable sequences result in a trivial, low-quality text (a phenomenon called *likelihood trap* [24]), so randomness helps to generate more natural responses. However, increasing randomness too much, through a parameter called temperature, will make the token probability distribution more uniform, increasing the probability that the model will choose incorrect tokens, and with it, the probability that the model will generate hallucinated content.

Imperfect Decoding Representation

During the decoding phase, LLMs use their final layer to predict the next token. However, this layer has limitations manifested in the following aspects:

- **Insufficient context attention:** During text generation, models are too focused on the partial content they have already produced, prioritizing fluency of response at the expense of faithfulness to the original context. In addition, language models exhibit localized focus within their attention mechanisms, prioritizing nearby words,

resulting in a deficit of contextual attention, and increasing the risk of generating faithfulness hallucinations. For example, if the user asked *"Tell me about the French Revolution, and Napoleon's role in it"*, and the model responded *"Napoleon was a great strategist. He won many battles ..."*, the model focuses only on the figure of Napoleon, leaving out the French Revolution. This happens because the LLM focuses on the most recent words or more familiar concepts in the prompt (Napoleon), neglecting the overall context (the Revolution).

- **Softmax bottleneck:** Most language models use a softmax layer that operates on the final layer's representation to calculate the final probability of each word. However, softmax cannot represent all possible complex distributions: for example, if a probability distribution has multiple modes (multiple very probable words), the model struggles to handle it through the softmax layer [25], and this could lead the LLM to choose words that do not accurately reflect the original context, generating hallucinations.

2.3 Related Work

This section will discuss the main methods for detecting hallucinations that are present in the literature. As mentioned in the previous sections, the focus will be on factuality hallucinations as these are the ones that can spread misinformation, so the methods presented will mainly focus on detecting this type of hallucination. Furthermore, studies [26] have shown that human detection of LLM-generated misinformation is very challenging, and as a result, more and more studies have focused on the development of approaches and strategies that detect factuality hallucinations.

One of the classic and most natural methods for identifying hallucinations, is to provide the text to be evaluated as input to a classification model. Subsequently, the text will be processed by the model, and a vector representation of the text will be constructed by the model, which will classify it into hallucinated or non-hallucinated text, performing a supervised classification task. Researches that propose this approach are:

- **AILS-NTUA at SemEval-2024 Task 6: Efficient model tuning for hallucination detection and analysis:** This paper provides several approaches for the identification of hallucinations, and one of the proposed techniques involves the use of a model to perform a text classification task [27]. The model in question was pre-trained on Natural Language Inference datasets for understanding the relationship between sentences in a text, and then fine-tuned on summarization datasets. To adapt it to the specific task, the authors further fine-tuned it on 1000 annotated samples of the SHROOM dataset. The output of the model consists of a probability between 0 and 1, where 0 indicates hallucination, and 1 indicates factual consistency, so a threshold-based strategy for classifying texts as hallucinated or factual was also implemented. The model was evaluated on two types of dataset: a model-agnostic dataset, that is where the model that generated the passages is unknown, and a model-aware dataset where the model that generated the texts is known. The metrics chosen for evaluation are accuracy and Spearman's correlation index. The proposed approach achieves good results on both datasets: for the model-agnostic dataset, the model obtains 77.8% accuracy while for the model-aware dataset, the model obtains 79.5% accuracy. However, the best approach turns out to be a model ensemble, obtained by aggregating, through a majority voting mode, the predictions

of the classifier explained above with those of models trained in solving Natural Language Inference tasks. This approach obtains 80% accuracy in the model-aware dataset, and 78% in the model-agnostic dataset, outperforming the performance of the simple hallucination classifier.

- **A Token-level Reference-free Hallucination Detection Benchmark for Free-form Text Generation:** This research proposes HaDes, a benchmark dataset for token-level hallucination detection, obtained by collecting text segments from Wikipedia, and perturbing them to simulate hallucinations generated by LLMs [28]. These perturbed texts were then manually annotated, also using an automatic annotation strategy to handle the class imbalance problem, and limit the cost of human annotation. As an initial step to address the problem of hallucination detection on this dataset, several baselines were provided; among them, some Transformer-based models such as BERT and GPT-2, which exploit embedded knowledge to detect non-factual content, were tested. Specifically, the vector representation of text is obtained from the last layer of the model, which is then passed to a Multilayer Perceptron (MLP) for binary classification into hallucination or non-hallucination. Baseline performance on HaDes was evaluated with standard classification metrics such as accuracy, precision, recall, and F1-score, as well as AUC. GPT-2 is the best performing model achieving 71.5% accuracy, followed by BERT with 71% prediction accuracy.

However, supervised text classification methods are often used as baselines since they perform well only in specific datasets where there is a clear difference between hallucinated and non-hallucinated content. Indeed, given the great text generation capabilities of LLMs, hallucinated content is often indistinguishable in comparison to factual texts, making the distinction very complicated for both humans and models, requiring more effective and sophisticated strategies for more accurate distinction. In addition, text classification approaches require thousands of manually annotated texts, which are very difficult to obtain since human annotation is time-consuming and expensive.

Besides the classical supervised text classification approach, methods for detecting factuality hallucinations can be categorized into three main groups:

- **Knowledge retrieval:** These approaches consist of comparing the content generated by the LLM with a reliable external knowledge base.
- **Uncertainty estimation:** These methods are based on estimating the LLM's confidence in the responses generated, by analyzing its internal states.
- **Zero-resources and black-box:** This class refers to techniques that do not depend on external knowledge sources (zero-resources), and do not require access to the internal states of the model (black-box), but rely only on the analysis of model behavior, inputs, and outputs to identify potential discrepancies or inconsistencies.

In the following sections, these methods will be discussed in detail as well as the advantages and disadvantages of each method, and the main hallucination detection approaches belonging to each category.

2.3.1 Knowledge Retrieval Methods

This category of methods uses external knowledge sources such as structured databases (*e.g.*, knowledge graphs), online encyclopedias (*e.g.*, Wikipedia), APIs, or search engines to check the factuality of the outputs generated by the model. These methods are highly flexible and reliable, being able to be applied to different domains, and being based on reliable knowledge sources, as well as facilitating the explanation of an outcome. However, they are highly dependent on the quality and correctness of knowledge; in fact, if the references turn out to be incorrect, ambiguous, or outdated, the effectiveness of these methods could suffer. In addition, information retrieval and processing could considerably increase the computational cost of the work.

FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation

The main goal of this paper is to evaluate the factuality of long texts generated by LLMs, and the authors address this problem by proposing FActScore, a method of evaluating the truthfulness of a language model that represents the percentage of atomic facts (pieces of information) supported by a given external knowledge [29]. The process of evaluating an LLM occurs in two stages:

1. The first step is to break a model-generated content into a series of short statements, called atomic facts, each containing a piece of information. This subdivision into atomic facts provides a more fine-grained evaluation of the text.

2. The second step is to assign a binary label to each statement, depending on whether or not it is supported by external knowledge.

At this point, given M a language model, X a set of prompts, C a knowledge source, $y = M_x$ the response of model M to the prompt $x \in X$, and A_y a list of atomic facts, the FActScore of y will be defined as:

$$f(y) = \frac{1}{|A_y|} \sum_{a \in A_y} \mathbb{1}_{\{a \text{ is supported by } C\}}$$

where $|A_y|$ represents the number of atomic facts. Consequently, the FActScore of model M will be the average of the FActScores of all the answers of M.

The models evaluated using this method are ChatGPT, InstructGPT, and PerplexityAI, and were evaluated on the generation of biographies, as they are objective, and contain specific information. The data collection and generation process was made according to the following steps:

1. 183 people who have a page on Wikipedia were sampled from Wikidata.

2. The evaluated LLM was prompted to generate a biography of the person, filtering out the responses in which the model abstained.

3. Human annotators break the generation into a series of atomic facts.

4. Another group of human annotators labels atomic facts as supported or not-supported, based on the English Wikipedia.

The main findings of the evaluation can be summarized as follows:

- **All language models struggle with factual precision:** InstructGPT and ChatGPT achieve FActScores of 42.5% and 58.3%, respectively, while PerplexityAI reaches 71.5%. The score of PerplexityAI is lower than expected since this model uses a search engine for its responses. This means that if it had directly copied the text from the Wikipedia page of the entity, it would have obtained the highest score. Moreover, ChatGPT and PerplexityAI improve their factual accuracy by frequently abstaining from answering, unlike InstructGPT, which rarely abstains, a behavior likely due to its training.
- **Error rates are higher if the entity is rare:** As the rarity of entities increased, a decrease in the FActScores related to those specific entities was noted.
- **Error rates are higher for facts mentioned later in the generation:** The later part of the generation has worse precision, due to the fact that information mentioned earlier is more frequent in the pre-training data, and error propagation affects the later part of the generation (see *Exposure bias* in Section 2.2.2).

Moreover, since human annotation is expensive and time-consuming, the authors automated the process of labeling generations, entrusting it to another LLM that had the task of checking the knowledge source, and labeling atomic facts accordingly. In this way, the authors were able to estimate the FActScore of several LLMs automatically, achieving remarkable results. However, we will not go into the details of the methodology and results obtained in the second part of the paper, as comparing the truthfulness of language models is beyond our scope. The main takeaway from this paper is the development of a metric capable of evaluating the factuality of texts generated by LLMs, and able to detect hallucinations, since this metric has also been used in several papers as a method, basis or inspiration for detecting hallucinations, after it gained popularity [30] [31] [32].

Retrieving Supporting Evidence for Generative Question Answering

The authors of this paper propose two experiments to understand the degree to which an LLM returns hallucinated outputs, and whether it is able to recognize hallucinated responses once it is subjected to external knowledge [33].

The experiments are conducted using GPT-3.5-Turbo as language model while MS MARCO was used as dataset [34]. MS MARCO is a benchmark dataset for information retrieval, question-answering, and natural language generation developed by Microsoft, and it is based on real queries made by users on Bing. In addition, two different retrieval methods were used: the first is based on Okapi BM25 [35], a ranking function used in search engines to estimate the relevance of documents with a given query, while the second is based on a more sophisticated neural retrieval method consisting of a document retrieval part and a re-ranking part.

The first experiment consists of the following steps:

1. Submit a question to the LLM, and prompt it to answer it.
2. Combine the query with the answer generated by the model, and use the result to query a corpus of passages expected to contain the supporting evidence.

3. Validate the generated response by prompting the model to compare it with the knowledge retrieved using the combined query.

Results show how the LLM classifies 93% of the responses generated as supported by the knowledge retrieved. Moreover, to ensure the reliability of the language model classifications, answers were randomly sampled and manually inspected. The manual inspection reveals an accuracy of 80% for answers classified as supported by knowledge while the accuracy rises to 90% for answers classified as not supported by knowledge, thus with more false positives than false negatives. Errors come from partial support of the evidence against the generated response, temporal inconsistencies or the LLM confusing the generated response with the evidence, highlighting a need for better prompt engineering techniques.

The second experiment aims to evaluate the generated response at a more granular level, breaking it down into a series of factual statements, and prompting the LLM to evaluate each statement individually. The steps followed in the second experiment can be summarized as follows:

1. Prompting the LLM to answer the question.
2. Prompt the LLM to extract a list of factual statements from the generated response.
3. Combine the statements one by one with the original question, and use the result as a query to retrieve a corpus of passages expected to contain the supporting evidence.
4. Prompting the LLM to evaluate each factual statement against the retrieved knowledge. If the LLM evaluates the statement as not supported by knowledge, then it is prompted to correct the statement using the knowledge retrieved.

The results show that the model believes that approximately 85% of the factual statements it generated are supported by the knowledge retrieved. In addition, the automatic pipeline did not find hallucinations in approximately 70% of the responses generated by the model. Again, to understand whether the model is able to correctly validate an assertion against retrieved evidence, a manual inspection of a sample of results was carried out to ensure the reliability of the classification. The model achieves an accuracy of 80% for answers classified as supported by knowledge, while accuracy drops to 60% for answers classified as not supported by knowledge. The authors also point out that the LLM often acts contrary to the prompt or fabricates evidence to support statements or answers, thus risking to generate even more hallucinations.

RefChecker: Knowledge-Centric Hallucination Detection

This work developed by Amazon AWS AI addresses the problem of hallucinations in LLMs by proposing RefChecker, an automated framework that identifies hallucinations across different tasks [36]. RefChecker decomposes the statements generated by LLMs into triplets, and compares them against a reference knowledge. Furthermore, to reflect real use cases, the framework is tested on three different context settings, each with its own dataset of 100 questions:

- **Zero Context:** The model responds based solely on its own internal knowledge, without access to external sources. The dataset used in this context is the dev set of NaturalQuestions, an open domain question answering dataset which contains

human annotated responses from Wikipedia articles that are often used to train the LLMs, making it a suitable parameter for representing their internal knowledge [37]. Thus, these annotated responses are used as reference knowledge to compare the answers generated by the model.

- **Noisy Context:** The model receives additional information from an external source, but this may be partially irrelevant or contain noise. The dataset used in this context is MS MARCO, in which each question contains reference documents retrieved from the Internet that are used as input for the model, and as reference knowledge [34].
- **Accurate Context:** The model receives reliable and noise-free additional information, such as in the context of text summarization or text rewriting. For this context, databricks-dolly-15k, an open source dataset of instruction-following records, was used [38]. This dataset contains a field indicating the category of the task: thanks to this, 100 examples were sampled from the categories *summarization*, *closed_qa*, and *information_extraction*, as these categories also have a context representing a reliable reference knowledge to be compared with the response generated by the model.

The framework of RefChecker is divided into two stages: an extractor that decomposes the original text into triplets facilitating more accurate evaluation and finer detection of hallucinations, and a checker that compares the response generated by the model with the reference knowledge, classifying it as *entailment*, *neutral* or *contradiction*. A zero-tolerance approach is used to label the response generated by an LLM: if at least one triplet is regarded as contradictory by the reference knowledge, the entire response is regarded as *contradiction*, if no triplet is contradictory but at least one is *neutral* then the response is regarded as neutral while only if all triplets are supported by the knowledge then the response generated by the model is regarded as correct. In addition, several LLMs were tested both as extractors and checkers in order to find the best combination of the two.

Initially, the authors tested the approach on different granularities, such as sentence, sub-sentence, and entire response, finding that triplet splitting is the one that allows a more effective identification of hallucinations. Subsequently, the approach was tested in comparison with other benchmark methods in the field of hallucination detection, such as SelfCheckGPT (which will be discussed in the following pages) and FActScore, using Pearson's and Spearman's correlation indices as reference metrics, to calculate the correlation of the results of the methods with those obtained through human evaluation. The analyses carried out are divided into two phases, and the results obtained can be summarized as follows:

- The first analysis phase is conducted on the dataset in which SelfCheckGPT was tested. The best combination of extractor and checker turns out to be Mistral-7B + GPT-4, which outperforms SelfCheckGPT by more than two points.
- The second analysis phase was conducted on the three context settings of RefChecker. In this case, the best combination of extractor and checker turns out to be GPT-4 + GPT-4, which far outperforms the alternative methods in all contexts.

The authors conclude by stating that the experiments carried out show that this knowledge-centered approach leads to superior performance compared to prior methods.

2.3.2 Uncertainty Estimation Methods

A study conducted by the University of California shows that if the LLM is very uncertain about the response generated, then this response is more likely to be hallucinated [39]. Therefore, this category of methods aims to estimate the uncertainty of the LLM by analyzing its internal states in order to determine the probability that an output contains hallucinations. These methods solve the problem of knowledge retrieval, as they are not dependent on external sources, and also provide a numerical measure of the model's uncertainty, taking a step towards explainable decisions. However, an uncertain answer is not always wrong: indeed, the uncertainty of the model about the answer may be due to an unclear query, or the answer may actually be ambiguous. Furthermore, these methods assume access to the internal states of LLMs, which is not always guaranteed.

On Early Detection of Hallucinations in Factual Question Answering

The aim of the authors of this paper is to exploit the elements associated with text generation to detect hallucinations. Indeed, according to the authors, although hallucinated texts are very similar to truthful texts, the input, output, and internal states of the model could provide clues to help detect non-factual generations in open-ended question answering [40].

The elements of the model, called "artifacts" by the authors, considered for hallucinations detection are:

- **Softmax probabilities:** The authors hypothesize that the softmax distribution can be used to detect hallucinations. In particular, they argue that the distribution is more uniform when the model generates a hallucinated response. For the sake of simplicity, the authors focus on the probability distribution related to the generation of the first token.
- **Token attributions:** To detect hallucinations, the authors also evaluate feature attribution, that is the importance the model gives to each token provided as input (*e.g.*, to answer the question *"What is the capital of Germany?"* the model will give more importance to the words *"capital"* and *"Germany"*). The authors argue that when a model gives importance to a few words in the input, it is more likely to answer correctly, whereas if the attention is distributed over several tokens, the model is more likely to hallucinate. To generate the token attributions, the authors use the method of Integrated Gradients, as it is efficient, and can operate in any architecture [41].
- **Self-attention and hidden activations:** The authors argue that the internal states of the model might differ between correct and hallucinated responses, so they analyze two types of internal states: self-attention scores, which represent how much the model focuses on other parts of the input while processing a given token, and fully connected activation layers, which represent non-linear transformations of tokens. The authors focus particularly on the interaction between the last token of the question, and the first token of the generated response, analyzing the internal states in the last Transformer layer.

To detect hallucinations, the authors divide the dataset into train and test, and train four classifiers, each with a different set of inputs:

1. Softmax probabilities of the first token generated.

2. Integrated Gradients attributions of the first generated token.

3. Self-attention scores between the last token in the question and the first in the answer.

4. Fully-connected activations of the first generated token.

Experiments were also made by combining the artifacts on all generated tokens but no significant improvements were noted.

The experiments were conducted by analyzing the responses of three different models: OpenLLaMa, OPT-2, and Falcon. Each of these models was considered in two different dimensions, one with fewer parameters and one with more parameters (*e.g.*, Falcon was considered in its 7 billion parameter version and in its 40 billion parameter version). This was done in order to see the effects of model size on performance in hallucination detection. Moreover, to evaluate this approach, two question-answering datasets were used:

- **T-REx:** A dataset composed of triplets of relationships between entities. These relationships are converted by the authors into questions provided to the model (for example, the triplet *France - Paris - Capital of* becomes the question *"What is the capital of France?"*, and the correct answer is *"Paris"*). To evaluate the approach, the authors focus mainly on three categories: Capitals, Founders, and Places of Birth [42].

- **TriviaQA:** A dataset consisting of a set of questions, each accompanied by several possible correct answers obtained from documentary sources. The authors randomly sample 10 000 questions from this dataset, and submit them to the model; the answer of the model is considered correct if it matches one of the answers considered acceptable within the dataset [43].

Model responses were classified as either hallucinated or not by means of a heuristic rule established by the authors that states that the model response **A** is hallucinated if, given a reference response **R**, then $\mathbf{R} \not\subseteq \mathbf{A}$. Classifiers were compared using the AUC metric, which was preferred over accuracy due to the unbalance of classes within the datasets. The results show that self-attention scores and fully-connected activations are the best predictors of hallucinations since they reach over 0.70 AUC scores, outperforming Softmax and IG Attributions in all datasets and models; in fact, the latter perform only slightly better than the random model. Furthermore, although the accuracy in detecting hallucinations would seem to increase with model size, the relationship between these two properties is not entirely clear. Finally, the method was compared with the BERTScore and n-gram variants of SelfCheckGPT, outperforming them.

INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection

This paper introduces INternal States for hallucInation Detection (INSIDE), a framework that exploits the internal states of an LLM for hallucination detection [44]. The approach has two main components:

- **EigenScore:** A metric measuring the semantic divergence of the responses of the model in the embedding space. Specifically, the authors of the paper consider the embedding of the last token of the response obtained from the intermediate layer of the LLM as the embedding of the whole answer. Then, considering K responses

generated by the model from the same question, they construct the covariance matrix of response embeddings as:

$$\Sigma = Z^T \cdot J_d \cdot Z$$

where $\Sigma \in \mathbb{R}^{K \times K}$ represents the covariance matrix that captures the relationship between different answers in the embedding space, $Z \in \mathbb{R}^{d \times K}$ represents the embedding matrix of the answers, and $J_d \in \mathbb{R}^{d \times d}$ represents the centering matrix, so that the rows and columns of Σ have zero mean. At this point, EigenScore will be defined as the logarithm of the determinant of the matrix $Z + \alpha \cdot I_k$ where $\alpha \cdot I_k$ is a regularization term in order to make Z full rank. Moreover, the determinant can also be calculated by solving the eigenvalues, so EigenScore can be computed as:

$$\text{EigenScore} = \frac{1}{K} \log\det\left(\Sigma + \alpha \cdot I_k\right) = \frac{1}{K} \sum_{i=1}^{K} \lambda_i$$

This quantity is crucial because when the model is sure of the answers, then the K responses will have a high semantic similarity. Consequently, the embeddings of the answers will be highly correlated, Σ will be close to being non-maximum rank, and the eigenvalues will be very close to 0. Conversely, when the model is uncertain, it will generate answers with low semantic similarity, and the eigenvalues of Σ will be very far from 0.

- **Feature clipping:** A method to mitigate the generation of hallucinations by truncating extreme activations of internal states. In particular, the authors noted how the distribution of activations in the penultimate layer of LLMs has many extreme features, which can lead the model to generate answers in an overconfident manner, thus hallucinating responses. The feature clipping method truncates the extreme activations of the model by using a threshold function that limits the values within a range $[h_{min}, h_{max}]$ of dynamically determined thresholds, thus reducing the probability of overconfident generations, and the risk of hallucinations.

The experiments were conducted using LLaMa-7B, LLaMa-13B, and OPT-6.7B, and the approach was evaluated on four question-answering datasets: SQuAD [45], CoQA [46], TriviaQA [43], and NaturalQuestions [37]. The first two datasets are open-book question answering datasets, that are where the model has access to external knowledge to generate the answer, while the last two are closed-book question answering datasets, where the model only relies on its internal knowledge to answer. The ground truth is obtained on the basis of two correctness measures: ROUGE-L and semantic similarity. When comparing the model response with the reference response in the dataset through these two metrics, if a value above a threshold (0.5 for ROUGE-L and 0.9 for semantic similarity) is obtained, then the response is labeled as factual, otherwise it is labeled as hallucinated. EigenScore is compared against other common methods for assessing model uncertainty such as Perplexity, Length-Normalized Entropy, and Lexical Similarity. To assess the uncertainty of the model and evaluate the responses, $K = 10$ responses are generated, and compared using the uncertainty metrics specified above. The performance of the approaches in identifying hallucinations is finally assessed using AUC and Pearson correlation coefficient. The Pearson correlation coefficient is used to measure the correlation between hallucination identification metrics, and correctness measures.

The results show how EigenScore outperforms the other methods on the SQuAD, CoQA, and NaturalQuestions datasets, regardless of the model and evaluation metrics

used. Specifically, it outperforms Lexical Similarity by 5.6% on CoQA, and 8.9% on SQuAD in terms of AUC. It can also be noted how the use of larger LLMs for response generation affects the ability to detect hallucinations, given the better performance of LLaMa-13B compared to smaller models such as LLaMa-7B and OPT-6.7B. In addition, it has been noticed that the integration of feature clipping within the generation process improves hallucination detection.

LLM Internal States Reveal Hallucination Risk Faced With a Query

The main goal of this work is to understand whether LLMs can estimate the risk of hallucination before response generation, based on the query provided as input by the user [47].

Specifically, the authors design a classifier whose architecture is based on a MLP, that uses Sigmoid Linear Unit (SiLU) as activation function. According to the authors, this architecture is particularly efficient, and can handle the complexity related to hallucination risk estimation. This classifier takes as input the internal states of the model corresponding to the last token in the query, and is used for two types of analysis:

- **Seen or unseen query in training data:** When an LLM has to answer a query that it did not see during training, it is more likely to produce hallucinations given the lack of reliable information to generate the answer. Therefore, the authors aim to investigate if the model can recognize whether or not it has seen a specific query during the training phase. For this analysis, the authors construct a dataset divided into two distinct groups of queries: a set of queries that the model is very likely to have already seen, consisting of BBC news from 2020, and a set of queries that the model has not seen, containing BBC news from 2024, which is later than the training period of the model, and cannot be present in the original training set. In addition, to make the comparison fair, the authors ensure that the two datasets have similar length and comparable semantic information. Next, these queries are shown to the LLM, and its internal states related to the last token in the query are extracted. These embeddings are provided as input to the MLP, which will be responsible for classifying the queries as either seen or unseen. The main goal of this analysis is to understand whether the model is self-aware of its knowledge.
- **Hallucination risk faced with the query:** It is not enough to know whether the LLM has seen a query, but rather it is necessary to assess the risk of hallucination for a specific query. Therefore, the authors consider 15 different types of tasks, including translation, summarization, and question-answering, with a total of 700 datasets, and prompt the model to generate the answers to the query. Then, these answers are labeled as hallucinations or not by combining several metrics, including Rouge-L, NLI that evaluates the logical consistency between the generated answer and a reference answer, and QuestEval that evaluates the truthfulness of an output in a generation task. Again, the query is shown to the LLM, and its internal states are extracted, and provided to the classifier, that will classify the query as either hallucination or non-hallucination. This analysis is done to test whether the model can estimate the risk of hallucination when facing a query.

For the experiments, LLaMa2-7B was chosen as the generative model, and the approach was compared with prompt-based baselines, where the model is asked whether it is able to answer a query accurately. The methods were compared through the metrics of accuracy and F1-score. The main results of the two analysis show how:

- The model is able to distinguish between seen and unseen data, and its internal states are a good indicator for its uncertainty, as both the accuracy and F1-score reach 80%.
- The capacity of the LLM to assess the risk of hallucination varies depending on the task: it is less effective in tasks such as translation (76.90% accuracy) but excels in tasks such as number conversion (96% accuracy).

In addition, some further analysis suggests that the deeper layers of the LLM are the best for estimating the risk of hallucination, and that the rate of hallucination varies based on the task, with open tasks such as title generation being more at risk of hallucination.

Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models

The main objective of this work is to address the main problems of hallucination identification methods, which often rely on post-processing techniques separated from the LLM inference process, which are computationally expensive and require manually labeled data [48].

To deal with these problems, the authors of the paper propose MIND, a framework that does not need labeled data, and can detect hallucinatory content generated by the model in real time by exploiting its internal states. MIND involves two main phases:

- **Unsupervised training data generation:** MIND requires an automatic data generation and annotation method. This process begins by retrieving a set of $w_1 \dots w_n$ articles from Wikipedia. Each article w_i is truncated at a randomly chosen entity e_i, resulting in a truncated article w_i'. Next, the article w_i' is given as input to the LLM which is asked to generate a continuation G_i based on w_i', during which the internal states S_i of the model are also recorded. At this point, if the generation of the model G_i begins with the entity e_i, then G_i is labeled as non-hallucination; conversely, if G_i does not begin with e_i, and does not contain it within the text, G_i is labeled as hallucination. Finally, each article generates a data tuple $D_i = (L_i, w_i, G_i, H_i, S_i)$, which includes the LLM L_i selected for the generation, the original Wikipedia article w_i, the generated text G_i, the internal states S_i of the model during generation, and the hallucination label H_i.
- **Hallucination classifier training:** The classifier used to predict hallucinations is a MLP using the Rectified Linear Unit (ReLU) activation function. This classifier receives as input the internal states of the model, and returns as output a binary label indicating whether the content is a hallucination or not. By internal states we mean the contextualized embedding extracted from the hidden state of the last token in the last layer of the LLM, since preliminary experiments have shown that using this embedding yields the best prediction results. Once this classifier is trained, it is implemented in the LLM architecture, enabling the identification of hallucinations in real time by providing a probability that the model output is hallucinated.

The effectiveness of MIND was evaluated on HELM, a reference dataset created by the authors themselves. The creation of HELM can be summarized in the following steps:

1. A data generation phase in which LLMs were prompted to generate text as a continuation of some Wikipedia articles.

2. A human annotation phase in which annotators assessed the truthfulness of content generated by LLMs using mainly Wikipedia as the main source for fact-checking. Annotation was done by splitting the generated passage into sentences, and each sentence was labeled as factual or hallucination, and in the latter case, annotators were asked to highlight the position of the hallucination within the sentence.

The key feature of HELM is that each sentence is accompanied by the internal states of the model detected during generation. The dataset constructed in this way allows the detection of hallucinations at sentence level and at passage level. To compare HELM, some baselines were chosen, including SelfCheckGPT and SAPLMA, which are the state-of-the-art in hallucination detection. As for metrics, AUC and Pearson correlation coefficient with human annotated data were chosen.

The results show that MIND is effective for all LLMs tested, confirming the hypothesis that hallucination detection based on contextualized embeddings is valid. Moreover, MIND outperforms all reference-free baselines at both sentence and passage levels, confirming its robustness. However, in some models, the NLI variant of SelfCheckGPT outperforms MIND, even though the latency time of SelfCheckGPT is high; consequently, MIND appears to be the most balanced solution between effectiveness and speed. Finally, the authors analyze the impact of the individual components of MIND, pointing out that using customized datasets, and increasing the number of training data can lead to improved performance in hallucination detection. In contrast, the depth of the neural network has no significant impact on MIND performance.

The Internal State of an LLM Knows When it's Lying

This study is based on the idea that when an LLM processes hallucinated content, it is aware that it is doing so, and this awareness could be detected by analyzing the internal states of the LLM during generation or during the processing of the input [22].

In this paper, the authors propose Statement Accuracy Prediction based on Language Model Activations (SAPLMA), a method for establishing the truthfulness of a statement by processing the internal states of an LLM extracted from its hidden layers, through a classifier. However, it is not clear which of the internal layers contains the information about the assertion, so several layers are considered, thus developing different variants of the same algorithm. The layer considered are the last layer, which is mainly focused on the generation of the next token, the twenty-eighth, the twenty-fourth, the twentieth, and the sixteenth layer, which is the intermediate one. The experiments were conducted on two LLMs, OPT-6.7B and LLaMa2-7B, both consisting of 32 layers. The extracted internal states are then the input of a neural network consisting of three hidden layers with a ReLU activation function, and a sigmoid output, returning the probability of a statement being true or false.

The algorithm was tested on a dataset consisting of statements labeled as true or false, relating to six different categories: cities, inventions, chemical elements, animals, companies, and scientific facts. For the latter category, statement generation was done by asking ChatGPT to provide scientific facts well known to humans (*e.g.*, the sky is often cloudy when it is about to rain), and then asked to generate the exact opposite so that it becomes a false statement (*e.g.*, the sky is often sunny when it is about to rain). Instead,

for the other five categories, the authors considered property tables, and the statements were extracted in the following way:

- **True statements** were obtained by combining a subject with a real property (*e.g.*, the atomic number of hydrogen is 1).
- **False statements** were obtained by randomly permuting the elements in the table (*e.g.*, the atomic number of hydrogen is 34).

For greater reliability, for each topic in the dataset, the model is trained on all other topics, and tested on the topic in question, so that the classifier does not rely on clues provided by the topic but only on information given by the internal states of the LLM, and determines which statement it believes is true and which is false.

SAPLMA was compared against three different baselines:

1. A classifier with the same structure as the one used for SAPLMA, trained on the statement embeddings generated by BERT.
2. Few-shot learning using OPT-6.7B, to test whether the LLM knows if a statement is true or false.
3. Given a statement X, measure the probability of the sentences "it is true that X", and "it is false that X", taking the highest probability.

Experiments show that SAPLMA outperforms all baselines, and that OPT-6.7B seems to perform better using the 20-th layer. However, experiments with LLaMa2-7B show that the results are more accurate using the 16-th layer, so the optimal layer to use for SAPLMA highly depends on the LLM considered. Moreover, the results obtained with LLaMa are much better, reaching an accuracy of 82% using the 16-th layer, compared to an accuracy of 71% obtained by OPT using the 20-th layer. The authors also tested SAPLMA on statements generated by the OPT model, and labeled by human annotators: again, SAPLMA far exceeded the baselines, achieving an accuracy of 63% against just over 50% of accuracy achieved by BERT.

2.3.3 Zero-Resources and Black-Box Methods

The methods explained in the previous sections assume access to a source of knowledge or to the internal states of the model, but this is not always possible: retrieving knowledge can often be difficult, and access to the LLM's internal states is only granted for open models, while for proprietary models (*e.g.*, OpenAI's GPT-4), access to the internal structure of the model is not allowed. The zero-resources and black-box methods aim to overcome these limitations, by detecting hallucinations of an LLM through strategies for analyzing model behavior. In this way, the consistency of the generated outputs and the uncertainty of the LLM can be assessed without retrieving external knowledge or extracting the internal states of the model. These methods are applicable to all models and are often computationally efficient. However they could be unreliable in complex cases such as questions with many valid answers or multiple interpretations.

SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models

The aim of this work is to propose an effective method for detecting hallucinations in LLM that is independent of external knowledge databases, and does not require access to the probability distribution of model outputs [49].

The authors propose SelfCheckGPT, a method based on the idea that if the model has knowledge about a concept, then the sampled responses will tend to be similar. Conversely, if the model has no knowledge, the sampled responses will tend to diverge, and contradict each other. The method is based on stochastic sampling of responses: let R be the response generated by a user query, and whose factuality is to be evaluated. Starting from the same query and the same LLM, N stochastic responses $S_1 \dots S_N$ are generated. Then, each sentence of the response R is evaluated against each sample response and it is assigned a score between 0 and 1, where 0 is assigned to a factual sentence while 1 is assigned to a hallucinated sentence. For the evaluation of the factuality of sentences, the authors propose five variants:

- **SelfCheckGPT with BERTScore:** This variant exploits BERTScore, a measure of similarity between contextual embeddings obtained from models such as BERT or RoBERTa. The key idea is that if a piece of information is repeated in several sampled responses, then it is likely to be reliable, whereas if it only appears in one sample then it is likely to be a hallucination. Let $\mathcal{B}(\dots)$ be the BERTScore between two sentences, this variant calculates the average BERTScore of each sentence r_i of the response to be evaluated R with the most similar sentence from each sampled response. In formula:

$$\mathcal{S}_{BERT}(i) = 1 - \frac{1}{N}\sum_{n=1}^{N} \max_k \mathcal{B}(r_i, s_k^n))$$

  where s_k^n represents the k-th sentence in the n-th sample.

- **SelfCheckGPT with question-answering:** This variant generates questions from the answer that has to be evaluated. Subsequently, an independent response system answers these questions based on the stochastically generated answers. If a piece of information is consistent across the generated samples, then the answers will be similar, while if the answers are different, the information is likely to be a hallucination. Given a question q, firstly the system answers using the i-th sentence r_i, and then the other sampled responses. Then, the answers are compared, and an inconsistency score is calculated for the sentence r_i and the question q, based on the number of matching answers. In formula:

$$\mathcal{S}_{QA}(i, q) = \frac{N_m}{N_m + N_n}$$

  where N_n and N_m are the number of responses that match and the number of responses that do not match, respectively. Finally, the inconsistency score for the i-th sentence will be the average inconsistency score across all the questions.

- **SelfCheckGPT with n-gram:** This variant uses the sampled responses to create a new language model that approximates the LLM, in order to estimate its token probabilities. The main idea is that if the number N of sampled responses were

very large, then the new model would approximate the language distribution of the original LLM, and in this way one could compute the probability of a sentence within the answer R; if this sentence had a very low probability, then it could be a hallucination. The probability of a sentence i within the response R will be the average of the probabilities of the tokens within the sentence. In formula:

$$\mathcal{S}^{\text{Avg}}_{n\text{-gram}}(i) = -\frac{1}{J}\sum_{j}\log\tilde{p}_{ij}$$

where $\tilde{p}_{ij}$ is the probability of the j-th token in the i-th sentence.

- **SelfCheckGPT with Natural Language Inference (NLI):** The goal of this variant is to determine whether a sentence within the response R is contradicted or entailed by the sampled responses using a pre-trained model for NLI tasks. The contradiction probability is calculated as:

$$P(\text{contradict}|r_i, S^n) = \frac{\exp(z_c)}{\exp(z_c) + \exp(z_e)}$$

where z_e and z_c are the logits for the entailment and contradiction classes. Thus, the score of the i-th sentence of the answer R will be the average of the contradiction probability with respect to the sampled responses. In formula:

$$\mathcal{S}_{NLI}(i) = \frac{1}{N}\sum_{i=1}^{N} P(\text{contradict}|r_i, S^n)$$

and if this value is high, it means that the sentence is often contradicted by the other samples, and could be a hallucination.

- **SelfCheckGPT with LLM prompt:** This variant uses an LLM to assess the consistency of a sentence with respect to the sampled responses, by asking the model whether the i-th sentence of R is supported by the sampled response S^n. The model will be constrained to answer yes or no, and an affirmative answer will be mapped into 0 while a negative answer will be mapped into 1. Thus, the inconsistency score of the sentence will be the average of all scores obtained on the sampled answers. In formula:

$$\mathcal{S}_{\text{Prompt}}(i) = \frac{1}{N}\sum_{n=1}^{N} x_i^n$$

where x_i^n represents the score obtained by the i-th sentence when compared against the sampled response S^n.

SelfCheckGPT with its variants was tested on a dataset created using an LLM to generate Wikipedia articles, and manually annotating them to check for non-factual information. Starting with WikiBio [50], a dataset containing the first paragraph of Wikipedia articles related to a specific concept, GPT-3 was asked to generate a new article about the concept. Subsequently, each sentence of the new article was classified in three different levels of factuality:

- **Major inaccurate:** The sentence is completely false.
- **Minor inaccurate:** The sentence contains partial non-factual information but it is still related to the topic.

- **Accurate:** The sentence is factual.

Finally, the factuality of the entire generated article is calculated by averaging the factuality of its sentences.

For the experiments, GPT-3 was used for the generation of the response sample ($N = 20$ and temperature $= 1.0$ for stochasticity), while as LLM to be used for the prompt variant, the authors considered both GPT-3 and GPT-3.5-Turbo. Initially, a sentence-level hallucination detection analysis was conducted, dividing the sentences of the passages into non-factual (major-inaccurate and minor-inaccurate) and factual. This analysis was performed taking AUC precision (AUC-PR) as the evaluation metric. From the results, it was observed that the prompt version of SelfCheckGPT is the most efficient but also the most computationally expensive; however, SelfCheckGPT with NLI comes very close to its performance, being the best compromise between effectiveness and efficiency. In addition, the n-gram variant is also effective, but it was observed that as n increases, performance decreases.

After analyzing the effectiveness of SelfCheckGPT in identifying sentence factuality, the authors shift the focus to the evaluation of the factuality of an entire text passage. The inconsistency score of a passage is calculated as the average of the inconsistency scores of the sentences it contains, and Spearman's and Pearson's correlation indices were considered as evaluation metrics. Again, the prompt version proved to be the best, obtaining the highest correlation with human annotations, followed by SelfCheckGPT with NLI. Further studies were conducted by varying the number of the sampled responses, and it was found that as N increases, the performance of the method saturates: the optimal N was considered to be between 4 and 6.

SAC^3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency

The aim of this research is to provide a method for identifying hallucinations that overcomes certain limitations that the authors have identified with regard to self-consistency-based methods. Indeed, a model could generate erroneous but consistent answers, and therefore a method based on self-consistency would fail to detect this type of hallucination. Furthermore, an LLM could generate factual content, but as the temperature increases, the responses could become erroneous and inconsistent, thus indicating a false positive [51].

The authors propose SAC^3, a sampling-based method that exploits a semantic cross-check mechanism, mainly based on three steps:

1. Given an initial question, an LLM is used to construct several semantic variants of the same question. Furthermore, semantic equivalence is checked in pairs, and if a question is found to be not equivalent to the original, it is discarded.

