Review

Not peer-reviewed version

From Illusion to Insight: A Taxonomic Survey of Hallucination Mitigation Techniques in LLMs

Ioannis Kazlaris * , Efstathios Antoniou , Konstantinos Diamantaras , Charalampos Bratsas

Posted Date: 27 August 2025

doi: 10.20944/preprints202508.1942.v1

Keywords: mitigation; hallucinations; Large Language Models (LLMs); taxonomy

Review

From Illusion to Insight: A Taxonomic Survey of Hallucination Mitigation Techniques in LLMs

Ioannis Kazlaris *, Efstathios Antoniou, Konstantinos Diamantaras and Charalampos Bratsas

Department of Information and Electronic Engineering, International Hellenic University (IHU),
57400 Thessaloniki, Greece; antoniou@ihu.gr (E.A.); kdiamant@ihu.gr (K.D.); cbratsas@ihu.gr (C.B.)
* Correspondence: ikazlaris@ihu.gr

Abstract

Large Language Models (LLMs) exhibit remarkable generative capabilities but remain susceptible to hallucinations—outputs that are fluent yet inaccurate, ungrounded, or inconsistent with source material. This paper presents a method-oriented taxonomy of hallucination mitigation strategies in text-based Large Language Models (LLMs), encompassing six categories: Training and Learning Approaches, Architectural Modifications, Input / Prompt Optimization, Post-Generation Quality Control, Interpretability and Diagnostic Methods, and Agent-Based Orchestration. By synthesizing over 300 studies, we identify persistent challenges including the lack of standardized evaluation benchmarks, attribution difficulties in multi-method frameworks, computational trade-offs between accuracy and latency, and the vulnerability of retrieval-based methods to noisy or outdated sources. We highlight underexplored research directions such as knowledge-grounded fine-tuning strategies balancing factuality with creative utility; and hybrid retrieval–generation pipelines integrated with self-reflective reasoning agents. This taxonomy offers both a synthesis of current knowledge and a roadmap for advancing reliable, context-sensitive mitigation in high-stakes domains such as healthcare, law, and defense.

Keywords: mitigation; hallucinations; Large Language Models (LLMs); taxonomy

1. Introduction

In this paper, we examine hallucinations in Large Language Models (LLMs), defined as instances where generated content appears coherent and plausible but contains factual inaccuracies or unverifiable claims [1,12,13]. These multifaceted manifestations can range from fabricated citations and logical inconsistencies to erroneous statistics and invented biographical details [87,292] , thus rendering problematic the use of LLMs in applications where factuality and accuracy are of utmost importance, such as healthcare [93,197], law [135,173,177], and defense [195]. Some of the underlying causes include, but are not limited to, noisy training data, underrepresentation of minority viewpoints, outdated information, and the next-token log-likelihood objective that prioritizes plausible continuation over factual accuracy. Research has shown that hallucinations arise from the complex interplay between factual accuracy and generative capability [45,130,186]. Despite being mathematically inevitable, a plethora of mitigation strategies have been developed across pre-generation, generation, and post-generation phases. Pre-generation techniques may include such as fine-tuning with human feedback [307] and contrastive learning [65] (distinct from contrastive decoding, which is applied during text generation), while architectural enhancements incorporate retrieval-based modules [259] and memory augmentations [168] to ground content in verifiable sources. During generation, structured prompting techniques such as Chain of Thought (CoT) [53], guide reasoning toward evidence-based conclusions while post-generation safeguards employ self-consistency checks [268], self-verification and fact-checking systems [267], and self-verification loops [121,268], along with human-in-the-loop evaluations [118] to detect and rectify factual inconsistencies, while internal model probing techniques and lightweight classifiers detect hallucination patterns through latent signals and linguistic cues [82,156]. AI agentic frameworks where a number of agents consult external tools, negotiate and self-reflect on their decisions complete our taxonomy [199,253].

We organize the remainder of this paper as follows: section 0 defines and categorizes hallucinations, discusses their underlying causes, and provides a brief overview of existing mitigation methods. Related research studies are presented in section 0, while our research methodology, the proposed taxonomy and the contributions of this paper are outlined in section 0. An analytical discussion of mitigation methods is presented in section 0. Section 0 outlines benchmarks for evaluating hallucinations in LLMs. A brief-outline for high-stake applications is presented in 0. Finally, section 0 discusses the challenges in addressing hallucinations, and section 0 concludes the paper.

2. Understanding Hallucinations

2.1. Definition of Hallucinations

Hallucinations in LLMs refer to instances where the generated content appears grammatically correct and coherent, yet it is factually incorrect, irrelevant, ungrounded (i.e., cannot be traced to reliable sources), or logically inconsistent [1,12,13]. Distinct from simple, obvious errors and biases, hallucinations frequently manifest themselves as false claims about real or imaginary entities (people, events, places, facts), fabricated citations and data (for instance, non-existent biographical details) [87,292], erroneous statistics and inaccurate numerical values or conclusions that do not follow logically from their premises [12,286]. Defining hallucinations is challenging because their classification depends heavily on the task, the logical relation between input and output, and pragmatic factors such as ambiguity, vagueness, presupposition, or metaphor [246,298]. The presence of hallucinations may be attributed to a multi-faceted and complex interplay of various factors such as sub-optimal training data [73,238], pre-training and supervised fine-tuning issues [281,304,306], or the probabilistic nature of sequence generation [45,130,223] as will be further analyzed in 0. While significant advancements have occurred in LLMs, hallucinatory output still poses significant concerns over the reliability and trustworthiness of machine-generated text in both academic and industrial contexts and especially in critical domains such as healthcare [197], law [135,177] and defense [195], where significant harm may be caused to individuals.

However, it can be argued that hallucinations in LLMs represent not merely technical limitations but rather fundamental mathematical inevitabilities inherent to their architecture and function: Training LLMs for predictive accuracy inevitably results in hallucinations, regardless of model architecture or data quality, due to the probabilistic nature of language generation and the inherent limitation in LLMs to learn all computable functions as is shown in [45,130,186] and additionally supported by computational learning theory constraints and the absence of complete factual mappings [45,130]. This tension arises because the next-token prediction objective favors statistically likely continuations, prioritizing linguistic plausibility over epistemic truth.

Despite this paradox between factual accuracy and probabilistic capability, a number of research papers challenge their characterization as universal limitations by positing that this inevitability is not uniformly spread across all contexts or tasks [39] and that hallucinations are not mere artifacts but exploitable features for adversarial robustness [184]. Research has shown that intentional hallucination generates out-of-distribution concepts which boost performance in tasks such as poetry/storytelling by 24% via associative chain disruption thus leading to more unexpected narratives or even to extreme confabulation patterns if unregulated [126] while analysis on layer-wise intermediate generations suggests that there exist specific layers in the architecture of an LLM where hallucinations and creativity can be balanced [278]. Similarly, in tasks such as hypothesis generation in scientific inquiry or brainstorming ideas, the unlikely combinations of entities or reasoning processes produced by hallucinations may uncover insights or ideas that would not emerge through conventional methods as shown in [192] where combinatorial entropy can produce viable hypotheses in physics and biology or in [303] where controlled hallucination is used for brainstorming via reflective prompting. Given this dual nature of hallucinations, we believe that their evaluation needs to occur within the context of specific applications and under cultural sensitivity awareness, so as to distinguish between scenarios that demand factual rigor and those that benefit from creative exploration [3,14,21].

2.2. Categories of Hallucinations

We can categorize hallucinations in LLMs into distinct types based on their relationship to source material, model knowledge boundaries, and linguistic structure. Following the majority of researchers, we distinguish between intrinsic hallucinations (factuality errors) and extrinsic hallucinations (faithfulness errors) as foundational categories [6,12,13,62,131,137,286,315] and distinct from other phenomena such as bias, which some view as a hallucination subtype [12,13,165,203,279] while others approach as a separate representational skew not necessarily involving factual error [1,21,28]. Furthermore, we explicitly position factuality (intrinsic) and faithfulness (extrinsic) hallucinations as subcategories of the broader intrinsic/extrinsic taxonomy, thus aligning with research in [219] which emphasizes that factuality requires alignment with external truths, while faithfulness requires alignment with source material. Acknowledging the research of [300], where a task-independent categorization of hallucinations is presented (Factual Mirage and Silver Lining in addition to their subcategories), we summarize the major categories thus:

- Intrinsic hallucinations (factuality errors) occur when a model generates content that contradicts established facts, its training data, or referenced input [6,12,13,62,137,286,315]. Following the taxonomic names in [292] the subtypes of this category may include (but are not limited to):
    - Entity-error hallucinations, where the model generates non-existent entities or misrepresents their relationships (e.g., inventing fake individuals, non-existent biographical details [87] or non-existent research papers), often measured via entity-level consistency metrics [98], as shown in [13,28,208,286].
    - Relation-error hallucinations, where inconsistencies are of a temporal, causal, or quantitative nature, such as erroneous chronologies or unsupported statistical claims [12,28,286].
    - Outdateness hallucinations, which are characterized by outdated or superseded information, often reflecting a mismatch between training data and current knowledge as shown in [1,13,117,131].
    - Overclaim hallucinations, where models exaggerate the scope or certainty of a claim, often asserting more than the evidence supports [6,28,279,292].
- Extrinsic hallucinations (faithfulness errors) appear when the generated content deviates from the provided input or user prompt. These hallucinations are generally characterized by the inability to verify the generated output which may or may not be true but, in either case, it is either not directly deducible from the user prompt or it contradicts itself [12,13,62,219,279,292]. Extrinsic hallucinations may manifest as:
    - Incompleteness hallucinations, which occur when answers are truncated or necessary context that is useful is omitted [12,28,131].
    - Unverifiability hallucinations, where outputs that neither align with available evidence nor can be clearly refuted are present [279,292].
    - Emergent hallucinations, defined as those arising unpredictably in larger models due to scaling effects [92]. These can be attributed to cross-domain reasoning and modality fusion especially in multi-modal settings or Chain of Thought (CoT) prompting scenarios [13,92,123], multi-step inference errors [147] and abstraction or alignment issues as shown in [28,62] and [123]. For instance, self-reflexion demonstrates mitigation capabilities, effectively reducing hallucinations only in models above a certain threshold (e.g., 70B parameters), while paradoxically increasing errors in smaller models due to limited self-diagnostic capacity [292].