2. Given the answer s_0 to the initial question generated by an LLM target $\mathcal{T}$, the goal is to identify whether s_0 is hallucinated. A LLM verifier $\mathcal{V}$ is considered, and the models are prompted to answer the original question and the semantically equivalent questions generated in the previous step. Next, using the LLM verifier $\mathcal{V}$ and the LLM target $\mathcal{T}$, samples of answers are stochastically extracted both from the original question s_0 and from the semantically equivalent questions.

3. The last stage is divided into three checks: first, using the target model $\mathcal{T}$, it is checked whether, by asking the same question, the answer will always be the same; if so, the model will give consistent answers, otherwise there may be inconsistencies and a potential hallucination. Next, it is checked whether, using semantically equivalent questions, the model continues to give consistent answers. Finally, it is assessed whether, by asking the same question to two different models, the answer given is the same; if not, there could be inconsistencies, and the answer s_0 could be a hallucination. For each of these checks, scores are assigned to s_0 that will be combined to obtain a final score: if this score is high, then s_0 is likely to be a hallucination while if it is low, it means that the answer is probably reliable.

The method was evaluated on four different datasets:

- **Prime number:** 500 questions about whether a number between 1 000 and 20,000 is prime.
- **Senator search:** 500 questions about the presence of a U.S. senator with a specific university.
- **HotpotQA:** 250 questions randomly drawn from HotpotQA [52]. Answers are generated with GPT-3.5-Turbo and are manually annotated.
- **NaturalQuestions:** 250 questions randomly drawns from NaturalQuestions [37] where answers are created with the same settings as HotpotQA.

The experiments were conducted using GPT-3.5-Turbo as the target LLM, and Falcon-7B-Instruct and Guanaco-33B were tested as verifier models, while AUC was chosen as the metric to evaluate the approach. SAC^3 is compared against the self-consistency method, which involves sampling the answers provided by the LLM and semantically comparing them in order to check for consistency. The results show that SAC^3 clearly outperforms the self-consistency baseline in all the datasets considered: on the prime number and senator search datasets, it reaches 99.4% and 97.0% AUC, respectively, while on the HotpotQA and NaturalQuestions datasets, it reaches 88.0% and 77.2% AUC, respectively. Further analysis shows that the use of 2-4 perturbed questions leads to a fair trade-off between effectiveness and efficiency.

LM vs LM: Detecting Factual Errors via Cross Examination

This research, inspired by legal truth-seeking mechanisms, aims to detect hallucinations by proposing a framework involving cross-examination between models [53].

The method involves the use of an examined LLM, who generated the information, and an examiner LLM who asks questions and checks the consistency of the answers, and these models interact guided by specific prompts in a multi-turn dialogue. The key idea is that an incorrect statement is likely to lead to inconsistencies with other statements generated by the model. Through cross-examination, the examiner tries to detect these inconsistencies, facilitating the identification of factual errors. Specifically, the process is divided into three steps:

1. The examiner is given the task of checking the truthfulness of a statement, and formulates a series of questions to be answered by the model examined.

2. After receiving the answers, the examiner decides whether further questions are necessary to clarify any doubts; if so, the examiner generates new questions, and the examinee answers them. This interaction continues until the examiner has clarified the doubts or until a maximum number of turns of interaction has been reached.

3. The examiner makes a judgment on the factuality of the statement, rejecting or accepting it, based on the answers given by the model examined. If the examiner does not provide a clear answer, the validity of the statement is considered as rejected.

The approach is tested on five different datasets:

- **Lama:** A dataset containing statements structured as subject-relation-object [54].
- **TriviaQA:** Dataset of general culture questions on various topics [43].
- **NaturalQuestions:** Dataset consisting of questions posed by real users [37].
- **PopQA:** Dataset of questions obtained by converting a knowledge tuple retrieved from Wikidata [55].
- **Falsehoods:** Dataset consisting only false statements and created by the authors themselves. Given a question, the LLM is asked to produce an incorrect but plausible answer, and after verifying that the answer is indeed incorrect, it is included in the dataset.

The experiments were conducted using LLaMa-7B, GPT-3.5-Turbo, and GPT-3, the evaluation metrics considered are accuracy, recall, and F1-score, and the ground truth is identified by comparing the LLM response with the correct response in the dataset. To apply the method, the response of the examined LLM is converted into a statement. The results show how the proposed approach far exceeds the baselines considered on all datasets: in particular, the method excels in recall, which means that it is able to detect erroneous claims generated by the LLM.

InterrogateLLM: Zero-Resource Hallucination Detection in LLM-Generated Answers

The authors of this paper propose InterrogateLLM, a framework that detects LLM-generated hallucinations using a verification mechanism based on the reconstruction of the original query [56]. The main idea is that, starting from a hallucinated content, the model will have difficulty reconstructing the original query correctly. Specifically, InterrogateLLM is divided into the following steps:

1. An LLM generates an answer A^* to a question Q.

2. Using the generated answer A^*, one or more LLMs are asked to reconstruct the query that originated the answer, resulting in a set of reconstructed queries Q^*.

3. The reconstructed queries Q^* are represented as vectors in an embedding space, and are compared with the original query Q, using the cosine similarity function.

4. The average of the results is computed to obtain a final score, which is then compared with a threshold τ. If the score exceeds τ, then the queries are similar to the original query, and A^* is likely to be factual. Conversely, if the similarity score is lower than τ, it means that the queries diverge from the original, and it is likely that A^* is a hallucination.

InterrogateLLM is evaluated on three datasets adapted by the authors:

- **Movies dataset:** Dataset containing information on movies released up to 2017 [57]. Given the title and the release year of a movie, an LLM was prompted to generate the entire cast, and if the Intersection Over Union (IOU) metric was less than 80%, the response was labeled as hallucination.
- **Books dataset:** Dataset containing information on over 200,000 books [58]. Given the title of a book, an LLM was prompted to generate the year of publication and author, and if these do not match the actual data, the answer is labeled as hallucination.
- **Global Country Information:** Dataset containing information on 181 countries [59]. The model is prompted to determine the capital of a country, and if this does not match, the answer is labeled as a hallucination.

Experiments were conducted using GPT-3, LLaMa-2-7B, and LLaMa-2-13B, and the method is compared against several baselines including SelfCheckGPT with LLM prompt. For the evaluation of the approaches, the authors considered the metrics of AUC and balanced accuracy. The results show how InterrogateLLM outperforms all baselines, particularly compared to SelfCheckGPT with LLM prompt, highlighting its limitations. Indeed, in many cases, the stochastic responses generated by SelfCheckGPT contains the same error as the original response, necessitating the query reconstruction approach introduced by InterrogateLLM to reduce the risk of accidentally confirming hallucinations. The authors also explore how the temperature of the LLM affects the performance of the approach, increasing it in a dynamic way during the process. This allows for more exploration, enabling the model to be more creative during the reconstruction of the query, and at the same time, allows for more robust detection, highlighting discrepancies between the reconstructed queries and the original query.

Chapter 3

Methodology

After outlining the main categories of hallucination detection approaches in the literature, this section will provide a detailed analysis of specific methods and potential variations in the approaches. The main goal is to compare different detection strategies within the categories discussed previously, and evaluate their effectiveness across various datasets and models. For this analysis, one approach from each of the three hallucination detection categories, along with a supervised classification method, has been considered. Specifically:

- **Supervised text classification:** Models were trained to classify input text as factual or hallucinated, using labeled data for supervised learning.
- **Uncertainty estimation:** Drawing inspiration from strategies that estimate uncertainty of the LLM using its hidden states, internal states were extracted from multiple layers of an open-source model. These extracted representations were then fed into binary classifiers to distinguish between hallucinated and non-hallucinated instances. Various machine learning models were explored as classifiers, ranging from SVM and Logistic Regression to Deep Neural Networks.
- **Zero-resources and black-box:** From this category, SelfCheckGPT was selected as it is considered the state-of-the-art approach for zero-resources and black-box hallucination detection. For the analysis, the variants of SelfCheckGPT that were tested are NLI, BERTScore, and LLM prompting.
- **Knowledge retrieval:** Inspired by the knowledge retrieval approaches, an automated retrieval method was developed using the Google Search API. The retrieved information was then integrated into a few-shot prompting strategy and into SelfCheckGPT, in order to assess its impact and potential performance improvements obtained from the introduction of the knowledge.

Additionally, this section will detail the models considered, experimental setups, prompts used, and modifications tested for each approach.

3.1 Supervised Text Classification

The first approach considered in this comparison is a text classification method, in which pre-trained Transformer-based models are fine-tuned for hallucination detection, and employed to classify text as either factual or hallucinated. Figure 3.1 illustrates the pipeline of the text classification approach, which consists of the following steps:

1. The input text is provided to the model.
2. The model processes the text independently, generating a corresponding vector representation.
3. Based on this representation, the model classifies the text as either true (factual) or false (hallucinated).

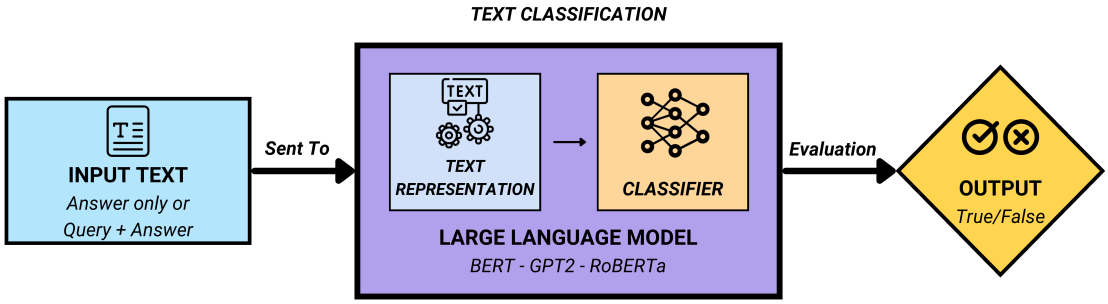

Figure 3.1: Pipeline of the text classification method for hallucination detection.

The analysis was carried out under two distinct scenarios:

- **Only answer:** In this scenario, the model is provided solely with the statement to be evaluated, assessing its factuality in isolation.
- **Query + answer:** In this scenario, both the user's query and the generated response are provided as input to the model, separated by a single space character.

This distinction aims to investigate whether the inclusion of additional contextual information enhances the model's ability to assess factuality, and extract more relevant textual features when both the query and the response are available.

3.1.1 Models

For the text classification approach, various Transformer-based models were selected for the construction of the vector representation of the text, because they can capture contextual relationships and identify inconsistencies within texts. In particular, pre-trained models were fine-tuned for the specific task of hallucination detection. The following sections provide an overview of the models used and their main characteristics.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language model developed by Google AI in 2018 for better understanding and processing of natural language [60]. Its key innovation lies in its bidirectional context analysis, which allows it to consider both preceding and following words when interpreting a given term: indeed, unlike earlier models that processed text in a unidirectional manner, the bidirectional approach of BERT enables a deeper understanding of linguistic context. This feature makes it particularly suitable for classification and sentiment analysis purposes, and as a result, BERT has achieved state-of-the-art performance across a wide range of NLP applications, also outperforming task-specific architectures.

The training phase of BERT takes place in two main stages:

- **Pre-training:** During the pre-training phase, BERT is trained on two unsupervised tasks using large amounts of text from BookCorpus [61] and the English Wikipedia. The first task, Masked Language Modeling (MLM), involves randomly masking certain words in the input, requiring the model to predict the hidden words based on the surrounding context. The second task, Next Sentence Prediction (NSP), provides the model with pairs of sentences, and tasks it with determining whether these sentences appear consecutively in the training corpus.
- **Fine-tuning:** After the pre-training phase, BERT undergoes additional fine-tuning on task-specific datasets to adapt it for various applications, such as text classification and Named Entity Recognition (NER).

This combination of pre-training and fine-tuning, allows BERT to excel in a variety of NLP applications.

GPT-2

GPT-2 (Generative Pre-trained Transformer 2) is a Transformer-based language model developed by OpenAI in 2019 [62]. With 1.5 billion parameters, GPT-2 was one of the largest language models of its time, demonstrating remarkable accuracy across various NLP tasks. Unlike BERT, GPT-2 employs a unidirectional autoregressive architecture, meaning that text is processed based only on the context preceding the analyzed word. This architecture allows it to excel at predicting the next token within a sequence. Furthermore, the architecture is based on attention mechanisms that allow the model to focus on different parts of the input during generation, improving the consistency and relevance of the generated text. As a result, GPT-2 is particular effective in tasks for text generation tasks such as summarization, translation, and question answering, but can also be adapted to other NLP tasks such as text classification.

GPT-2 was trained on WebText, a dataset curated by OpenAI that comprises Reddit posts with a minimum of three upvotes. This dataset was specifically designed to provide high-quality and diverse textual data for training the language model.

RoBERTa

RoBERTa (Robustly Optimized BERT Approach) is an enhanced version of BERT developed by Facebook AI in 2019. It improves upon BERT by expanding the training dataset,

and refining the training procedure to achieve superior performance across a wide range of NLP tasks. Although RoBERTa retains the core architecture of BERT, including self-attention mechanisms to process textual relationships and bidirectional context modeling, it introduces key modifications to its training strategy. These changes include:

- **Removal of NSP objective:** BERT is trained to predict whether two sentences appear consecutively within the training corpus. RoBERTa removes this phase of the training by focusing only on MLM, as experiments showed that performance did not benefit significantly.
- **Dynamic masking in MLM:** In BERT, input masks are only applied once and remain fixed throughout the training. In contrast, RoBERTa introduces dynamic masking where the masked inputs change at each training epoch, improving generalization.
- **Larger batch sizes:** RoBERTa uses larger batch sizes (8 000 sequences per batch), improving performance during MLM.

Furthermore, in contrast to the 16GB dataset used for training BERT, RoBERTa was trained on more than 160GB of uncompressed text, significantly expanding the data available for learning. These datasets include:

- **BookCorpus and English Wikipedia:** Original data used to train BERT [61].
- **Common Crawl News:** A collection of over 63 million English news articles, gathered between September 2016 and February 2019 [63].
- **OpenWebText:** An open-source replication of the WebText corpus used to train GPT-2, consisting of Reddit posts with at least three upvotes [64].
- **Stories:** A dataset derived from Common Crawl, designed to enhance commonsense reasoning and language modeling [65].

The combination of these expanded training datasets, along with improvements in the training process, enables RoBERTa to surpass BERT's performance on multiple NLP benchmarks.

3.1.2 Experimental Setup

The models chosen for classification, along with their corresponding tokenizers for sentence processing, were obtained from Hugging Face using the Hugging Face Transformers library [66]. Specifically, the models were imported utilizing the *AutoModelForSequenceClassification* method from the Transformers library, which added a classification head at the final layer of the model. This configuration allowed the textual representations generated by the models to be processed by the classifier, which subsequently categorized them as either hallucinated or factual. In particular, the specific versions of the models selected are:

- **BERT:** The chosen version is *bert-base-uncased*, which consists of 110 million parameters, and ignores case sensitivity. This decision was made because capitalization is not a crucial factor in detecting hallucinations.[1]

[1] `https://huggingface.co/google-bert/bert-base-uncased`

- **GPT-2:** The *gpt2* version was used, which consists of 137 million parameters.[2]
- **RoBERTa:** The selected model is *roberta-base*, which has 125 million parameters.[3]

The model sizes were chosen to ensure reasonable training times while maintaining a comparable scale across models, allowing for a fair performance comparison.

Each input sequence was either truncated or padded to a fixed length of 512 tokens, and processed in batches of 8 during training and evaluation. The models were trained for 5 epochs using the AdamW optimizer, with a learning rate of $2 \times 10^{-5}$and a weight decay of 10^{-2}. The entire training procedure was executed on a Tesla T4 GPU using Google Colab.

3.2 Uncertainty Estimation

Inspired by the paper that introduced SAPLMA [22], along with other methodologies for estimating uncertainty in LLMs through the analysis of internal states, this approach consists of providing an LLM with the statement to be evaluated. Internal states are then extracted from multiple layers of the model, and used as input for a binary classifier to determine the factuality of the statement, leveraging the representations derived from the internal activations of the LLM. Figure 3.2 illustrates the pipeline of this approach, which is specifically structured as follows:

1. The text is provided as input to the model, which processes it, and generates an output.
2. While the model's generated output is discarded, internal states from various hidden layers of the LLM are extracted.
3. The extracted internal states are passed to a binary classifier, which assesses the truthfulness of the statement, and categorizes it as either true (factual) or false (hallucinated).

The analysis was performed under two distinct scenarios:

- **Only answer:** In this setting, the LLM processes only the statement whose truthfulness needs to be evaluated.
- **Query + answer:** Here, the model receives a sequence comprising both the query and the generated response, separated by a single space character.

This distinction aims to assess whether providing both the question and the answer enables the model to extract additional contextual information, enhance its confidence in the statement's truthfulness, and generate richer internal representations that can aid the classifier in determining the factuality of the statement.

[2] `https://huggingface.co/openai-community/gpt2`
[3] `https://huggingface.co/FacebookAI/roberta-base`

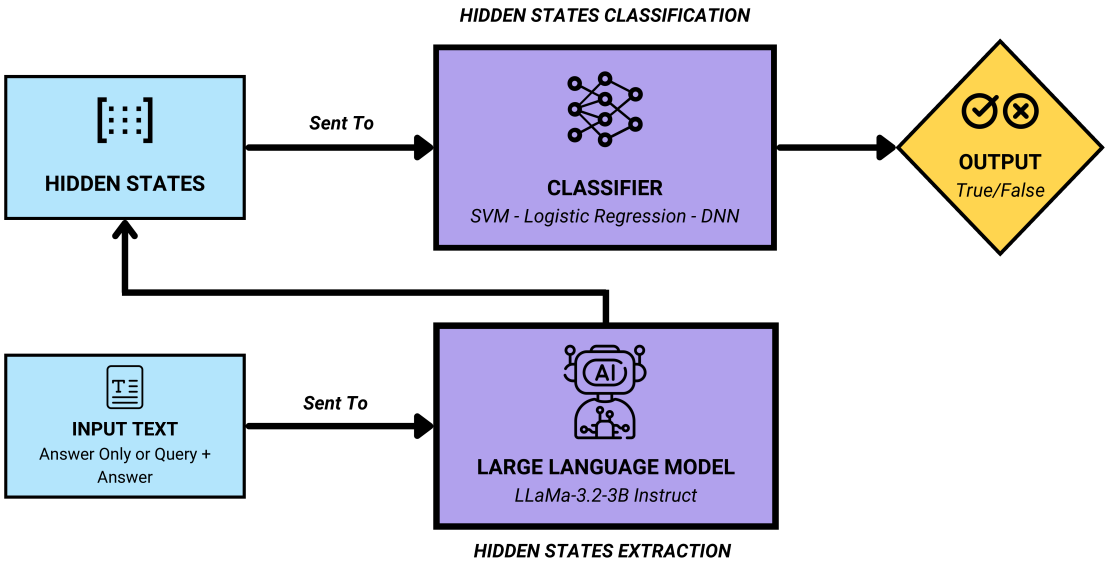

Figure 3.2: Pipeline of the hallucination detection approach based on LLM uncertainty estimation through the analysis of its internal states.

3.2.1 Models

This section provides a detailed overview of the models employed to classify the internal states of the LLM. In the original SAPLMA framework, classification was performed using a neural network comprising three hidden layers with a progressively decreasing number of units (256, 128, 64), each utilizing the ReLU activation function. However, in this study, additional machine learning classifiers, as well as deeper neural networks exceeding the complexity of the model used in SAPLMA, were explored to assess potential variations in hallucination detection performance.

Traditional Classification Models

In the implementation of classifiers, traditional machine learning models were also considered, as they remain widely employed in binary classification tasks. These models offer a straightforward and computationally efficient implementation while achieving robust performance across various classification problems.

The first model examined in this study is Support Vector Machine (SVM), which is a supervised learning algorithm primarily employed for classification tasks. Its main goal is to solve a constrained optimization problem to identify the optimal hyperplane that separates the different classes within a dataset. This hyperplane is chosen not only to achieve class separation but also to maximize the margin, defined as the distance between the hyperplane and the nearest data points from each class, known as support vectors [67].