More importantly, certain types of hallucinations can compound over time in lengthy prompts or multi-step reasoning tasks, thus triggering a cascade of errors. This phenomenon, known as the snowball effect, further erodes user trust and model reliability [147,182,269].

2.3. Underlying Causes of Hallucinations

Hallucinations in LLMs stem from a variety of interwoven factors spanning the entire development pipeline. Although the impact of each cause may vary depending on the task and model architecture, we present them here in a non-hierarchical format to emphasize their interconnectedness. These factors include (but are not limited to) model architecture and decoding strategies [45,130,186,223,292], data quality, bias, memorization and poor alignment [27,73,223,238,304,305,307], pre-training and supervised fine-tuning issues [281,304,306], compute scale issues and under-training [292,306], prompt design and distribution shift [169,232,326], and retrieval-related mismatches [190,257].

In terms of model architecture, the use of decoding strategies and the inherently probabilistic nature of sequence-to-sequence modeling is a primary cause of hallucinations. For instance, the next-token log-likelihood objective commonly used during training which prioritizes plausible continuation over factual accuracy [45,130] has led to a significant body of research on decoding strategies such as nucleus sampling [291], contrastive decoding [64], and confidence-aware decoding [284] among others. Recent work also connects hallucinations to a lack of explicit modeling of uncertainty or factual confidence [48,275,314].

A model's vast and oftentimes noisy training data may often suffer from the inclusion of inaccuracies, contradictions (due to the presence of entirely or partially conflicting data also known as knowledge overshadowing), and biases, the presence of which may hinder the model's ability to generate factually reliable content [165,270]. While biases can cause hallucinations, and hallucinations represent a significant category of LLM output errors, in this review we specifically examine the nature and mitigation of generated text that is factually ungrounded or inconsistent. If, for instance, a model is trained on imbalanced data, i.e. data skewed towards a majority perspective, favoring specific cultural differences or even personal beliefs, it might generate content that contradicts the realities or experiences of minority groups due to lack of exposure to their narratives [3,63,223]. These biases may be attributed to the underrepresentation of specific groups or ideas, making it challenging for models to generate content that is truly ethical and free from prejudice.

Data duplication and repetition, both of which may lead to overfitting and memorization issues, further degenerate the quality of the model output as shown in [73] and [238]. Ambiguous data may hinder the quantification or elicitation of model uncertainty, with research suggesting that scaling, although generally beneficial, may have reached a fundamental limit, where models merely memorizing disambiguation patterns instead of reasoning [27]. Additionally, preference-fine-tuning achieved with Reinforcement Learning from Human Feedback (RLHF) can reward fluent but significantly polarized answers, a form of alignment-induced hallucination [306]. This kind of RL-based preference optimization has been linked to sycophantic agreement with user misconceptions further exacerbating factual inconsistencies [304] while recent work demonstrates that such hallucinations arise partly from reward hacking during alignment, where models prioritize reward signals over truthfulness and are addressed with constrained fine-tuning methods [208].

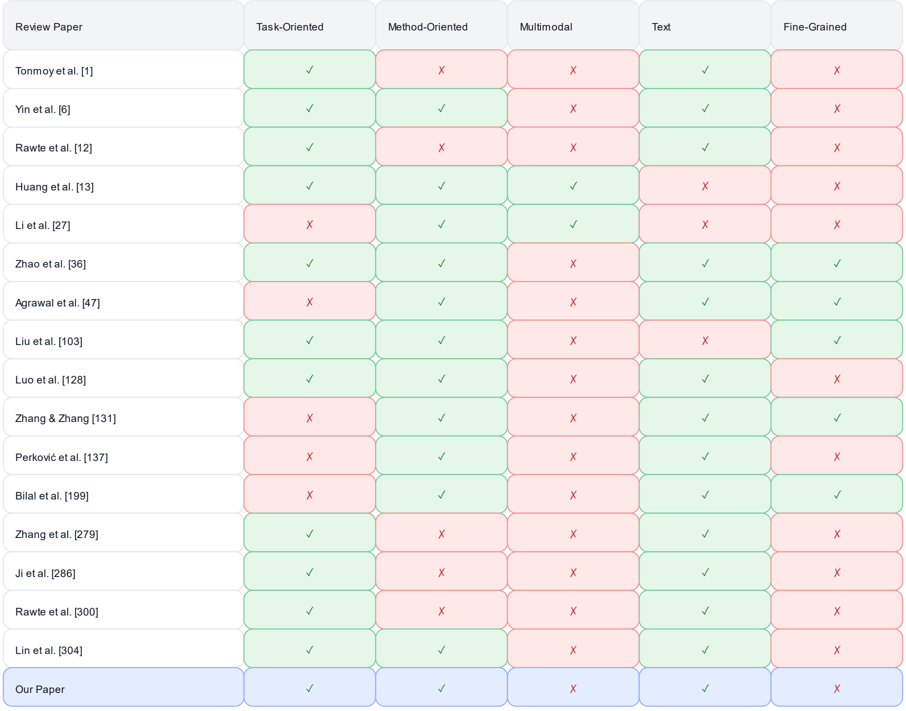
The high computational cost and prolonged training times required for training LLMs are additional factors that affect the quality of LLM generation: training data are not frequently updated thus hindering their ability to provide up-to-date and accurate information [304,306]. Controlled experiments using LLM behavioral probing have also demonstrated that verbatim memorization of individual statements at the sentence level as well as statistical patterns of usage at the corpora level (template-based reproduction) can also trigger hallucinations [238,281], while another line of research has demonstrated that many LLMs are under-trained, and that optimal performance requires scaling both model size and the number of training tokens proportionally [306].

In terms of prompting techniques, poor or insufficient prompting can significantly contribute to the generation of hallucinations since the omission of important information or failure to restrict the model's output can introduce ambiguity [75]. In many such cases, the model defaults to probabilistic pattern-matching which favors generalization over user intent as there is little or no information in the prompt that would enable it to deliver a more concise and targeted response [169,232,326]. Adversarial or out-of-distribution prompts can also have a detrimental effect as shown in [53] and [75] while in long-context models, inaccuracies may also emerge from summarization or retrieval failures during extended sequence attention, which can cause the model to lose track of grounding over time [269,296]. Furthermore, low-resource settings lack mitigation options requiring fine-tuning; zero-shot detection methods like self-consistency probes [329] or hesitation analysis [117] detect hallucinations without significant computational overhead.

3. Related Works

Research on hallucinations in LLMs has expanded rapidly, offering a diverse range of perspectives on their causes, forms, and mitigation strategies. Several surveys provide comprehensive overviews of hallucination mitigation across the LLM pipeline, from pre-generation to post-generation phases as seen in [1,6,12,13,128,286,300]. Much of the existing literature has adopted a task-oriented taxonomy, categorizing mitigation techniques according to downstream applications (e.g., summarization, question answering, code generation) [103] or by the level of system intervention (e.g., pre-training, fine-tuning, or decoding) [12,279,286,300]. While this body of research offers valuable insights into where hallucinations occur and what tasks they affect, it often underrepresents the how—i.e., the core methodological principles underlying mitigation strategies. Some works have begun to address this by introducing method-oriented taxonomies which classify strategies based on techniques such as data augmentation, retrieval integration, prompt engineering or external fact-checking [13,137,304]. Notably, some of these studies focus on specific mitigation classes such as retrieval-augmented generation [36,131] ambiguity and uncertainty estimation [27], knowledge graphs [47], multi-agent reinforcement learning [199], thus offering deep insights into a narrow family of techniques. Others, like [13], begin to outline broad method-based categories but do not always explore fine-grained distinctions such as whether a method is self-supervised, classifier-guided, contrastive, or interpretability-driven.

As Figure 1 illustrates, our paper aims to complement this current body of research by offering an updated, method-oriented taxonomy that focuses exclusively on hallucination mitigation in textual LLMs. Our taxonomy is designed to clarify strategies based on their methodological foundation—such as training and learning approaches, architectural modifications, prompt engineering, output verification, post-processing, interpretability, and agent-based orchestration. By delving into the structural properties of existing mitigation strategies, we hope to clarify overlapping concepts and identify gaps that may serve as opportunities for future research.



Review Paper	Task-Oriented	Method-Oriented	Multimodal	Text	Fine-Grained
Tonmoy et al. [1]	✓	✗	✗	✓	✗
Yin et al. [6]	✓	✓	✗	✓	✗
Rawte et al. [12]	✓	✗	✗	✓	✗
Huang et al. [13]	✓	✓	✓	✗	✗
Li et al. [27]	✗	✓	✓	✗	✗
Zhao et al. [36]	✓	✓	✗	✓	✓
Agrawal et al. [47]	✗	✓	✗	✓	✓
Liu et al. [103]	✓	✓	✗	✗	✓
Luo et al. [128]	✓	✓	✗	✓	✗
Zhang & Zhang [131]	✗	✓	✗	✓	✓
Perković et al. [137]	✗	✓	✗	✓	✗
Bilal et al. [199]	✗	✓	✗	✓	✓
Zhang et al. [279]	✓	✗	✗	✓	✗
Ji et al. [286]	✓	✗	✗	✓	✗
Rawte et al. [300]	✓	✗	✗	✓	✗
Lin et al. [304]	✓	✓	✗	✓	✗
Our Paper	✓	✓	✗	✓	✗