A distinctive feature of SVM is its ability to handle non-linear classification problems through the use of kernel functions. These functions transform the original feature space into a higher-dimensional space, where the data can be linearly separable. The most common kernel functions are the Radial Basis Function (RBF), sigmoid, and polynomial

kernels, each offering different advantages depending on the data distribution [68]. Although SVM was initially developed in the 1960s, and later refined in the 1990s, they remain highly considered for classification purposes, despite the advent of more advanced techniques. Their versatility and interpretability continue to make them a valuable tool for various real-world applications, including image classification and biological research, such as protein classification.

The second traditional model considered is logistic regression, which is a statistical approach that belongs to the category of supervised learning methods, and is primarily employed for binary classification tasks. By applying the sigmoid function, it transforms a linear combination of input features into a probability value ranging between 0 and 1, representing the likelihood that the input belongs to a specific class. The final classification decision is made by comparing this probability to a predefined threshold, typically set at 0.5. Logistic regression is widely utilized in machine learning for its efficiency and interpretability. Beyond this domain, it is also extensively applied in various fields, including marketing, healthcare, and social sciences. For example, in marketing, it is used to predict customer behavior; in healthcare, it assists in disease diagnosis and risk assessment; and in social sciences, it facilitates the analysis of binary outcomes in empirical research [69] [70].

Deep Neural Networks

Deep Neural Networks (DNNs) are a class of deep learning models composed of multiple interconnected layers positioned between the input and output. These networks are designed to emulate the functioning of the human brain, leveraging artificial neurons to extract complex patterns, and effectively model nonlinear relationships within the data [71] [72].

The typical architecture of a DNN consists of:

1. **Input layer:** The initial layer that receives raw features from the dataset.
2. **Hidden layers:** A series of intermediate layers that process inputs through weighted connections and activation functions. These layers enable the network to extract meaningful patterns, and model complex nonlinear functions.
3. **Output layer:** The final layer that generates the prediction, typically using activation functions such as softmax or sigmoid. These functions transform the network's outputs into probability values, allowing the model to determine the most likely class.

DNNs offer the advantage of automatically learning features from data, eliminating the need for manual feature engineering. Additionally, they excel at capturing complex nonlinear relationships within the data, which might be challenging to identify using traditional machine learning methods. Moreover, DNNs are capable of processing unstructured data, such as text, images, and audio, making them highly effective in domains such as computer vision, speech recognition, and NLP.

3.2.2 Experimental Setup

This section outlines the methodological decisions made during the implementation process, including the selection of the LLM for internal state extraction, the layers used for

this purpose, and the implementation strategies adopted for both traditional and neural classifiers. The entire extraction procedure was executed using Google Colab, as well as the training process, using a Tesla T4 GPU.

Selection of Large Language Model and Extraction of Internal States

In the original implementation of SAPLMA, the models used for extracting internal states during text processing were LLaMa-2-7B and OPT-6.7B. Both models consist of 32 layers, each containing 4096 hidden states. However, these models may now be considered outdated. Therefore, a more recent LLM, LLaMa-3.2, was selected for internal state extraction. Specifically, the *Llama-3.2-3B* version, available on Hugging Face, was chosen.[4] This model comprises 3 billion parameters, 28 layers, and 3072 hidden states per layer. To enhance efficiency and reduce computational time, the model was quantized before use.

According to [22], the specific layers that contain the most relevant information regarding the truthfulness of a statement remain uncertain. Thus, in SAPLMA, internal states were extracted from the last hidden layer, the fourth, eighth, and sixteenth layers from the end. The methodology proposed in this study aligns with the approach proposed in [22]: indeed, hidden activations are extracted from layers positioned at the same distances from the end as those extracted in SAPLMA. However, while [22] employed LLaMa-2-7B and OPT-6.7B—both with 32 layers, leading to the selection of layers 16, 24, 28, and 32—the LLM used in this study has 28 layers, resulting in the selection of layers 12, 20, 24, and 28. Although this configuration does not perfectly match [22], since SAPLMA does not explore the layers preceding the middle, it was maintained since these layers may contain valuable information relevant to the classification of the statement. Additionally, the classifiers were trained using two different approaches: first, by providing the internal states from each of the four layers individually, and second, by supplying all four layers simultaneously. This was done to assess whether the model could extract more information, and identify patterns more effectively when the internal states were provided concurrently. Finally, always consistently with the methodology proposed for SAPLMA, the internal states corresponding to the final token of the evaluated response were extracted.

The code used for extracting hidden activations from the language model was obtained from GitHub, and is largely based on the implementation of SAPLMA.[5] However, the original code did not support the use of quantized LLMs. To enhance efficiency in model loading, and facilitate the extraction of internal states, modifications were made to incorporate the capability to load quantized LLMs.

Traditional Classification Methods

The traditional classification methods employed in this study to classify the internal states of the LLM were implemented using the *scikit-learn* library [73]. Specifically:

- **SVM:** The selected kernel function for SVM is RBF, mathematically defined as:

$$k(x, y) = \exp\left(\|-\gamma\,(x - y)\|^2\right)$$

[4] `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`
[5] `https://github.com/balevinstein/Probes`

where $\gamma > 0$ is a parameter that must be specified. In this implementation, γ has been set to *"scale"*, which corresponds to:

$$\gamma = \frac{1}{N \times \mathrm{Var}\,(x)}$$

where N represents the number of features (3072), and $\mathrm{Var}(x)$ denotes the variance of the input instance x. Additionally, the parameter C, which determines the strength of the regularization, has been set to 1.[6]

- **Linear regression:** The l_2 penalty is applied to regulate model complexity and enhance generalization to unseen data, thereby mitigating the risk of overfitting through a regularization constraint on the model parameters. Additionally, the regulatization strength parameter C has been set to 1. The optimization algorithm selected for solving the minimization problem is *"lbfgs"*, a quasi-Newton method known for its efficiency in optimization while maintaining low computational memory requirements.[7]

Deep Neural Networks

Two different DNN architectures were employed. The first corresponds to the neural network used in the original SAPLMA implementation, consisting of three layers with a decreasing number of hidden units (256, 128, 64). Each hidden layer utilizes the ReLU activation function, while the output layer employs a sigmoid activation function.

The second neural network, in contrast, was designed with five layers, making it deeper than the original SAPLMA implementation. This modification aimed to assess whether increasing network depth could lead to improved performance, and enhance the model's ability to extract relevant patterns from the internal states. Similarly to the first architecture, this network also follows a structure with a decreasing number of hidden units (1024, 512, 256, 128, 64), with ReLU activation functions applied to the hidden layers and a sigmoid activation function in the output layer.

Both neural networks were implemented and trained using TensorFlow [74]. Each network was trained for 5 epochs with a batch size of 32 instances. The Adam optimizer was employed to optimize the model parameters, while binary cross-entropy was chosen as the loss function to effectively handle the binary classification task.

3.3 Zero-Resources and Black-Box

In the literature, SelfCheckGPT is recognized as a state-of-the-art approach for hallucination detection, and is frequently used as a benchmark for assessing newly developed methodologies. Therefore, it was selected for implementation as a representative method within the category of zero-resources and black-box approaches. In accordance with the methodology outlined in [49], the response under evaluation will be compared against a set of responses generated using the same query. The methodology adopted considers the following variants: SelfCheckGPT with BERTScore, SelfCheckGPT using a NLI model, and

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC`

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

SelfCheckGPT with LLM prompt. Figure 3.3 illustrates the pipeline of the methodology, which is structured as follows:

1. The query that generated the response under evaluation is submitted to the LLM.
2. The model is prompted to generate N response samples based on the same query.
3. The original response is segmented into individual sentences, and compared with the N sampled responses using one of the SelfCheckGPT variants.
4. Following the comparison, SelfCheckGPT assigns a hallucination score to the answer under evaluation. This score ranges from 0 to 1, where 1 indicates hallucination, and 0 signifies a factual response. This hallucination score is converted into a binary value (true or false) through a threshold function:

$$\hat{y} = \begin{cases} \text{True}, & \text{if } S_{\text{BERT}}(r) \leq \tau \\ \text{False}, & \text{if } S_{\text{BERT}}(r) > \tau \end{cases}$$

where τ is the threshold optimized to maximize accuracy against the ground truth.

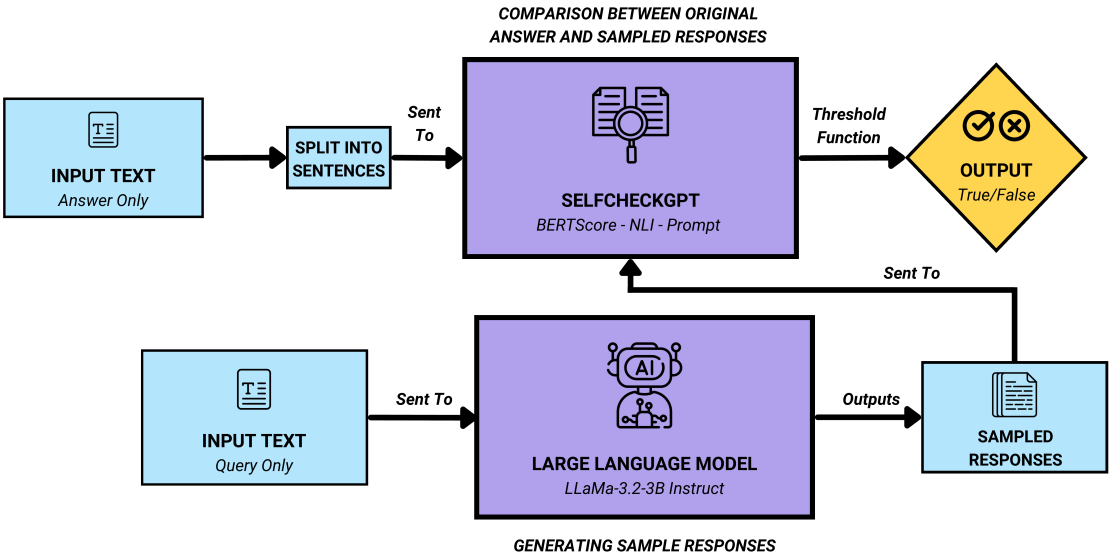

Figure 3.3: Pipeline of the zero-resources and black-box hallucination detection approach based on SelfCheckGPT.

This methodology relies on the premise that if the response under evaluation is factual, the responses generated from the same query will exhibit a high degree of similarity. In contrast, if the response contains hallucinated information, the generated responses will be inconsistent or contradictory to it, thereby indicating lower factuality.

3.3.1 SelfCheckGPT with BERTScore

Figure 3.4 details the pipeline of the SelfCheckGPT approach based on BERTscore. Considering the response r, the process is specifically divided into the following steps:

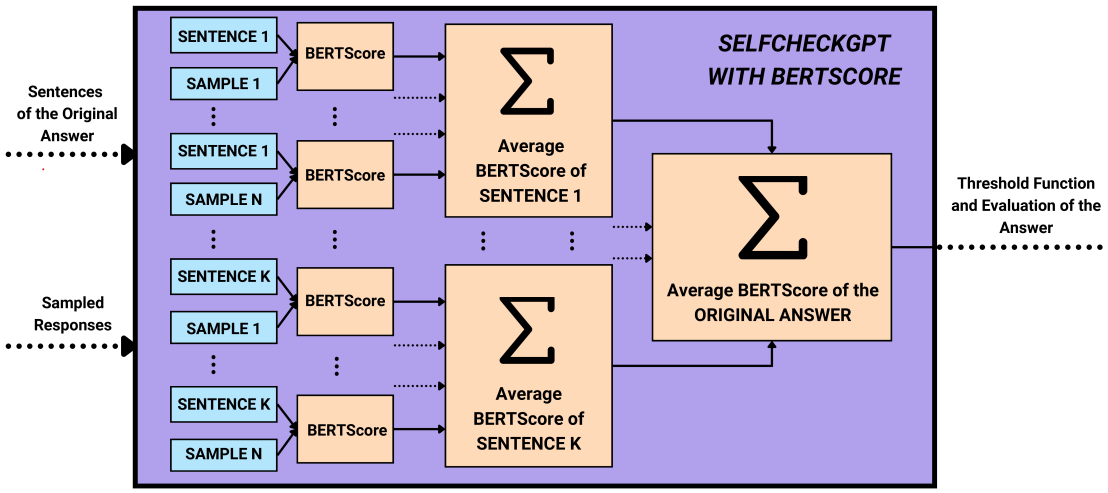

Figure 3.4: Pipeline of SelfCheckGPT based on BERTScore.

1. The response r is segmented into K sentences which are analyzed alongside N sampled responses $s_1, \ldots, s_N$.

2. Each sentence r^k from the response r is sequentially compared with each sampled response s_j, and a BERTScore is computed for each pair $\left(r^k, s_j\right)$. BERTScore is a semantic similarity metric that evaluates the correspondence between a reference response and a candidate response. It constructs contextualized embeddings using BERT, and calculates similarity via cosine similarity [75].

3. Once r^k has been compared with all the sampled response s_j for all $j = 1, \ldots, N$, the average hallucination score for the sentence r^k is computed as follows:

$$\mathcal{S}_{BERT}(r^k) = \frac{1}{N} \sum_{j=1}^{N} \left(1 - \mathcal{B}\left(s_j, r^k\right)\right)$$

   where $\mathcal{B}(.,.) \in [0,1]$ represents the BERTScore between two sentences. Since BERTScore quantifies the similarity between two sentences, it is inverted to ensure that $\mathcal{S}_{BERT}(r^k)$ accurately represents the hallucination score for the sentence r^k.

4. After computing $\mathcal{S}_{BERT}(r^k)$ for all $k = 1, \ldots, K$, the total hallucination score for the response r is obtained by averaging the hallucination scores across all sentences:

$$\mathcal{S}_{BERT}(r) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{S}_{BERT}\left(r^k\right)$$

   This final score ranges between 0 and 1, where 0 indicates factuality, and 1 indicates hallucination, thus serving as hallucination score for the response r.

5. The computed hallucination score is then mapped to a binary value (true/false) using a threshold function, determining whether the response r is classified as factual or hallucinatory.

The underlying principle of this approach is that if the original response is factual, then generating additional responses using the same query should yield content that is semantically similar to the original response. In this scenario, BERTScore will effectively capture and quantify this similarity.

3.3.2 SelfCheckGPT with NLI

Natural Language Inference (NLI) is the task of determining the logical relationship between a given premise and hypothesis. Specifically, the goal is to classify the hypothesis into one of three categories:

- **Entailment:** The hypothesis logically follows from the premise.
- **Contradiction:** The hypothesis contradicts the premise.
- **Neutral:** There is no clear logical relationship between the premise and the hypothesis.

For example, given the premise *"A man is playing the guitar"*, the hypothesis *"Someone is making music"* falls under entailment, as it logically follows from the premise. Conversely, if the hypothesis was *"Someone is playing the piano"*, it would fall under contradiction, as it contradicts the original statement. NLI plays a crucial role in various NLP applications, including text summarization, machine translation, and fact-checking, where it is essential to determine whether a given meaning can be inferred from different textual variations.

Building on the principles of the NLI task, [49] proposes to use the contradiction score as a measure of hallucination in a given response. Figure 3.5 illustrates the pipeline of the SelfCheckGPT approach based on NLI. Given a response r to be evaluated, the methodology follows these steps:

1. The response r is segmented into K sentences which are analyzed alongside N sampled responses $s_1, \ldots, s_N$.
2. Each sentence r^k is processed by an NLI model together with a sampled response s_j, where r^k serves as the premise, and s_j as the hypothesis. The NLI model calculates the probability that s_j contradicts r^k, which is formally expressed as:

$$P\left(\text{contradict}|r^k, s_j\right) = \frac{\exp\left(z_c\right)}{\exp\left(z_c\right) + \exp\left(z_e\right)} \in [0, 1]$$

   where z_e and z_c denote the logits corresponding to the entailment and contradiction classes, respectively.

3. Once r^k has been compared with all the sampled response s_j for all $j = 1, \ldots, N$, the average hallucination score for the sentence r^k is computed as:

$$\mathcal{S}_{NLI}(r^k) = \frac{1}{N} \sum_{j=1}^{N} P\left(\text{contradict}|r^k, s_j\right)$$

4. After computing $\mathcal{S}_{NLI}(r^k)$ for all $k = 1, \ldots, K$, the overall hallucination score for the response r is determined by averaging the individual sentence scores:

$$\mathcal{S}_{NLI}(r) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{S}_{NLI}\left(r^k\right)$$

5. The computed hallucination score is then mapped to a binary value (true/false) using a threshold function, determining whether the response r is classified as factual or hallucinatory.

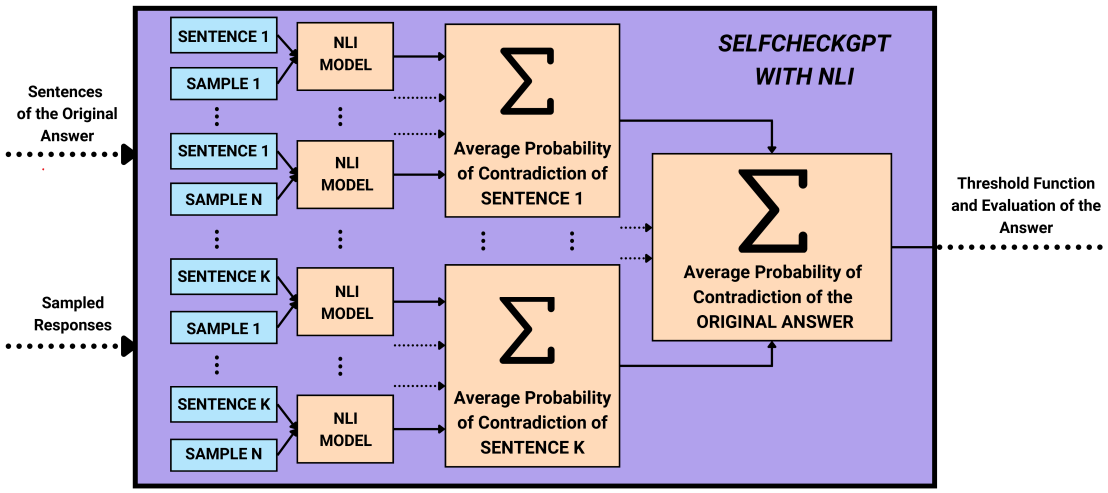

Figure 3.5: Pipeline of SelfCheckGPT based on NLI.

This approach is founded on the premise that if the response under evaluation is factual, the sampled responses are more likely to support it, resulting in a low probability of contradiction. Conversely, if the response is hallucinated, the sampled responses will tend to contradict it, leading to a higher probability of contradiction.