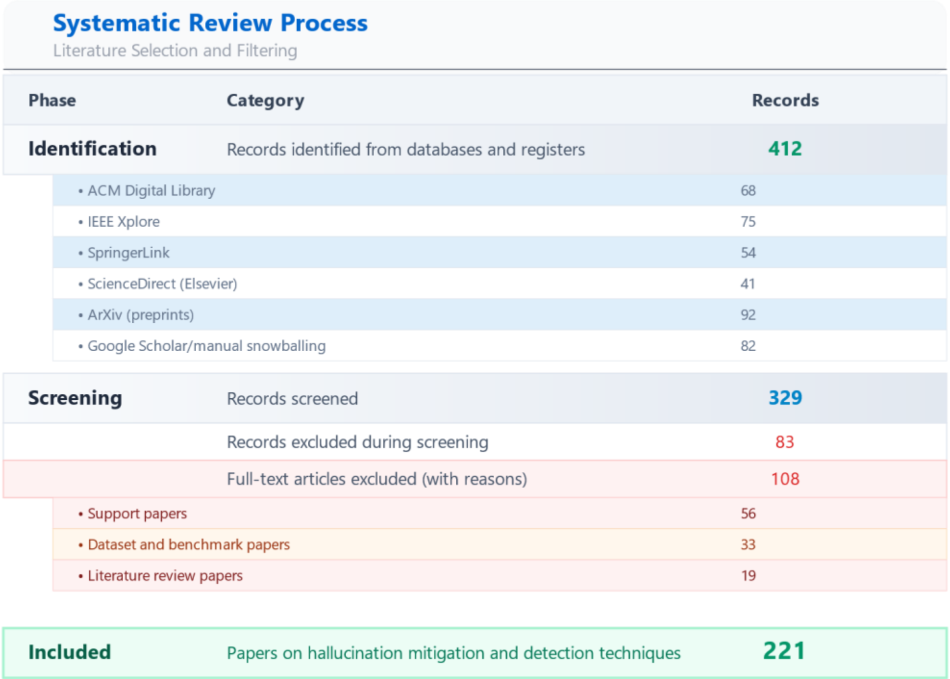
Figure 1. Comparison of hallucination survey taxonomies in LLM research, highlighting differences in categorization schemes such as task-oriented, system-level, and method-based classifications.

4. Review Methodology, Proposed Taxonomy, Contributions and Limitations

4.1. Review Methdology

Defining a single, unchanging taxonomy to classify and address hallucinations is a challenging task that can be attributed at least partially to the complex and interwoven nature of the factors that cause them. In this paper, we try to address this challenge by presenting a 5-stage hierarchical framework:

- Literature Retrieval: We systematically collected research papers from major electronic archives—including Google Scholar, ACM Digital Library, IEEE Xplore, Elsevier, Springer, and ArXiv—with a cutoff date of August 12, 2025. Eligible records were restricted to peer-reviewed journal articles, conference papers, preprints under peer review, and technical reports, while non-academic sources such as blogs or opinion pieces were excluded. A structured query was used, combining keywords: ("mitigation" AND "hallucination" AND "large language models") OR "evaluation". In addition, we examined bibliographies of retrieved works to identify further relevant publications.
- Screening: The screening process followed a two-stage approach. First, titles and abstracts were screened for topical relevance. Records passing this stage underwent a full-text review to assess eligibility. Out of 412 initially retrieved records, 83 were excluded as irrelevant at the screening stage. The 329 eligible papers were then examined in detail and further categorized into support studies, literature reviews, datasets/benchmarks, and works directly proposing hallucination detection or mitigation methods. The final set of 221 studies formed the basis of our taxonomy. This process is summarized in the PRISMA-style diagram below.



Systematic Review Process

Literature Selection and Filtering

Phase	Category	Records
Identification	Records identified from databases and registers	412
	• ACM Digital Library	68
	• IEEE Xplore	75
	• SpringerLink	54
	• ScienceDirect (Elsevier)	41
	• ArXiv (preprints)	92
	• Google Scholar/manual snowballing	82
Screening	Records screened	329
	Records excluded during screening	83
	Full-text articles excluded (with reasons)	108
	• Support papers	56
	• Dataset and benchmark papers	33
	• Literature review papers	19
Included	Papers on hallucination mitigation and detection techniques	221

Figure 2. PRISMA flow diagram of study selection.

- Paper-level tagging, where every study was assigned one or more tags corresponding to its employed mitigation strategies. Our review accounts for papers that propose multiple methodologies by assigning them multiple tags, ensuring a comprehensive representation of each paper's contributions.
- Thematic clustering, where we consolidated those tags into six broad categories presented analytically in 0. This enabled us to generate informative visualizations that reflect the prevalence and trends among different mitigation techniques.
- Content-specific retrieval: To gain deeper insight into mitigation strategies, we developed a custom Retrieval-Augmented Generation (RAG) system based on the Mistral language model as an additional research tool, which enabled us to extract content-specific passages directly from the research papers.

4.2. Proposed Taxonomy and Review Organization

Building upon the systematic methodology outlined in 0, our review directly addresses the challenges inherent in classifying diverse hallucination mitigation techniques. A key difficulty is that the majority of the papers implement a number of strategies (oftentimes referred to as *frameworks*), and therefore it is not always clearly discernible what the primary mitigation method is and to what extent its results can be safely attributed to it. We acknowledge this fact by explaining at the start of every major category our placement rationale. More specifically, our proposed taxonomy categorizes hallucination mitigation strategies into six primary, method-oriented categories briefly outlined in Figure 3 and more analytically in the Appendix A.

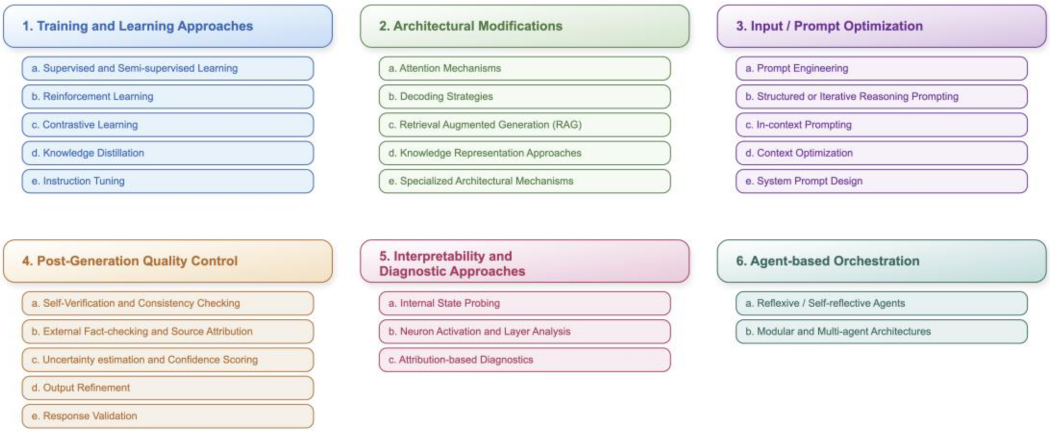

1. Training and Learning Approaches
a. Supervised and Semi-supervised Learning
b. Reinforcement Learning
c. Contrastive Learning
d. Knowledge Distillation
e. Instruction Tuning
2. Architectural Modifications
a. Attention Mechanisms
b. Decoding Strategies
c. Retrieval Augmented Generation (RAG)
d. Knowledge Representation Approaches
e. Specialized Architectural Mechanisms
3. Input / Prompt Optimization
a. Prompt Engineering
b. Structured or Iterative Reasoning Prompting
c. In-context Prompting
d. Context Optimization
e. System Prompt Design
4. Post-Generation Quality Control
a. Self-Verification and Consistency Checking
b. External Fact-checking and Source Attribution
c. Uncertainty estimation and Confidence Scoring
d. Output Refinement
e. Response Validation
5. Interpretability and Diagnostic Approaches
a. Internal State Probing
b. Neuron Activation and Layer Analysis
c. Attribution-based Diagnostics
6. Agent-based Orchestration
a. Reflexive / Self-reflective Agents
b. Modular and Multi-agent Architectures

Figure 3. Visualization of the proposed taxonomy for hallucination mitigation, organizing strategies into six method-based categories.

1. Training and Learning Approaches (0): Encompasses diverse methodologies employed to train and refine AI models, shaping their capabilities and performance (e.g., Supervised Learning, Reinforcement Learning, Knowledge Distillation).
2. Architectural Modifications (0): Covers structural changes and enhancements made to AI models and their inference processes to improve performance, efficiency, and generation quality (e.g., Attention Mechanisms, Decoding Strategies, Retrieval Augmented Generation).
3. Input/Prompt Optimization (0): Focuses on strategies for crafting and refining the text provided to AI models to steer their behavior and output, often specifically to mitigate hallucinations (e.g., Prompt Engineering, Context Optimization).
4. Post-Generation Quality Control (0): Encompasses essential post-generation checks applied to text outputs, aiming to identify or correct inaccuracies (e.g., Self-verification, External Fact-checking, Uncertainty Estimation).
5. Interpretability and Diagnostic Approaches (0): Encompasses methods that help researchers understand why and where a model may be hallucinating (e.g., Internal State Probing, Attribution-based diagnostics).
6. Agent-based Orchestration (0): Includes frameworks comprising single or multiple LLMs within multi-step loops, enabling iterative reasoning, tool usage, and dynamic retrieval (e.g., Reflexive Agents, Multi-agent Architectures).

In section 0, we provide a brief overview for each of these six categories, describing its general principles. Following this, we present and briefly discuss research papers that implement methods within that category. To better clarify the idiosyncrasies of each of the methods presented, we also cite “support papers”, i.e. papers that do not directly target the mitigation of hallucinations but instead provide useful information, evidence or results regarding the mitigation method in question.

4.3. Contributions and Key Findings

To complement our taxonomy, we present three visualizations that synthesize key trends in hallucination mitigation research and may indicate potential future research directions.