3.3.3 SelfCheckGPT with LLM Prompt

The SelfCheckGPT approach with LLM prompt utilizes a LLM to assess whether a given response sentence is supported by the sampled responses. Figure 3.6 provides a detailed representation of the SelfCheckGPT pipeline based on LLM prompts. Given a response r, the methodology is structured as follows:

1. The response r is segmented into K sentences which are analyzed alongside N sampled responses $s_1, \ldots, s_N$.

2. Using a specifically designed prompt, the model is queried to determine whether the sentence r^k is supported by the sampled response s_j. The output generated by the LLM is then mapped to a numerical score x_j^k according to the following conversion:

$$x_j^k = \begin{cases} 0, & \text{if the LLM answers “Yes”} \\ 1, & \text{if the LLM answers “No”} \\ 0.5, & \text{if the LLM answers neither “Yes” nor “No”} \end{cases}$$

3. Once r^k has been compared with all the sampled response s_j for all $j = 1, \ldots, N$,

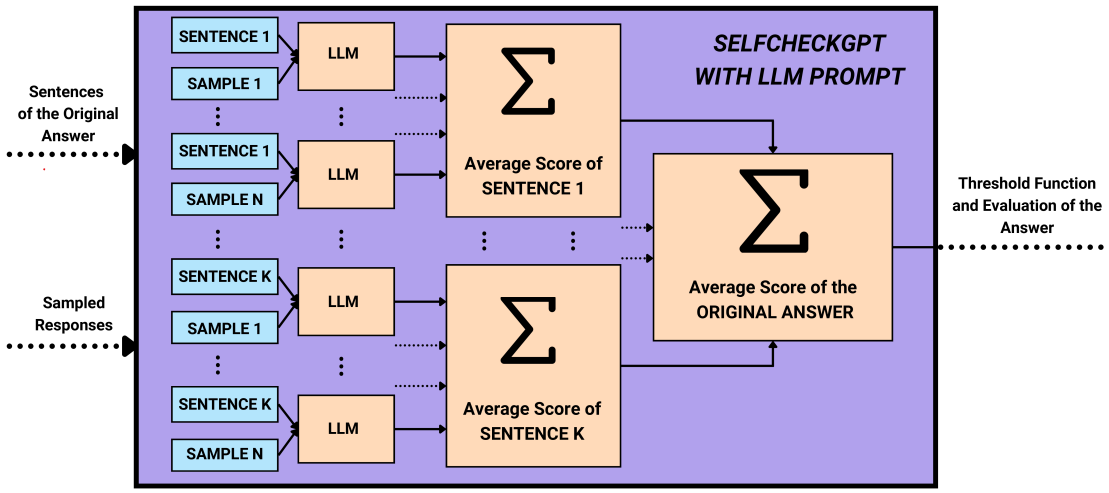

Figure 3.6: Pipeline of SelfCheckGPT based on LLM prompt.

the average hallucination score for the sentence r^k is computed as:

$$\mathcal{S}_{LLM}(r^k) = \frac{1}{N}\sum_{j=1}^{N} x_j^k$$

4. After computing $\mathcal{S}_{LLM}(r^k)$ for all $k = 1, \ldots, K$, the overall hallucination score for the response r is determined by averaging the individual sentence scores:

$$\mathcal{S}_{LLM}(r) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{S}_{LLM}\left(r^k\right)$$

5. The computed hallucination score is then mapped to a binary value (true/false) using a threshold function, determining whether the response r is classified as factual or hallucinatory.

3.3.4 Models

In the original implementation of SelfCheckGPT, the authors utilized GPT-3 to generate the sampled responses [49]. However, in this methodology, a model from the LLaMa family, developed by Meta, was employed. A distinctive feature of this family of models is its ability to achieve state-of-the-art performance across various NLP tasks despite being trained exclusively on publicly available datasets, although the specific datasets used were not disclosed [76].

In particular, LLaMa-3.2, released by Meta in September 2024, was adopted for this study. The architecture of LLaMa 3 models, apart from minor differences, closely resembles that of GPT-3, as both are based on decoder-only Transformer architectures. The exceptional performance of LLaMa 3 can be attributed to the high quality and diversity

of the training data, which significantly enhances the model's capabilities, enabling it to achieve results comparable to those of leading state-of-the-art models in various NLP tasks [77].

3.3.5 Experimental Setup

This section provides a detailed discussion of the implementation choices made, including the selection of the LLM for SelfCheckGPT, the number of sampled responses, and the specific configurations adopted for its variants. The code utilized in these methodologies was obtained from GitHub, and was made available by the authors of SelfCheckGPT.[8] However, the original implementation did not account for the use of quantized LLMs in the prompt-based approach. Therefore, modifications were made to the code to incorporate this feature, enhancing the overall efficiency of the process. All the procedures and experiments were executed on a Tesla T4 GPU using Google Colab.

Model Selection and Parameters Choice for SelfCheckGPT

The model selected for response generation, along with its corresponding tokenizer, was obtained from Hugging Face using the Transformers library [66]. Specifically, the chosen version is *Llama-3.2-3B-Instruct*, a LLaMa model consisting of 3 billion parameters.[9] Moreover, the model was quantized before use, to ensure efficiency and reduce computational time.

In line with the configuration used by the authors of SelfCheckGPT, the sampled responses were generated by setting the LLM temperature to 1, with a maximum output length of 128 new tokens. Additionally, while the original SelfCheckGPT paper employs 20 sampled responses ($N = 20$), the study also demonstrates that comparable results can be achieved with a lower number of samples, specifically between 4 and 6, as increasing N beyond this range tends to yield diminishing returns. Consequently, in this methodology, the number of sampled responses was set to 5 ($N = 5$). Figure 3.7 presents the prompt used to instruct the LLM in generating the sampled responses. The model was explicitly directed to leverage its internal knowledge, and provide highly detailed answers. This approach ensures that the sampled responses contain comprehensive information, facilitating a more effective comparison with the original response.

Prompt for Generating Sampled Responses

Based on your knowledge, answer the following question giving as much detail as you can.
Question: [QUESTION]
Answer:

Figure 3.7: Prompt submitted to the LLM for generating sampled responses.

[8] `https://github.com/potsawee/selfcheckgpt`
[9] `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`

Configuration for SelfCheckGPT with NLI

For the SelfCheckGPT approach based on NLI, the same model utilized by the authors was adopted, namely *deberta-v3-large-mnli*, retrievable from Hugging Face.[10] This model, built upon the DeBERTa architecture, has been fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset. The MNLI dataset comprises over 400 000 sentence pairs annotated with textual entailment labels, making it a robust resource for NLI tasks [78].

Configuration for SelfCheckGPT with LLM Prompt

For the SelfCheckGPT approach based on LLM prompt, the evaluation of the comparison between the sampled responses and the original response was originally conducted using GPT-3 and GPT-3.5-Turbo, as stated by the authors of SelfCheckGPT [49]. In this methodology, however, the same model used for response generation, *Llama-3.2-3B-Instruct*, was employed for this task. As in the previous cases, the model was quantized prior to use to enhance computational efficiency. Figure 3.8 presents the prompt employed to compare the sentences of the original response with the sampled responses. This methodology adopts the same prompt utilized by the authors of SelfCheckGPT.

Prompt for Comparing Original Answer and Sampled Responses

Context: [CONTEXT]
Sentence: [SENTENCE]
Is the sentence supported by the context above? Answer Yes or No.
Answer:

Figure 3.8: Prompt submitted to the LLM to compare the sentences of the original response with the sampled answers.

3.4 Knowledge Retrieval

Inspired by the hallucination detection approaches based on knowledge retrieval, it was also decided to explore methods that leverage real-world knowledge. However, most of these approaches rely on access to a structured knowledge base [33] or require human annotators to compare the responses generated by LLMs against reliable sources [29], which is a time-consuming and expensive process. Given the absence of a dedicated knowledge base and human annotators, an automated knowledge retrieval system was developed using Google. Specifically, a custom Google search engine was created to facilitate the retrieval of relevant information through API queries. Figure 3.9 illustrates the pipeline for the knowledge retrieval process utilizing the Google Search API. The procedure is structured into the following steps:

1. The query is submitted to the Google search engine.
2. Using the Google Search API, the search engine interacts with the Web, and retrieves a set of URLs relevant to the query.

[10] `https://huggingface.co/potsawee/deberta-v3-large-mnli`

3. The first URL from the retrieved set is extracted, and its contents are processed by Beautiful Soup.

4. Beautiful Soup extracts the main body content from the HTML. Note that, in certain instances, the Google search engine may fail to retrieve any results from the Web. In such cases, the process continues, but no external knowledge will be retrieved for that specific query.

5. The retrieved knowledge is indexed by means of an embedding model. An embedding model is a model that can create numerical representations of pieces of text, and identify and retrieve relevant information given a user's query.

6. The numerical representations of the texts are stored in a vector database, enabling its interpretation and utilization by the LLM.

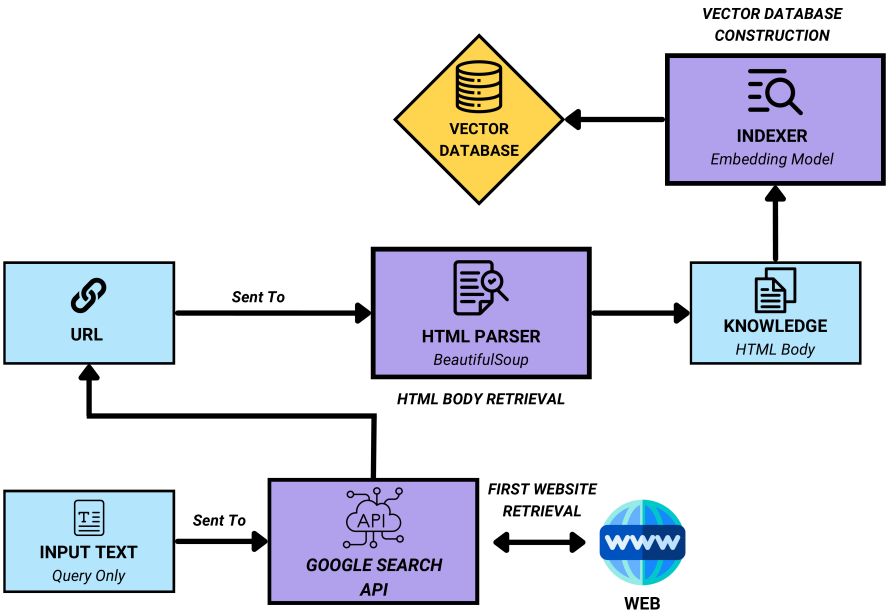

Figure 3.9: Process for knowledge retrieval through Google Search API.

The retrieved knowledge is incorporated into the SelfCheckGPT and the few-shot prompting approaches, as detailed in the following sections, to evaluate whether the introduction of external knowledge influences the performance of these hallucination detection methods.

3.4.1 Few-Shot Prompting with External Knowledge

Few-shot prompting involves presenting the model with a small number of examples to enhance its understanding of the task to be performed. These examples establish the contextual framework necessary for task completion, guiding the model toward an accurate solution. To further improve performance, the knowledge retrieved via the Google Search API was incorporated into this framework, providing the LLM with additional information to enhance the accuracy of its task execution. Figure 3.10 illustrates the pipeline of the few-shot prompting approach enhanced with knowledge, which is structured into the following steps:

1. The user query is processed by the embedding model.
2. The embedding model identifies relevant information related to the query, and retrieves specialized knowledge corresponding to the given input.
3. The retrieved specialized knowledge, along with a set of examples and the query-response pair to be evaluated, is provided to the LLM.
4. Using a structured prompt, the LLM analyzes the response under evaluation in relation to the specialized knowledge, and determines its factuality, returning a classification of either true (factual) or false (hallucinated).

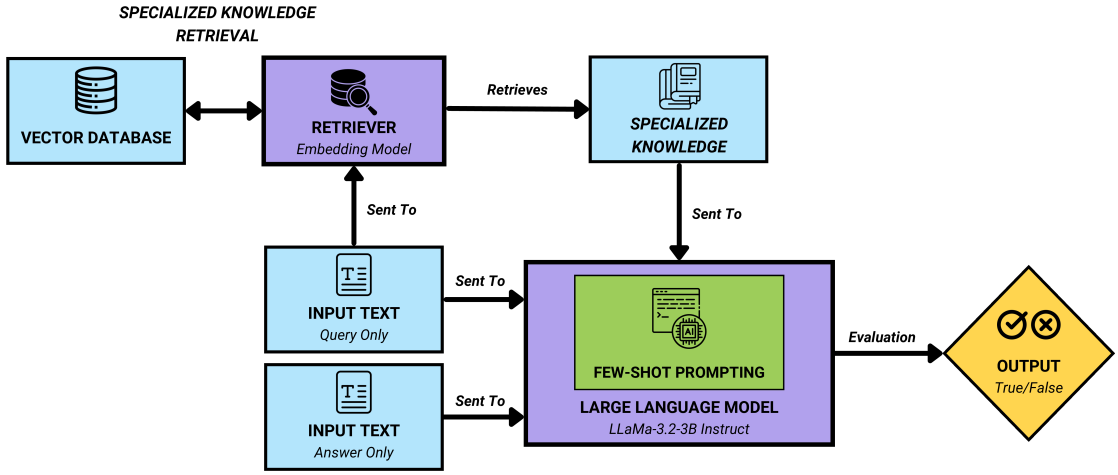

Figure 3.10: Pipeline of the hallucination detection approach based on few-shot prompting using the retrieved knowledge.

3.4.2 SelfCheckGPT with External Knowledge

The knowledge has also been integrated into the SelfCheckGPT framework to enhance the generation of sampled responses. By providing the model with relevant knowledge, the sampled responses are enriched with more accurate information. Consequently, when these responses are compared with the response under evaluation, the model can more effectively determine whether the response constitutes a hallucination or not. Figure 3.11 illustrates the pipeline of SelfCheckGPT augmented with external knowledge. The process is structured into the following steps:

1. The user query is processed by the embedding model.
2. The embedding model retrieves specialized knowledge relevant to the given query.
3. The retrieved knowledge, along with the user query, is provided as input to the LLM.
4. The model is prompted to generate N response samples based on both the query and the retrieved knowledge.

5. The original response is segmented into individual sentences which are then compared with the N sampled responses using one of the SelfCheckGPT variants.

6. Based on this comparison, SelfCheckGPT assigns a hallucination score to the evaluated response. This score ranges from 0 to 1, where 1 indicates a hallucination, and 0 signifies factual content. The hallucination score is then converted into a binary value (true/false) using a threshold function:

$$\hat{y} = \begin{cases} \text{True}, & \text{if } S_{\text{BERT}}(r) \leq \tau \\ \text{False}, & \text{if } S_{\text{BERT}}(r) > \tau \end{cases}$$

where τ is the threshold optimized to maximize classification accuracy based on the ground truth.

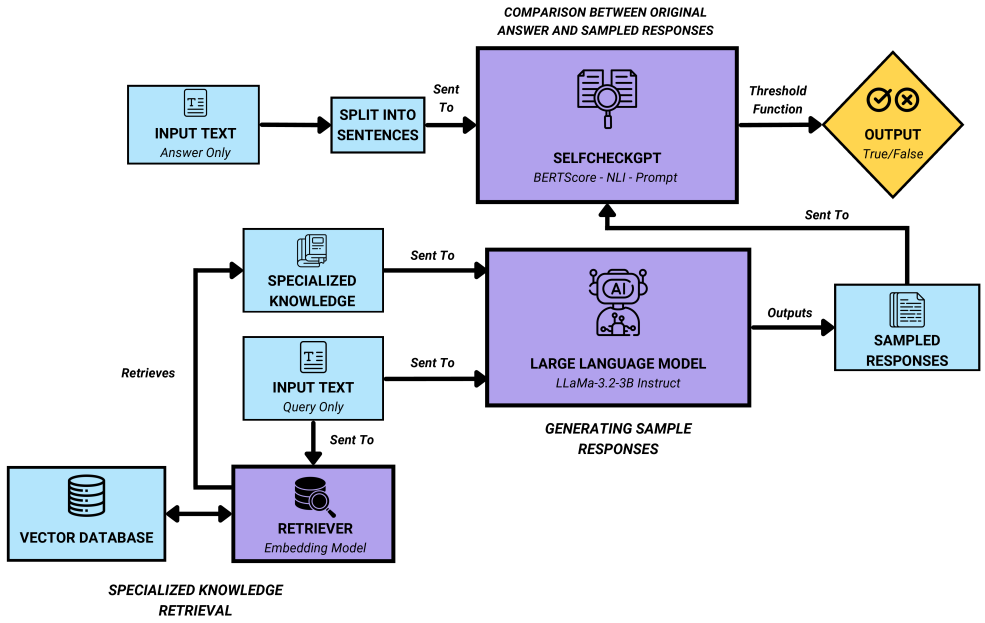

Figure 3.11: Pipeline of the hallucination detection approach based on SelfCheckGPT using the retrieved knowledge.

Independently of the incorporated knowledge, the SelfCheckGPT variants operate in the same manner as described in the Sections 3.3.1, 3.3.2, and 3.3.3. Furthermore, the model employed for few-shot prompting, the generation of sampled responses, and SelfCheckGPT based on LLM prompt is *Llama 3.2-3B-Instruct*, which has been previously described in Sections 3.3.4 and 3.3.5.

3.4.3 Retrieval-Augmented Generation

The integration of external knowledge into SelfCheckGPT and few-shot prompting is achieved through a process known as Retrieval-Augmented Generation (RAG). This technique enhances the capabilities of LLMs by retrieving relevant documents from an external knowledge base using semantic similarity computations. By incorporating retrieved information, RAG helps mitigate the issue of non-factual content generation [79]. RAG methodologies can be categorized into three main approaches, that will be detailed in the following sections [80].

Naive RAG

The Naive RAG methodology represents the most fundamental approach to RAG and was the first to be introduced. Within this study, Naive RAG is the methodology that is used to integrate external knowledge into the proposed approaches. It follows a structured process consisting of the following key steps:

1. An indexing phase, where external knowledge is processed to facilitate efficient retrieval. This involves cleaning the retrieved data and segmenting it into smaller, meaningful text chunks to optimize retrieval performance. Each chunk is then converted into a numerical representation using an embedding model and subsequently stored in a vector database.

2. A retrieval phase, where a user submits a query, which is processed by the same embedding model used during the indexing phase to generate a corresponding vector representation. The system then computes semantic similarity scores between the query vector and the stored chunk vectors, selecting and retrieving the top-k most relevant chunks based on these similarity scores.

3. A generation phase, where the query and the retrieved chunks are incorporated into a structured prompt, which is then provided to the LLM. The model generates a response by leveraging both the retrieved documents and its internal parametric knowledge.

Despite its structured approach, Naive RAG presents several limitations at both the retrieval and generation stages. The retrieval process may fail to select the most relevant documents, potentially omitting critical information and reducing the informativeness of the response. Additionally, if the retrieved data contains biased, toxic, or inappropriate content, these issues could inadvertently be reflected in the model's output, compromising the quality and reliability of the generated responses.

Advanced RAG

Advanced RAG refines the Naive RAG framework by introducing several enhancements aimed at optimizing both pre-retrieval and post-retrieval processes, ensuring that retrieved information is both highly relevant and effectively utilized.

In particular, the indexing phase is improved by incorporating metadata and segmenting data into smaller, more precise chunks, thereby increasing the accuracy of information retrieval. Additionally, the original query undergoes refinement through query expansion and rewriting techniques, making it clearer and better suited for retrieval tasks. The post-retrieval phase is also optimized to ensure that only the most essential information is passed to the LLM. This is achieved by re-ranking retrieved chunks, prioritizing the most relevant ones at the beginning of the LLM's context window, and compressing document content to eliminate redundant or repetitive information.

These enhancements significantly improve the effectiveness, relevance, and accuracy of the LLM-generated responses, while also reducing the likelihood of hallucinations.

Modular RAG

Modular RAG represents a significant advancement over Naive RAG and Advanced RAG, introducing a more flexible and adaptable approach to retrieval and knowledge integration. Unlike its predecessors, Modular RAG enables customization and optimization at various stages of the retrieval process by incorporating specialized modules and restructuring retrieval pipelines. These enhancements lead to greater accuracy, efficiency, and adaptability in information retrieval and generation.

One of the key improvements in Modular RAG is its ability to enhance search and retrieval by allowing direct queries across multiple data sources, including databases and knowledge graphs. Additionally, it introduces a long-term memory mechanism that enables the system to store and retrieve relevant information across multiple interactions, thereby creating an adaptive and evolving knowledge base. Another major innovation is query expansion, which improves retrieval by combining parallel vector searches with intelligent re-ranking techniques. This allows the system to retrieve a broader range of relevant information while prioritizing the most meaningful results. Furthermore, Modular RAG supports flexible retrieval approaches, dynamically adapting to task-specific and domain-specific requirements. For instance, it can replace traditional retrieval with LLM-generated content, enabling context-aware synthesis without relying on external knowledge bases, It can leverage internal model weights to retrieve relevant knowledge, significantly improving performance on knowledge-intensive tasks. Moreover, Modular RAG integrates with other AI technologies, such as fine-tuning retrievers for higher precision in document selection, optimizing generators for personalized and domain-specific responses, and incorporating advanced learning techniques to enhance retrieval quality.