Figure 4 depicts the distribution of research papers across individual mitigation techniques. Unsurprisingly, Prompt Engineering dominates the landscape, reflecting its low-cost, flexible nature and its prominence in guiding model behavior without retraining. It is followed by strategies such as Supervised and Semi-supervised Learning, External Fact-checking and Output Refinement, all of which indicate the field’s growing interest in refining or verifying outputs post hoc. Less populated areas such as Self-reflective Agents and Knowledge Distillation suggest areas that require further exploration.

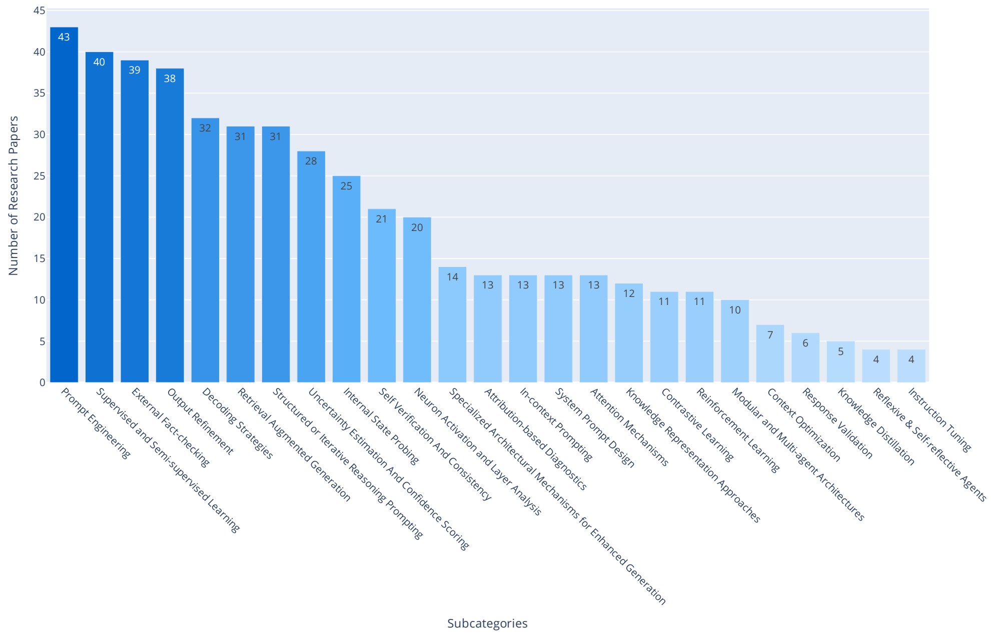


Figure 4. Distribution of research papers across individual hallucination mitigation subcategories, showing the prevalence of methods such as prompt engineering, external fact-checking, and decoding strategies.

Figure 5 traces the temporal growth of hallucination mitigation research. The trendline shows a noticeable surge in publications from 2022 onward, particularly following the public attention garnered by generative LLMs like ChatGPT. Notably, over 100 papers were published in 2023 alone, signaling both the urgency and the complexity of the hallucination problem as LLMs become integrated into high-stakes applications.

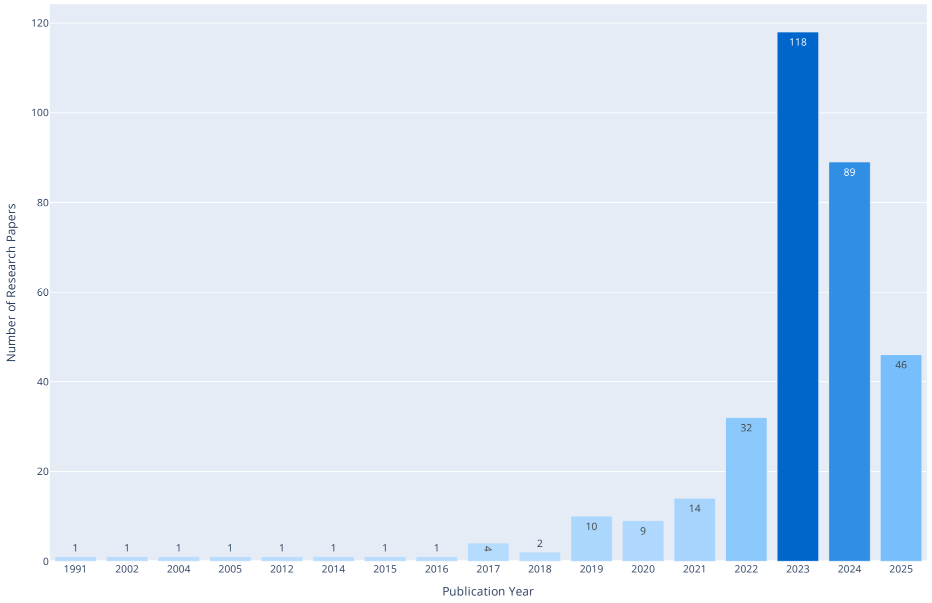


Figure 5. Yearly publication trend of hallucination mitigation research in LLMs, indicating a sharp increase in scholarly output since 2022.