By offering unprecedented flexibility, scalability, and efficiency, Modular RAG represents an evolution in RAG, making it a highly adaptable and powerful framework for knowledge retrieval and integration.

3.4.4 Experimental Setup

This section provides a detailed overview of the implementation choices made for the integration of the knowledge and the approaches that incorporate the retrieved information. The knowledge retrieval procedure was executed using Google Colab, as well as both few-shot prompting and SelfCheckGPT, using a Tesla T4 GPU.

Few-Shot Prompting with External Knowledge

The *jina-embeddings-v3* model [81] was selected as the embedding model for constructing the knowledge vector database, and serving as the retriever. It was obtained from Hugging Face using the Transformers library.[11] Once retrieved, the knowledge was segmented into chunks of 256 characters with an overlap of 25 characters to prevent abrupt cuts between concepts, and ensure continuity. The retriever was configured to retrieve the five most relevant documents.

Figure 3.12 shows the prompt provided to the model for classifying texts as factual or hallucinatory. The prompt was structured to include a definition of hallucination, offering the LLM a reference framework for classification. Additionally, the model was instructed

[11] `https://huggingface.co/jinaai/jina-embeddings-v3`

to utilize both its internal knowledge and the specialized knowledge retrieved by the retriever to enhance classification accuracy. The prompt also accounted for scenarios where external knowledge might be unavailable or insufficient, in which case the model was directed to rely solely on its internal knowledge.

Prompt for Few-Shot Prompting with Knowledge

I want you to act as a response judge.
Given a user query, a knowledge, and a response by an LLM, your objective is to determine if the response is an hallucination or not.
In the context of NLP, an "hallucination" refers to a phenomenon where the LLM generates text that is incorrect, nonsensical, or not real. Based on your knowledge, on the knowledge provided, and on the definition of hallucination provided, analyze the user query and the response of the LLM, and answer the following question: is the response factual or not?
BE CAREFUL: sometimes the knowledge may be empty or not useful, in which case you have to respond based only on your knowledge.
Answer True if you consider the response factual, False otherwise. You don't have to provide any explanation.
EXAMPLE 1
User query: [USER QUERY]
Knowledge: [KNOWLEDGE]
LLM response: [LLM RESPONSE]
Answer: [ANSWER]
...
EXAMPLE N
User query: [USER QUERY]
Knowledge: [KNOWLEDGE]
LLM response: [LLM RESPONSE]
Answer: [ANSWER]
LLM TURN
User query: [USER QUERY]
Knowledge: [KNOWLEDGE]
LLM response: [LLM RESPONSE]
Answer:

Figure 3.12: Prompt submitted to the LLM for few-shot prompting with knowledge.

The few-shot prompting approach was evaluated by providing the model with 1, 5, and 10 examples, assessing its performance across different levels of contextual guidance.

SelfCheckGPT with External Knowledge

For this approach, the selected embedding and retriever model is *bge-base-en-v1.5*,[12] obtained from Hugging Face via the Transformers library [82]. To construct the vector database, the retrieved knowledge was segmented into chunks of 128 characters, with an overlap of 25 characters to preserve contextual continuity. The retriever was configured

[12] `https://huggingface.co/BAAI/bge-base-en-v1.5`

to retrieve the seven most relevant documents. Figure 3.13 illustrates the prompt used for generating response samples utilizing the knowledge integrated through RAG. In this prompt, the LLM is explicitly instructed to incorporate specialized knowledge into the response generation process, ensuring higher accuracy in the sampled responses, which are then compared with the response under evaluation.

Prompt for Generating Sampled Responses with Knowledge

Based on your knowledge and on the context provided, answer the following question giving as much detail as you can.
Question: [QUESTION]
Context: [KNOWLEDGE]
Answer:

Figure 3.13: Prompt submitted to the LLM for generating the sampled response using the retrieved knowledge.

Regarding the number of generated responses, the temperature setting, and the prompt used for response comparison, the configuration remains consistent with the specifications outlined in Section 3.3.5.

Chapter 4

Results

After describing the methodologies employed for the various approaches, this chapter will examine the datasets used for evaluation, detailing their creation and preparation for the experiments, and will present the results of the conducted experiments.

4.1 Datasets

One of the primary challenges in hallucination detection lies in the limited availability of benchmark datasets specifically designed for this task. Constructing such datasets typically requires human annotators to evaluate the factuality of given passages, and classify them as either true or false. However, this process is both time-consuming and costly. As a result, an alternative approach involves leveraging LLMs to automatically annotate passages using a knowledge retrieval mechanism. By providing relevant external knowledge to the LLM, it becomes possible to assess the factuality of a passage with greater accuracy. However, relying on LLMs for dataset creation presents an inherent risk: since LLMs themselves can generate hallucinations, there is a possibility of producing inaccurate datasets, creating a self-reinforcing issue. To mitigate this risk, a common strategy involves a hybrid approach: LLMs are used to annotate the factuality of passages, while human annotators verify the accuracy of a random sample of these annotations. This validation step helps ensure the reliability of the dataset, and maintains the performance of the LLM-based annotation system.

Another critical challenge in creating benchmark datasets for hallucination detection is that it is not sufficient to merely have datasets containing passages labeled as true or false; rather, these passages must be generated by an LLM. Additionally, some studies intentionally prompt the LLM to produce hallucinations for dataset creation [83]. However, this approach risks compromising the dataset's integrity, and consequently, the reliability of hallucination detection methods. To ensure the dataset remains accurate and representative, hallucinations must emerge naturally during the LLM's generation process, without any deliberate influence. This preserves both the authenticity of the dataset, and the validity of the evaluations conducted using it.

For the evaluation and comparison of the approaches outlined in the previous chapter, three datasets were selected. Each dataset includes a user query, the corresponding response generated by an LLM, and a label indicating the factuality of the response. The datasets considered are:

- **FactAlign:** A dataset utilized for training the fKTO alignment algorithm, designed to enhance the factual accuracy of LLM-generated responses.
- **FactBench:** A dataset developed to assess the effectiveness of automatic fact-checking systems. It comprises three distinct datasets, each annotated by human experts.
- **FELM:** A benchmark dataset for evaluating factual accuracy in LLMs, annotated by human experts, and encompassing multiple knowledge domains.

The following sections will provide a detailed description of the creation process for each dataset, and the preparation steps undertaken for the experiments conducted in this study.

4.1.1 FactAlign

This dataset takes its name from an alignment framework designed to enhance the factual accuracy of LLMs in generating long-form responses [84]. Specifically, it is utilized to train models using the fKTO algorithm, an extension of the Kahneman-Tversky Optimization (KTO) alignment method [85]. KTO is an efficient and scalable approach for aligning LLMs with human preferences, enabling them to generate more factually accurate responses while minimizing hallucinations, and improving the overall quality of the information produced.

Dataset Construction Process

For the evaluation of long-form text generations, [84] employs a multi-step approach, similar to FActScore [29]. First, the generated text is segmented into sentences, and for each sentence, an LLM generates sentence-related statements. These statements are then compared against retrieved knowledge to assess their factual accuracy. The evaluation process relies on three key metrics:

- **Factual precision:** Measures the proportion of statements that are supported by retrieved knowledge. This metric corresponds to FActScore as detailed in Section 2.3.1 [29]. However, a model could achieve a high precision score by generating only a limited number of highly confident statements.
- **Factual recall:** Accounts for the amount of information expected by the user, penalizing models that generate insufficient content.
- **Factual F1-score:** Combines factual precision and factual recall, making it a more reliable metric than factual precision alone.

The dataset used for evaluation was constructed following these steps:

1. 38 topics were identified, and for each topic, GPT-4-Turbo was prompted to generate 30 queries requesting information about specific aspects of that topic.
2. Each query was fed into an LLM (in our case, Gemma-2B) to generate a response.
3. Using GPT-3.5-Turbo, the responses were first divided into sentences, and then further decomposed into atomic statements following the FActScore methodology [29].

4. For each statement, GPT-3.5-Turbo generated a query to retrieve relevant knowledge.

5. The generated queries were processed by a retriever that sourced relevant information from a Wikipedia corpus.

6. The retrieved knowledge was presented to GPT-3.5-Turbo, which determined whether each statement was supported by the retrieved evidence.

7. A sentence was considered factually accurate if all of its associated statements were supported by the retrieved knowledge.

8. The entire response was deemed factual if the Factual F1-score across all sentences exceeded 75%.

The dataset was obtained from Hugging Face in the DatasetDict format, and it comprised 2 562 observations, each consisting of a prompt, the corresponding generated response, and a factuality label.[1] Among these observations, 1 307 were labeled as true, while 1 255 were labeled as false. Below is an example of an observation from the FactAlign dataset (the generation is cut short due to space limit).


```
{
    "prompt": [{"content": "What is the Phaedrus Dialogue? Provide as many
    specific details and examples as possible (such as names of people,
    numbers, events, locations, dates, times, etc.)",
    "role": "user"}],
    "completion": [{"content": "The Phaedrus Dialogue is a philosophical
    text written by the ancient Greek philosopher Phaedrus... that
    is both accessible and profound.",
    "role": "assistant"}],
    "label": "False"
}
```


Each observation in the dataset is structured as a nested JSON. Therefore, it was necessary to extract both the question and the response generated by the model from the JSON format. Furthermore, the suffix encouraging the LLM to generate the maximum amount of information was removed (*"Provide as many..."*). Below is the observation after extracting the prompt and the generated answer, and removing the suffix.


```
{
    "prompt": "What is the Phaedrus Dialogue?",
    "completion": "The Phaedrus Dialogue is a philosophical text written by
    the ancient Greek philosopher Phaedrus ... that is both accessible and
    profound.",
    "label": "False"
}
```


[1] https://huggingface.co/datasets/chaoweihuang/factalign-gemma2-f1_0.75

Data Preparation for Classification Approaches

FactAlign exhibits a fairly balanced distribution among the classes, consequently no rebalancing procedures are required.

For textual classification, the dataset was partitioned into a training set, validation set, and test set, comprising 1 639, 410, and 513 examples, respectively. To maintain consistency across experiments, a fixed seed was used to ensure that the same elements were included in each set, both in the scenario where only the response was provided to the model, and in the scenario where both the prompt and response were included.

For the classification of internal states, the dataset was divided only into training and test set, comprising 2 049 and 513 examples, respectively, without the inclusion of a validation set (it is included in the training set). To maintain consistency and ensure comparability across models, a fixed seed was used, guaranteeing that the composition of the sets remained identical across different experiments.

Data Preparation for Prompting Approaches

For the prompt-based approaches, that are few-shot prompting with knowledge and SelfCheckGPT (both with and without knowledge), only questions were considered for simplicity. Specifically, only prompts ending with a question mark were selected, facilitating knowledge retrieval via the Google Search API, and simplifying the LLM's task in classifying the factuality of responses and generating sampled responses within the SelfCheckGPT approach. After filtering, a random selection of 100 questions was made. This limitation was necessary due to time and computational constraints, requiring a balance between the number of examples and processing efficiency. Additionally, to ensure comparability between SelfCheckGPT and few-shot prompting with knowledge, as well as consistency across different methods within the same approach, a fixed random seed was used to guarantee that the selected 100 observations remained unchanged across experiments.

4.1.2 FactBench

This dataset was specifically designed to evaluate FactCheck-GPT, a multi-step framework developed for the detection and correction of factual errors in responses generated by LLMs [86].

Dataset Construction Process

FactBench was constructed by integrating three different benchmark datasets designed for hallucination detection in language models:

- **Knowledge-based FacTool:** A dataset developed for evaluating FacTool, a framework designed to detect factual errors in LLM-generated responses by leveraging external knowledge retrieval through the Google Search API or Google Scholar [87]. The dataset was built by collecting 50 prompts from FactPrompts, a dataset containing real-world queries sourced from platforms such as Quora, and fact-checking datasets like TruthfulQA [88]. Responses to these prompts were generated using ChatGPT, and manually annotated as true or false by human annotators.
- **FELM-WK:** A subset of the FELM dataset related to the domain of world knowledge, the construction of which will be detailed in Section 4.1.3.

- **HaluEval:** A benchmark dataset for detecting and evaluating hallucinations in LLMs [83]. The dataset construction involved selecting 52 000 prompts, and implementing a filtering process to identify those most prone to hallucinations. This process included:

  1. Generating three responses per prompt using ChatGPT.
  2. Computing the average semantic similarity between the three responses.
  3. Selecting the 5 000 queries with the lowest similarity scores, ensuring that only the most challenging queries were retained in the dataset.
  4. Submitting the selected 5 000 prompts to ChatGPT for a second round of response generation.
  5. Manually annotating the responses as true or false, based on their factual accuracy.

The dataset was made available by the authors on GitHub, and it comprised 4 835 examples, each consisting of a prompt, a response generated by an LLM, and a manually assigned factuality label.[2] Among these, 3 838 examples were labeled as true, 995 examples as false, and 2 examples did not have a factuality label assigned, which have been excluded from the dataset. Furthermore, one entry in the dataset lacked the response generated by the LLM. Consequently, this entry was excluded from the dataset to maintain consistency and completeness, thus lowering the number of observations in the dataset to 4 832. Below is an example of an observation from the FactBench dataset.


```
{
    "prompt": "Describe the composition of Jupiter.",
    "response": "Jupiter is a gas giant planet composed mainly of hydrogen
    and helium. It has a small rocky core with a mass of around 10/30 Earth
    masses, surrounded by a thick layer of metallic hydrogen, which
    gradually transitions to molecular hydrogen with increasing altitude.
    Jupiter's atmosphere is composed of roughly 75% hydrogen and 24% helium,
    with small amounts of methane, water vapor, ammonia, and other
    molecules. The planet's complex weather systems include strong winds and
    massive storms, such as the famous Great Red Spot.",
    "response_label": "False"
}
```


Data Preparation for Classification Approaches

FactBench exhibits a significant class imbalance, necessitating a rebalancing strategy to ensure the reliability of the evaluations. To address this, a downsampling procedure was applied, whereby all observations labeled as false were retained, and an equivalent number of observations were randomly selected from those labeled as true. To maintain consistency across all classification approaches, a fixed random seed was used. As a result, the final dataset comprised 1 990 observations, evenly distributed between true and false labels.

For text classification, the dataset was divided into a training set, validation set, and test set, comprising 1 273, 319, and 398 observations, respectively. A fixed random seed was applied to ensure consistency across analysis, guaranteeing that the partitioning of

[2] https://github.com/yuxiaw/Factcheck-GPT

the dataset remained identical across different evaluations.

With regard to the classification of internal states, the dataset was split into training and test set only, consisting of 1 592 and 398 examples, respectively. Again, a fixed random seed was used to maintain comparability across different approaches.

Data Preparation for Prompting Approaches

For the prompting-based approaches, only prompts formulated as questions were considered, to streamline both knowledge retrieval and the processing tasks performed by the LLM. This was achieved by filtering prompts based on whether they concluded with a question mark. Additionally, due to computational and time constraints, a total of 100 examples were selected. However, to account for the dataset's imbalance, an equal number of true and false examples were randomly chosen, resulting in 50 observations labeled as true and 50 labeled as false. To ensure consistency and comparability across different approaches, a fixed random seed was applied.

4.1.3 FELM

FELM is a benchmark dataset designed for detecting hallucinations in LLMs [89]. Its distinctive feature lies in its inclusion of five domains, each representing a unique challenge for LLMs. Specifically, these domains are:

- **World knowledge:** One of the most widely used domains in fact-checking, this category encompasses general knowledge about specific entities such as films, countries, and places.
- **Science and technology:** Since scientific and technological information is less prevalent on the Web, LLMs are more likely to generate hallucinated content in this domain. FELM includes factual errors related to scientific claims and citations across various academic disciplines, such as physics and biology.
- **Recommendation and writing:** This domain focuses on common practical applications of LLMs, such as generating recommendations and producing written content. Due to the open-ended nature of many queries in this category, LLM-generated responses tend to be less constrained, making them more prone to hallucinations.
- **Reasoning:** Logical reasoning is a crucial capability for LLMs, enabling them to tackle complex problems. The Chain-of-Thought prompting technique [90], where the model follows a structured sequence of reasoning steps before arriving at a final answer, has become a standard approach. However, despite its effectiveness, this method remains highly susceptible to errors.
- **Math:** Mathematical problem-solving represents one of the most significant challenges for LLMs, as it requires both strong logical reasoning skills and the ability to rigorously apply mathematical principles to derive correct solutions.

By encompassing both general knowledge and specialized domains, FELM serves as a comprehensive and detailed benchmark for evaluating and identifying hallucinations in LLM-generated content.

Dataset Construction Process

The construction of FELM follows a structured methodology designed to ensure accuracy, reliability, and diversity. The process consists of the following steps:

1. Prompts are sourced from various platforms, including online forums (*e.g.*, Twitter, Quora) and benchmark datasets. The selection of sources is domain-specific to guarantee comprehensive coverage of different knowledge areas.

2. ChatGPT is employed to generate responses to the collected prompts in a zero-shot setting.

3. To facilitate analysis, responses are segmented based on their structure: structured responses (*e.g.*, text-based answers) are segmented using sentence boundaries while unstructured responses (*e.g.*, those related to mathematics or recommendations, which may include code, lists, or numbers) are segmented using ChatGPT to handle complex formatting.

4. A team of expert annotators evaluates the factual accuracy of each response segment, labeling them as true or false based on verifiable evidence.

5. To ensure the reliability of annotations, a rigorous verification process is conducted. This includes an independent review by an external evaluator, and random sampling of 100 examples per domain for additional validation.

The dataset was obtained from Hugging Face.[3] In this study, a response is labeled as true if all its segments are annotated as true, whereas it is deemed false if at least one segment is labeled as false. The dataset comprises 847 examples, each consisting of a prompt that generated the response, the response produced by the LLM, and a factuality label indicating whether the response is true or false. Among these examples, 281 are labeled as false, while 566 are labeled as true. An observation from the FELM dataset is provided below.


```
{
    "prompt": "Who is the President of the European Commission?",
    "response": "As of 2021, the President of the European Commission is
    Ursula von der Leyen.",
    "hallucination": "True"
}
```


Data Preparation for Classification Approaches

FELM exhibits an imbalance in its class distribution, necessitating rebalancing to ensure the reliability of assessments. To address this, a downsampling strategy was employed: all examples labeled as false were retained, while an equal number of true examples were randomly selected. A fixed random seed was applied to ensure consistency in the sampled observations. As a result, the balanced dataset comprises 562 examples, evenly distributed between true and false.

For text classification, the dataset was partitioned into training, validation, and test set, consisting of 359, 90, and 113 examples, respectively. A fixed random seed was used

[3] `https://huggingface.co/datasets/hkust-nlp/felm`

to maintain consistency across partitions, ensuring comparability between different classification approaches.

For the classification of internal states, the dataset was divided into training set (449 examples) and test set (113 examples), again using a fixed random seed to preserve comparability.

Data Preparation for Prompting Approaches

For the prompting approaches, only the science and world knowledge domains were considered, as the other domains contained prompts that posed significant challenges for our approach of knowledge retrieval (*e.g.*, mathematical prompts such as *"What is the value of the expression* $1! + 2! + 3! + \cdots + 10!$*?"*). Furthermore, only question-based prompts were included, identified by considering only the prompts that ended with a question mark. To address dataset imbalance, and for time and computational constraints, 100 examples were randomly selected, ensuring an equal distribution of 50 true-labeled and 50 false-labeled examples. A fixed random seed was applied to ensure consistency and comparability across different approaches.

4.1.4 Summary

Tables 4.1 and 4.2 provide an overview of the structure and partitioning of the datasets utilized in the experiments. In Table 4.1, the *Rebalanced* attribute specifies whether the dataset has undergone rebalancing procedures prior to its use in the experiments. In Table 4.2, SelfCheckGPT and Few-Shot Prompting encompass both the traditional versions and the versions incorporating integrated knowledge.

Dataset	# All	# True	# False	Rebalanced
FactAlign	2 562	1 307	1 255	✗
FactBench	1 990	995	995	✓
FELM	5 62	281	281	✓

Table 4.1: Description of the dataset structure used in the experiments.

Dataset	Text Classification			Hidden States Classification			SelfCheckGPT	Few-Shot Prompting
	# Train	# Val	# Test	# Train	# Val	# Test	#	#
FactAlign	1 639	410	513	2 049	-	513	100	100
FactBench	1 273	319	398	1 592	-	398	100	100
FELM	359	90	113	449	-	113	100	100

Table 4.2: Summary of dataset partitioning across different approaches.

Given that all the datasets used for evaluating the approaches are balanced, the metric considered for assessment is accuracy, which is defined as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP denotes factual responses correctly classified as factual, TN represents hallucinated responses correctly identified as hallucinations, FP corresponds to hallucinated

responses incorrectly classified as factual, and FN refers to factual responses mistakenly classified as hallucinations.

4.2 Baseline

Drawing inspiration from studies such as [22] and [47], few-shot prompting (without the use of external knowledge) has been considered as the comparation baseline to evaluate the approaches. This method involves presenting the model with a variable number of examples to establish the contextual framework, and guide it in performing the task effectively. Figure 4.1 illustrates the few-shot prompting approach adopted as the baseline, which is structured as follows:

1. The query, response, and relevant examples are provided as input to the LLM.
2. The model is instructed to assess the factual accuracy of the response based on its internal knowledge.
3. The LLM classifies the response as either true or false, depending on its factual accuracy.

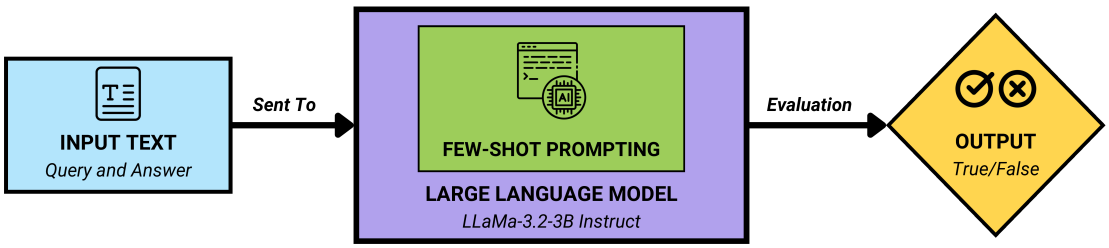

Figure 4.1: Pipeline of the hallucination detection approach based on few-shot prompting used as a baseline.

The model utilized for the few-shot prompting approach is *Llama-3.2-3B-Instruct*, which has been previously detailed in Sections 3.3.4 and 3.3.5.

4.2.1 Experimental Setup

Figure 4.2 illustrates the prompt used to instruct the LLM to classify responses as either factual or hallucinatory. The prompt includes a definition of hallucination to provide contextualization, and guide the model in understanding the task. Moreover, the prompt encourages the LLM to leverage its internal knowledge to solve the task.

Prompt for Few-Shot Prompting

I want you to act as a response judge. Given a user query and a response by an LLM, your objective is to determine if the response is an hallucination or not.
In the context of NLP, an "hallucination" refers to a phenomenon where the LLM generates text that is incorrect, nonsensical, or not real.
Based on your knowledge and on the definition of hallucination provided, analyze the user query and the response of the LLM, and answer the following question: is the response factual or not?
Answer True if you consider the response factual, False otherwise. You don't have to provide any explanation.
EXAMPLE 1
User query: [USER QUERY]
LLM response: [LLM RESPONSE]
Answer: [ANSWER]
...
EXAMPLE N
User query: [USER QUERY]
LLM response: [LLM RESPONSE]
Answer: [ANSWER]
LLM TURN
User query: [USER QUERY]
LLM response: [LLM RESPONSE]
Answer:

Figure 4.2: Prompt submitted to the LLM for few-shot prompting.

Similar to the few-shot prompting approach with integrated knowledge, this method was evaluated by presenting the model with 1, 5, and 10 examples.

4.3 Results

This section presents and analyzes the main findings from the experiments conducted. The discussion will examine the result obtained by each approach and will address the impact of knowledge within knowledge retrieval methods, the effect of providing both the query and the response instead of only the response, and the optimal layer for extracting internal states.

4.3.1 Supervised Text Classification

Table 4.3 presents the performance of the models employed in the text classification approach, compared against the selected baselines. The notations A and $Q{+}A$ indicate whether the model was provided with only the answer, or both the query and the answer, respectively. For each dataset, the combination between model and scenario which performs best is highlighted in bold, while the baseline which performs best for each dataset is highlighted in red.

	FactAlign		FactBench		FELM	
Baselines	**A**	**Q+A**	**A**	**Q+A**	**A**	**Q+A**
One-shot	-	50.0	-	62.0	-	**62.0**
Five-shot	-	57.0	-	53.0	-	56.0
Ten-shot	-	55.0	-	59.0	-	59.0
Models						
BERT	74.1	73.7	62.8	60.6	60.2	58.4
RoBERTa	75.0	**76.6**	63.8	**65.1**	53.1	61.1
GPT-2	75.4	75.6	63.8	60.0	61.9	**62.0**

Table 4.3: Evaluation of models in hallucination detection through text classification, compared with the baselines.

Performance Analysis on the Datasets

The dataset where the models achieve the highest performance is FactAlign, with an overall accuracy of approximately 75%. The best model turns out to be RoBERTa in the query+answer scenario, reaching 76% accuracy, followed closely by GPT-2, which achieves 75.6% and 75.4% in the query + answer and answer-only scenarios, respectively. A plausible explanation for the strong performance on FactAlign is the length of the answers contained within the dataset. Indeed, FactAlign is a dataset that contains very long answers, and the LLM that originally generated these responses may have introduced inconsistencies or contradictions, which are effectively detected by classification models. Additionally, the textual structure of the responses may exhibit specific linguistic patterns that the models can leverage for classification. It also can be noticed that supervised classification models significantly outperform the baseline approaches on FactAlign.

The models exhibit lower performance on FactBench and FELM compared to FactAlign. Specifically, on FactBench, the best combination remains RoBERTa in the query + answer scenario, achieving 65.1% accuracy. However, it is only three percentage points higher than the baseline.

On FELM, performance declines even further, with the best combination being GPT-2 in the query + answer scenario, reaching 62% accuracy, which merely matches the baseline performance. A possible explanation for this performance drop lies in the shorter responses in FactBench and FELM, which may provide less contextual and textual information for the models to identify patterns, and classify hallucinations accurately. Additionally, in the case of FELM, the limited size of the training set may hinder the model's ability to learn effectively, thereby impacting even more the overall performance.

Performance Analysis in Different Scenarios

No model appears to achieve significant performance improvements when provided with both the query and the response, as opposed to only the response. Notably, a substantial difference is observed in just one case: RoBERTa, when tested on FELM, shows an 8 percent increase, rising from 53.1% with only the response to 63.1% when given both the response and the query. For the other combinations, the observed gains do not seem to impact performance positively in a meaningful way; on the contrary, in some cases,

performance declines. This is evident in the case of GPT-2 on FactBench, where accuracy drops from 63.8% in the answer-only scenario to 60% in the query + answer scenario.

4.3.2 Hidden States Classification

Table 4.4 presents the results obtained from the extraction and classification of the internal states of the LLM. The notation *A* and *Q+A* specifies whether, during the extraction of internal states, the model was provided solely with the answer to be evaluated or with both the answer and the corresponding query, respectively. Furthermore, *Layer* denotes the specific layer from which the internal states used by the classifier were extracted and *All* indicates that internal states from all layers were provided to the model. *Average* represents the mean performance across all datasets for the specified model and layer. For each dataset, the combination between model, layer, and scenario which performs best is highlighted in bold. Underlined is represented the best average obtained for a specific model (excluding *All*). The baseline which performs best for each dataset is highlighted in red.

		FactAlign		**FactBench**		**FELM**		**Average**
Baselines	**Layer**	**A**	**Q+A**	**A**	**Q+A**	**A**	**Q+A**	
One-shot		-	50.0	-	62.0	-	62.0	
Five-shot		-	57.0	-	53.0	-	56.0	
Ten-shot		-	55.0	-	59.0	-	59.0	
Models								
SVM	12-th	66.9	71.9	61.8	64.6	54.9	58.4	61.9
	20-th	71.0	71.7	60.3	64.1	57.5	59.3	64.0
	24-th	70.8	**74.9**	61.8	64.3	61.1	59.3	65.4
	28-th	73.1	73.1	62.6	64.1	63.7	60.2	<u>66.1</u>
	All	71.5	73.7	62.6	**65.6**	63.7	60.2	66.2
Logistic Regression	12-th	68.8	72.7	56.5	58.5	55.8	55.8	61.4
	20-th	70.0	70.2	57.8	62.8	59.3	56.6	62.8
	24-th	68.0	70.4	57.5	61.6	55.8	63.7	62.8
	28-th	67.5	69.4	56.5	57.3	66.4	61.9	<u>63.2</u>
	All	69.0	72.5	57.0	57.8	**67.3**	60.2	64.0
DNN (3 layer)	12-th	56.5	68.2	59.4	58.3	60.2	62.8	<u>60.9</u>
	20-th	46.6	46.6	53.3	46.7	61.1	58.4	52.1
	24-th	46.6	46.6	46.7	46.7	46.9	46.9	46.7
	28-th	46.6	53.4	53.3	46.7	53.1	46.9	50.0
	All	53.4	46.6	53.3	53.3	53.1	53.1	52.1
DNN (5 layer)	12-th	66.1	66.3	57.8	57.0	58.4	60.2	60.1
	20-th	64.9	68.2	59.8	56.0	58.4	61.1	61.4
	24-th	72.5	66.7	60.1	62.1	54.0	60.2	<u>62.6</u>
	28-th	63.7	63.5	57.3	59.8	61.1	51.3	59.5
	All	71.2	70.4	53.3	58.3	55.8	61.9	61.8

Table 4.4: Evaluation of models in hallucination detection through hidden states classification, compared with the baselines.

Performance Analysis on the Datasets

Similar to the case of text classification, performance remains high on the FactAlign dataset but decreases on FactBench and FELM. On FactAlign, the model that performs best is SVM utilizing internal states extracted from the 24th layer of the LLM, achieving an accuracy of 74.9%. For FactBench, the highest accuracy is 65.6%, again obtained by SVM, but in this case, the model is trained on all the internal states extracted. Finally, on FELM, the best model is logistic regression trained on all the internal states extracted from the LLM, achieving an accuracy of 67.3%.

The decline in performance may once again be attributed to the nature of the FactAlign dataset, which contains particularly long responses. This increases the likelihood that the LLM, generating the responses, introduced contradictory information. The LLM from which internal states were extracted may have detected these inconsistencies, making its internal representations more reflective of the presence of this information. However, it is important to highlight that the best models surpassed the dataset baselines by a margin of at least three percentage points, although not all the models tested managed to outperform them.

Performance Analysis of the Classifiers

The only classifier evaluated by the authors of SAPLMA is a three-layer DNN [22]. However, the experimental results indicate that the most effective classifiers for classifying internal states are SVM and logistic regression, both of which are traditional machine learning models. This performance advantage is particularly evident for SVM, which consistently outperforms neural network-based models across all datasets, regardless of whether they have three or five layers.

Performance Analysis in Different Scenarios

In certain cases, a slight improvement in model performance can be observed when the internal states are extracted while the LLM is processing both the query and the response. These enhancements are particularly evident in traditional machine learning models, on the FactAlign and FactBench datasets. However, for models based on neural networks, these performance gains become less pronounced.

Performance Analysis for Different Layers

The authors of SAPLMA, in their study, identified that the most informative layers for extracting internal states, those providing the greatest insight into the factuality of responses, were typically located near the middle of the LLM (specifically, the 24th and 20th layers in their case) [22]. In this study, however, the optimal layers for classification appear to depend on the classifier used. For traditional machine learning models, contrary to the findings in SAPLMA, the most effective layers tend to be those closest to the output layer, as they are more involved in generating the next token. Conversely, for the three-layer DNN, the results align with SAPLMA, as the most effective layer is the 12th, which is the furthest from the output layer among those considered, and primarily focuses on extracting low-level features from the input. For the five-layer DNN, however, the best-performing layer for internal state extraction is the 24th, which is also close to the output layer. Nonetheless, the performance achieved using earlier layers remains com-

parable.

Furthermore, providing the classifier with all extracted layers appears to be an effective approach to enhancing classification performance in traditional classifiers. Indeed, incorporating all layers in SVM and logistic regression, leads, to a slight increase in average performance across the datasets, though the improvement is not substantial. However, for the three-layer DNN, performance remains suboptimal, whereas for the five-layer DNN, it returns to an average level.

4.3.3 SelfCheckGPT

Table 4.5 presents the results obtained using the zero-resources and black-box approach, that is SelfCheckGPT. For each dataset, the variant of SelfCheckGPT that perform best is highlighted in bold, while the baseline which performs best in each dataset is highlighted in red.

Methods	**Variant**	**FactAlign**	**FactBench**	**FELM**
Baselines	*One-shot*	50.0	62.0	62.0
	Five-shot	57.0	53.0	56.0
	Ten-shot	55.0	59.0	59.0
SelfCheckGPT	*BERTScore*	59.0	61.0	56.0
	NLI	**67.0**	**64.0**	67.0
	LLM Prompt	62.0	57.0	**69.0**

Table 4.5: Evaluation of zero-resources and black-box models in hallucination detection using SelfCheckGPT, compared with the baselines.

Performance Analysis on the Datasets

Unlike supervised classification approaches, where models achieved the highest accuracy on FactAlign, in this case, performance is more uniform across the datasets. Moreover, most of the variants outperform the established baselines.

Specifically, the variant of SelfCheckGPT that performs best on FactAlign is NLI, achieving 67% accuracy. Slightly lower performance is observed for SelfCheckGPT with LLM prompt and SelfCheckGPT with BERTScore, achieving 62% and 59% accuracy, respectively.

For FactBench, the variant that performs best remains NLI, achieving 64% accuracy, followed closely by BERTScore with 61% accuracy. In this case, the variant that performs worst is SelfCheckGPT with LLM prompt, which reaches 59% accuracy.

Finally, on FELM, the highest accuracy is achieved by SelfCheckGPT with LLM prompt (69%), followed by NLI (67%), while BERTScore performs the worst, with 56% accuracy.

Performance Analysis of Variants

The authors of SelfCheckGPT identified SelfCheckGPT with LLM prompt as the variant that performs best. However, they noted that this approach is computationally inefficient. As a result, they consider NLI to offer the best trade-off between effectiveness and efficiency, as it achieves performance comparable to LLM prompts but in a significantly more efficient way [49].

The findings of this study partially support these claims. Both NLI and LLM prompts emerge as the top-performing variants; however, NLI demonstrates greater effectiveness in two out of three datasets (FactAlign and FactBench), making it the best option overall.

Additionally, in line with the conclusions drawn by the authors of SelfCheckGPT, BERTScore consistently proves to be the least effective variant.

4.3.4 Knowledge Retrieval Approaches

Table 4.6 presents the results obtained from experiments conducted on methods based on knowledge retrieval. For each dataset, the variant that performs best is highlighted in bold, while the baseline which performs best for each dataset is highlighted in red.

Methods	Variant	FactAlign	FactBench	FELM
Baselines	*One-shot*	50.0	62.0	62.0
	Five-shot	57.0	53.0	56.0
	Ten-shot	55.0	59.0	59.0
SelfCheckGPT with Knowledge	*BERTScore*	64.0	59.0	58.0
	NLI	**66.0**	65.0	**71.0**
	LLM Prompt	62.0	**68.0**	69.0
Few-Shot Prompting with Knowledge	*One-shot*	54.0	62.0	63.0
	Five-shot	55.0	64.0	59.0
	Ten-shot	59.0	65.0	62.0

Table 4.6: Evaluation of models based on knowledge retrieval in hallucination detection, compared with the baselines.

Performance Analysis on the Datasets

On FactAlign, the variant that performs best is SelfCheckGPT with NLI, achieving 66% accuracy, followed by BERTScore and LLM prompt, with 64% and 62% accuracy, respectively. In contrast, performance declines with few-shot prompting, where the most effective variant is the ten-shot approach, achieving 59% accuracy.

On the FactBench dataset, the performance of SelfCheckGPT and few-shot prompting is more comparable. The best variant is SelfCheckGPT with an LLM prompt, which achieves 68% accuracy, followed by NLI with 65% accuracy. Notably, the ten-shot few-shot prompting variant matches the 65% accuracy of NLI, and all few-shot prompting variants outperform SelfCheckGPT with BERTScore.

Regarding FELM, the best variant remains SelfCheckGPT with NLI, achieving 71% accuracy, followed by LLM prompt at 69% accuracy. Although performance slightly declines for few-shot prompting on this dataset, it still surpasses SelfCheckGPT with BERTScore, which achieves only 58% accuracy.

It is worth noting that on FactAlign, all SelfCheckGPT variants outperform the baseline, while among few-shot prompting approaches, only the ten-shot variant does so. On FactBench, all variants surpass the baseline except SelfCheckGPT with BERTScore. On FELM, the baseline is outperformed by all variants except for SelfCheckGPT with BERTScore and few-shot prompting with five examples.

Performance Analysis of Variants

Overall, the most effective approach based on knowledge retrieval is SelfCheckGPT, which consistently outperforms few-shot prompting across all datasets. This is because SelfCheckGPT is specifically designed for hallucination detection, whereas few-shot prompting is a more general method.

Among the SelfCheckGPT variants, NLI remains the most effective and efficient, outperforming LLM prompting on two out of three datasets (FactAlign and FELM).

Regarding few-shot prompting, the best-performing variant is the ten-shot approach, followed by the five-shot variant, while the one-shot approach exhibits the lowest overall performance. This outcome is expected, as providing more examples helps the model better understand the task, enhances its generalization capabilities, and improves its effectiveness. However, it is worth noting that on FELM, the one-shot variant surpasses the others in performance.

Impact of Knowledge

Table 4.7 presents a comparison between SelfCheckGPT and few-shot prompting, with and without integrated knowledge. The notation *W/O* and *W* indicates whether the considered variant operates without or with the integration of retrieved knowledge, respectively. For each variant and dataset, the model version, either with or without knowledge, that achieves the highest performance is underlined (variants that exhibit identical performance in both conditions are not underlined).

Models	**Variant**	**FactAlign**		**FactBench**		**FELM**	
		W/O	**W**	**W/O**	**W**	**W/O**	**W**
SelfCheckGPT	*BERTScore*	59.0	<u>64.0</u>	<u>61.0</u>	59.0	56.0	<u>58.0</u>
	NLI	<u>67.0</u>	66.0	64.0	<u>65.0</u>	67.0	<u>71.0</u>
	LLM Prompt	62.0	62.0	57.0	<u>68.0</u>	69.0	69.0
Few-Shot Prompting	*One-shot*	50.0	<u>54.0</u>	62.0	62.0	62.0	<u>63.0</u>
	Five-shot	<u>57.0</u>	55.0	53.0	<u>64.0</u>	56.0	<u>59.0</u>
	Ten-shot	55.0	<u>59.0</u>	59.0	<u>65.0</u>	59.0	<u>62.0</u>

Table 4.7: Comparison between methods with and without integrated knowledge, to evaluate the impact of knowledge on their performance.

Overall, the introduction of integrated knowledge has a generally positive impact on the performance of the approaches, particularly when they are applied on the FactBench and FELM datasets. On these datasets, methods incorporating knowledge consistently perform as well as or better than their version without knowledge, with the exception of SelfCheckGPT with BERTScore.

However, on FactAlign, knowledge integration does not always lead to superior performance. In particular, for SelfCheckGPT with NLI and few-shot prompting with five shots, the versions without integrated knowledge outperform those with knowledge. Despite this, on average, approaches incorporating knowledge still achieve better performance.

Nevertheless, the expected performance boost from knowledge integration was not fully realized. This may be attributed to limitations in the knowledge retrieval process, which only considers the URL of the first retrieved website, which is typically the most popular, but not necessarily the most informative. Additionally, retrieval issues may arise due to access restrictions, such as CAPTCHA codes or other barriers that prevent the successful extraction of relevant information.

Conclusions and Future Work

This chapter presents the conclusions based on the results obtained in the experiments, and outlines potential future directions and improvements to enhance the performance of hallucination detection approaches.

Conclusions

This study addresses a fundamental issue in the field of LLMs, that is the generation of hallucinations. Based on existing literature, a taxonomy of hallucinations was established, distinguishing between factuality hallucinations, where the generated content contradicts real-world knowledge, and faithfulness hallucinations, where the output is inconsistent with the provided input or context. This research focuses specifically on factuality hallucinations, as they pose a significant risk of spreading misinformation. Additionally, the study also examined the main causes of hallucinations, categorizing them into issues arising from data, training, and inference.

A comprehensive evaluation of various hallucination detection methods was conducted. The study examined supervised text classification approaches, LLM uncertainty estimation through the analysis of internal states, knowledge retrieval-based methods, and zero-resources and black-box approaches, which do not require access to external knowledge bases or internal states of the model, relying instead on strategies that analyze the consistency of the output, in order to detect hallucinations.

The results indicate that no single approach consistently outperforms others across all contexts. Supervised text classification methods demonstrate strong performance primarily in cases involving long responses, where the likelihood of contradictory information within the text is higher. In such instances, the textual representation effectively captures these inconsistencies, allowing for accurate hallucination classification. However, in the case of shorter texts, where the presence of contradictory information is less probable, supervised classification approaches exhibit reduced effectiveness.

With regard to internal state extraction and classification approaches, the most effective classifiers are traditional machine learning models, such as SVM and logistic regression. In contrast, DNNs tend to exhibit greater difficulty in the classification of the internal states. Moreover, performance appears to be influenced by the depth of the extracted layer. Layers closer to the output, that are focused on the generation of the next token, seem to contain more relevant information for classification, whereas those closer to the input layer, primarily responsible for extracting low-level features from the input, appear to provide limited information about the response. As a result, with these layers as input, the classifiers struggle to achieve accurate classification. Furthermore, providing

the classifier with all extracted layers from the LLM, appears to enhance overall classification performance.

As a zero-resources and black-box approach for hallucination detection, SelfCheckGPT, a state-of-the-art method based on response resampling, was tested, along with its variants. This method demonstrated strong overall performance across all datasets. Among its variants, the most effective and efficient was the one utilizing NLI models.

With regard to knowledge retrieval, an automated retrieval method was developed using the Google Search API. The retrieved knowledge was then integrated into SelfCheckGPT and few-shot prompting through a RAG framework. The results indicate that SelfCheckGPT with integrated knowledge consistently outperforms few-shot prompting with knowledge. This outcome is expected, as SelfCheckGPT is specifically designed for hallucination detection, whereas few-shot prompting serves as a more general method for solving various tasks. Additionally, the impact of knowledge integration on method performance was analyzed, revealing that, with few exceptions, approaches with integrated knowledge generally achieve superior results compared to their non-integrated counterparts. However, the performance improvement remains marginal, primarily due to limitations in knowledge retrieval quality. Factors such as restricted website access for automated bots and CAPTCHA protections pose significant issues, making the retrieval phase a key challenge in the context of knowledge-based hallucination detection methods.

Future Work

After analyzing the results obtained, it is essential to explore potential improvements to the proposed approaches, and consider possible future developments in the field of hallucination detection. In particular, the following potential improvements have been identified:

- **Combination of multiple detection strategies:** The results indicate that certain detection strategies can be highly effective in specific contexts while proving less reliable in others. To mitigate this variability, a possible approach would be to integrate multiple detection methods, and determine whether a response is factual or hallucinated based on a majority vote across the combined strategies. This ensemble approach could enhance overall accuracy by leveraging the strengths of different methods.
- **Improvements in the knowledge retrieval process:** The findings suggest that approaches integrating external knowledge generally outperform those that do not. However, the observed improvements remain limited due to constraints in the knowledge retrieval process. To address these limitations, the query submitted to the search engine could be refined using a LLM to generate more precise and contextually relevant search terms. Furthermore, the current approach, which considers only the first retrieved website, represents a significant limitation, as search engines rank results primarily based on popularity rather than informativeness. Since the top-ranked site may not always provide the most comprehensive or accurate information, expanding the number of retrieved sources (*e.g.*, considering the first five results) could enhance the informativeness and relevance of the retrieved knowledge.
- **Optimization of the uncertainty estimation method:** In the current uncertainty estimation approach based on the analysis of internal states, only the internal

states extracted from a specific LLM and from selected layers were considered. Future research could focus on identifying the optimal layer for extracting internal states, in order to maximize the informativeness contained in the hidden states and improve classification accuracy. Additionally, different LLM architectures could be explored to determine which model provides the most valuable information through its internal states. However, this improvement is significantly constrained by the fact that access to internal states is typically limited to open-source LLMs, as many proprietary models do not offer such transparency.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[2] D. Baidoo-Anu and L. O. Ansah, "Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning," *SSRN Electronic Journal*, 2023. DOI: 10.2139/ssrn.4337484.

[3] K. Girotra, L. Meincke, C. Terwiesch, and K. T. Ulrich, "Ideas are Dimes a Dozen: Large Language Models for Idea Generation in Innovation," *SSRN Electronic Journal*, 2023. DOI: 10.2139/ssrn.4526071.

[4] L. Athota, V. K. Shukla, N. Pandey, and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2020, pp. 619–622. DOI: 10.1109/ICRITO48877.2020.9197833.

[5] L. Belzner, T. Gabor, and M. Wirsing, "Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study," in *Bridging the Gap Between AI and Reality*, B. Steffen, Ed., Cham: Springer Nature Switzerland, 2024, pp. 355–374, ISBN: 978-3-031-46002-9.

[6] Y. Zhang, Y. Li, L. Cui, *et al.*, *Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models*, 2023. arXiv: 2309.01219 [cs.CL].

[7] P. Narayanan Venkit, T. Chakravorti, V. Gupta, *et al.*, "An Audit on the Perspectives and Challenges of Hallucinations in NLP," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 6528–6548. DOI: 10.18653/v1/2024.emnlp-main.375.

[8] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of Hallucination in Natural Language Generation," *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023, ISSN: 0360-0300. DOI: 10.1145/3571730.

[9] L. Huang, W. Yu, W. Ma, *et al.*, "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," *ACM Trans. Inf. Syst.*, Nov. 2024, Just Accepted, ISSN: 1046-8188. DOI: 10.1145/3703155.

[10] T. Browne, *Pseudodoxia Epidemica, Or, Enquiries Into Very Many Received Tenents, and Commonly Presumed Truths* (Pseudodoxia Epidemica; Or, Enquiries Into Very Many Received Tenents, and Commonly Presumed Truths). T. H., 1646.

[11] S. Baker and T. Kanade, "Hallucinating faces," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, 2000, pp. 83–88. DOI: 10.1109/AFGR.2000.840616.

[12] C.-C. Hsu, C.-W. Lin, C.-T. Hsu, H.-Y. M. Liao, and J.-Y. Yu, "Face hallucination using Bayesian global estimation and local basis selection," in *2010 IEEE International Workshop on Multimedia Signal Processing*, 2010, pp. 449–453. DOI: 10.1109/MMSP.2010.5662063.

[13] A. Karpathy, *The unreasonable effectiveness of recurrent neural networks*, https://karpathy.github.io/2015/05/21/rnn-effectiveness/, Accessed: 2025-01-22, 2015.

[14] N. Carlini, F. Tramer, E. Wallace, *et al.*, *Extracting Training Data from Large Language Models*, 2021. arXiv: 2012.07805 [cs.CR].

[15] A. Chowdhery, S. Narang, J. Devlin, *et al.*, "PaLM: scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, no. 1, Jan. 2023, ISSN: 1532-4435.

[16] A. C. Steere, J. Coburn, L. Glickstein, *et al.*, "The emergence of Lyme disease," *The Journal of clinical investigation*, vol. 113, no. 8, pp. 1093–1101, 2004.

[17] S. Li, X. Li, L. Shang, *et al.*, "How Pre-trained Language Models Capture Factual Knowledge? A Causal-Inspired Analysis," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1720–1732. DOI: 10.18653/v1/2022.findings-acl.136.

[18] C. Kang and J. Choi, "Impact of Co-occurrence on Factual Knowledge of Large Language Models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 7721–7735. DOI: 10.18653/v1/2023.findings-emnlp.518.

[19] *Raden Adjeng Kartini*, Jan. 2025.

[20] B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang, "Exposing attention glitches with flip-flop language modeling," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23, New Orleans, LA, USA: Curran Associates Inc., 2023.

[21] K. Arora, L. El Asri, H. Bahuleyan, and J. Cheung, "Why Exposure Bias Matters: An Imitation Learning Perspective of Error Accumulation in Language Generation," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 700–710. DOI: 10.18653/v1/2022.findings-acl.58.

[22] A. Azaria and T. Mitchell, "The Internal State of an LLM Knows When It's Lying," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 967–976. DOI: 10.18653/v1/2023.findings-emnlp.68.

[23] M. Sharma, M. Tong, T. Korbak, *et al.*, *Towards Understanding Sycophancy in Language Models*, 2023. arXiv: 2310.13548 [cs.CL].

[24] H. Zhang, D. Duckworth, D. Ippolito, and A. Neelakantan, "Trading Off Diversity and Quality in Natural Language Generation," in *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, A. Belz, S. Agarwal, Y. Graham, E. Reiter, and A. Shimorina, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 25–33.

[25] H.-S. Chang and A. McCallum, "Softmax Bottleneck Makes Language Models Unable to Represent Multi-mode Word Distributions," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8048–8073. DOI: 10.18653/v1/2022.acl-long.554.

[26] C. Chen and K. Shu, *Can LLM-Generated Misinformation Be Detected?* 2024. arXiv: 2309.13788 [cs.CL].

[27] N. Grigoriadou, M. Lymperaiou, G. Filandrianos, and G. Stamou, "AILS-NTUA at SemEval-2024 Task 6: Efficient model tuning for hallucination detection and analysis," in *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 1549–1560. DOI: 10.18653/v1/2024.semeval-1.222.

[28] T. Liu, Y. Zhang, C. Brockett, *et al.*, "A Token-level Reference-free Hallucination Detection Benchmark for Free-form Text Generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6723–6737. DOI: 10.18653/v1/2022.acl-long.464.

[29] S. Min, K. Krishna, X. Lyu, *et al.*, "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12 076–12 100. DOI: 10.18653/v1/2023.emnlp-main.741.

[30] L. Jing, R. Li, Y. Chen, and X. Du, "FaithScore: Fine-grained Evaluations of Hallucinations in Large Vision-Language Models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 5042–5063. DOI: 10.18653/v1/2024.findings-emnlp.290.

[31] Y. Song, Y. Kim, and M. Iyyer, "VeriScore: Evaluating the factuality of verifiable claims in long-form text generation," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 9447–9474. DOI: 10.18653/v1/2024.findings-emnlp.552.

[32] V. T. Kim, M. Krumdick, V. Reddy, F. Dernoncourt, and V. D. Lai, "An Analysis of Multilingual FActScore," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 4309–4333. DOI: 10.18653/v1/2024.emnlp-main.247.

[33] S. Huo, N. Arabzadeh, and C. Clarke, "Retrieving Supporting Evidence for Generative Question Answering," in *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, ser. SIGIR-AP '23, Beijing, China: Association for Computing Machinery, 2023, pp. 11–20, ISBN: 9798400704086. DOI: 10.1145/3624918.3625336.

[34] T. Nguyen, M. Rosenberg, X. Song, *et al.*, "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset," *CoRR*, vol. abs/1611.09268, 2016. arXiv: 1611.09268.

[35] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Overview of the Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD: NIST, Jan. 1995, pp. 109–126.

[36] X. Hu, D. Ru, L. Qiu, *et al.*, "Knowledge-Centric Hallucination Detection," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 6953–6975. DOI: 10.18653/v1/2024.emnlp-main.395.

[37] T. Kwiatkowski, J. Palomaki, O. Redfield, *et al.*, "Natural Questions: A Benchmark for Question Answering Research," *Transactions of the Association for Computational Linguistics*, vol. 7, L. Lee, M. Johnson, B. Roark, and A. Nenkova, Eds., pp. 452–466, 2019. DOI: 10.1162/tacl_a_00276.

[38] M. Conover, M. Hayes, A. Mathur, *et al.* "Dolly: First Open Commercially Viable Instruction-Tuned LLM." (2023), [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

[39] Y. Xiao and W. Y. Wang, "On hallucination and predictive uncertainty in conditional language generation," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 2734–2744. DOI: 10.18653/v1/2021.eacl-main.236.

[40] B. Snyder, M. Moisescu, and M. B. Zafar, "On Early Detection of Hallucinations in Factual Question Answering," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24, Barcelona, Spain: Association for Computing Machinery, 2024, pp. 2721–2732, ISBN: 9798400704901. DOI: 10.1145/3637528.3671796.

[41] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[42] H. Elsahar, P. Vougiouklis, A. Remaci, *et al.*, "T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. Calzolari, K. Choukri, C. Cieri, *et al.*, Eds., Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[43] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1601–1611. DOI: 10.18653/v1/P17-1147.

[44] C. Chen, K. Liu, Z. Chen, *et al.*, *INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection*, 2024. arXiv: 2402.03744 [cs.CL].

[45] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[46] S. Reddy, D. Chen, and C. D. Manning, "CoQA: A Conversational Question Answering Challenge," *Transactions of the Association for Computational Linguistics*, vol. 7, L. Lee, M. Johnson, B. Roark, and A. Nenkova, Eds., pp. 249–266, 2019. DOI: 10.1162/tacl_a_00266.

[47] Z. Ji, D. Chen, E. Ishii, *et al.*, "LLM Internal States Reveal Hallucination Risk Faced With a Query," in *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, Y. Belinkov, N. Kim, J. Jumelet, H. Mohebbi, A. Mueller, and H. Chen, Eds., Miami, Florida, US: Association for Computational Linguistics, Nov. 2024, pp. 88–104. DOI: 10.18653/v1/2024.blackboxnlp-1.6.

[48] W. Su, C. Wang, Q. Ai, *et al.*, "Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 14 379–14 391. DOI: 10.18653/v1/2024.findings-acl.854.

[49] P. Manakul, A. Liusie, and M. Gales, "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9004–9017. DOI: 10.18653/v1/2023.emnlp-main.557.

[50] R. Lebret, D. Grangier, and M. Auli, "Neural Text Generation from Structured Data with Application to the Biography Domain," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1203–1213. DOI: 10.18653/v1/D16-1128.

[51] J. Zhang, Z. Li, K. Das, B. Malin, and S. Kumar, "SAC^3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 15 445–15 458. DOI: 10.18653/v1/2023.findings-emnlp.1032.

[52] Z. Yang, P. Qi, S. Zhang, *et al.*, "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380. DOI: 10.18653/v1/D18-1259.

[53] R. Cohen, M. Hamri, M. Geva, and A. Globerson, "LM vs LM: Detecting Factual Errors via Cross Examination," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 12 621–12 640. DOI: 10.18653/v1/2023.emnlp-main.778.

[54] F. Petroni, P. Lewis, A. Piktus, *et al.*, *How Context Affects Language Models' Factual Predictions*, 2020. arXiv: 2005.04611 [cs.CL].

[55] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 9802–9822. DOI: 10.18653/v1/2023.acl-long.546.

[56] Y. Yehuda, I. Malkiel, O. Barkan, J. Weill, R. Ronen, and N. Koenigstein, "InterrogateLLM: Zero-Resource Hallucination Detection in LLM-Generated Answers," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 9333–9347. DOI: 10.18653/v1/2024.acl-long.506.

[57] R. Banik, *The Movies Dataset*, https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset.

[58] O. Monk, *Books Dataset*, https://www.kaggle.com/datasets/saurabhbagchi/books-dataset.

[59] N. Elgiriyewithana, *Global Country Information Dataset 2023*, https://www.kaggle.com/datasets/nelgiriyewithana/countries-of-the-world-2023.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[61] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15, USA: IEEE Computer Society, 2015, pp. 19–27, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.11.

[62] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language Models are Unsupervised Multitask Learners*, 2019.

[63] S. Nagel, *News Dataset Available*, https://commoncrawl.org/blog/news-dataset-available, 2016.

[64] A. Gokaslan and V. Cohen, *OpenWebText Corpus*, http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[65] T. H. Trinh and Q. V. Le, *A Simple Method for Commonsense Reasoning*, 2019. arXiv: 1806.02847 [cs.AI].

[66] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[67] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152, ISBN: 089791497X. DOI: 10.1145/130385.130401.

[68] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[69] J. Cramer, "The Origins of Logistic Regression," *Tinbergen Institute, Tinbergen Institute Discussion Papers*, Jan. 2002. DOI: 10.2139/ssrn.360300.

[70] D. R. Cox, "The Regression Analysis of Binary Sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, Dec. 2018, ISSN: 0035-9246. DOI: 10.1111/j.2517-6161.1958.tb00292.x. eprint: https://academic.oup.com/jrsssb/article-pdf/20/2/215/49096552/jrsssb_20_2_215.pdf.

[71] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003.

[72] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, ISSN: 1935-8237. DOI: 10.1561/2200000006.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[74] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015.

[75] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *BERTScore: Evaluating Text Generation with BERT*, 2020. arXiv: 1904.09675 [cs.CL].

[76] H. Touvron, T. Lavril, G. Izacard, *et al.*, *LLaMA: Open and Efficient Foundation Language Models*, 2023. arXiv: 2302.13971 [cs.CL].

[77] A. Grattafiori, A. Dubey, A. Jauhri, *et al.*, *The Llama 3 Herd of Models*, 2024. arXiv: 2407.21783 [cs.AI].

[78] A. Williams, N. Nangia, and S. Bowman, "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122.

[79] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

[80] Y. Gao, Y. Xiong, X. Gao, *et al.*, *Retrieval-Augmented Generation for Large Language Models: A Survey*, 2024. arXiv: 2312.10997 [cs.CL].

[81] S. Sturua, I. Mohr, M. K. Akram, *et al.*, *jina-embeddings-v3: Multilingual Embeddings With Task LoRA*, 2024. arXiv: 2409.10173 [cs.CL].

[82] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-Pack: Packed Resources For General Chinese Embeddings," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '24, Washington DC, USA: Association for Computing Machinery, 2024, pp. 641–649, ISBN: 9798400704314. DOI: 10.1145/3626772.3657878.

[83] J. Li, X. Cheng, X. Zhao, J.-Y. Nie, and J.-R. Wen, "HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 6449–6464. DOI: 10.18653/v1/2023.emnlp-main.397.

[84] C.-W. Huang and Y.-N. Chen, "FactAlign: Long-form Factuality Alignment of Large Language Models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 16 363–16 375. DOI: 10.18653/v1/2024.findings-emnlp.955.

[85] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, and D. Kiela, "Model alignment as prospect theoretic optimization," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, Vienna, Austria: JMLR.org, 2024.

[86] Y. Wang, R. Gangi Reddy, Z. M. Mujahid, *et al.*, "Factcheck-Bench: Fine-Grained Evaluation Benchmark for Automatic Fact-checkers," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 14 199–14 230. DOI: 10.18653/v1/2024.findings-emnlp.830.

[87] I.-C. Chern, S. Chern, S. Chen, *et al.*, *FacTool: Factuality Detection in Generative AI – A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios*, 2023. arXiv: 2307.13528 [cs.CL].

[88] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3252. DOI: 10.18653/v1/2022.acl-long.229.

[89] S. Chen, Y. Zhao, J. Zhang, *et al.*, "FELM: Benchmarking Factuality Evaluation of Large Language Models," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 44 502–44 523.

[90] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 24 824–24 837.