Figure 6 aggregates the number of papers per top-level category in our taxonomy, revealing that Input/Prompt Optimization, Architectural Modifications, and Training and Learning Approaches constitute the bulk of current research. This higher-level summary not only validates the comprehensiveness of the taxonomy but also provides researchers with a macro-level understanding of where scholarly attention is concentrated and where new contributions might be most impactful.

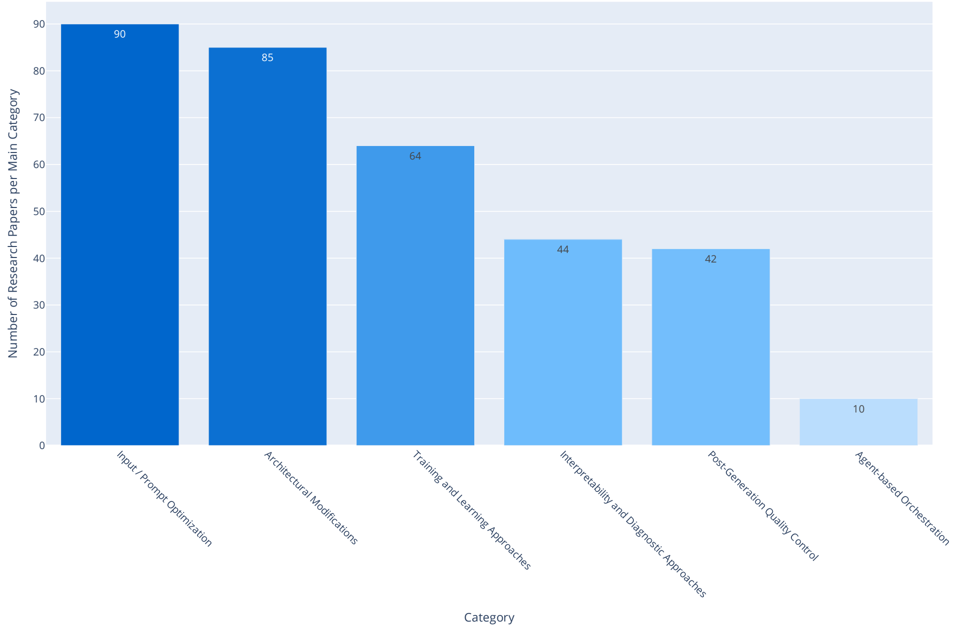


Figure 6. Aggregated number of research papers per top-level category in the proposed taxonomy.

Beyond organizing existing literature, our taxonomy also highlights potential research gaps and emerging directions. By systematically categorizing hallucinations according to their root causes, modalities of manifestation, and domain-specific implications, we can identify areas where scholarly attention is uneven. For example, while subcategories such as Prompt Engineering, Supervised and Semi-supervised Learning, and External Fact-checking dominate the field with the highest number of publications, other areas—including Knowledge Distillation, Response Validation, and Reflexive / Self-reflective Agents—are comparatively underexplored, suggesting opportunities for contributions. Similarly, the temporal distribution of publications shows a rapid surge since 2022, with 2023 marking a peak, indicating both a growing research interest and an evolving methodological landscape. Furthermore, our taxonomy reveals cross-cutting themes that transcend individual architectures or domains, positioning it not only as a synthesis of current knowledge but also as a roadmap for future innovation in hallucination mitigation.

5. Methods for Mitigating Hallucinations

5.1. Training and Learning Approaches

Training and Learning Approaches encompass a diverse set of methodologies employed to train and refine AI models. We have included foundational approaches like Supervised and Semi-supervised Learning, learning via interaction and reward signals in Reinforcement Learning, and differentiating examples using Contrastive Learning. We have also included techniques such as Knowledge Distillation transferring knowledge from larger to smaller models, and instruction tuning, which aligns models to follow natural language instructions.

5.1.1. Supervised and Semi-Supervised Learning

Supervised learning [167,307] and semi-supervised learning [17,68] are foundational approaches which rely on the use of high-quality, curated datasets to guide model training, ensuring that models embody accurate and factual knowledge, and the ability to generalize effectively. For supervised learning, annotated datasets provide the explicit signals needed to align the model's outputs with desired outcomes while semi-supervised learning complements this process by leveraging a combination of labeled and unlabeled data, maximizing the utility of available data while reducing annotation costs. The effectiveness of training depends significantly on the quality of the datasets used, since well-curated examples can help reduce the risk of bias and hallucinations [73,238,281,304,306]. These datasets undergo iterative reviews and validation checks to rectify inconsistencies, while diversity is equally critical, since it promotes model generalization across various scenarios. Based on the methodologies described, we group the papers of this subcategory thus:

- Fine-Tuning with factuality objectives, where techniques such as FactPEGASUS make use of ranked factual summaries for factuality-aware fine-tuning [105] while FAVA generates synthetic training data using a pipeline involving error insertion and post-processing to detect and correct fine-grained hallucinations [112]. Similarly, Faithful Finetuning employs weighted cross-entropy and fact-grounded QA losses to enhance faithfulness and minimize hallucinations [209]. Principle Engraving fine-tunes the base LLaMA model on self-aligned responses that adhere to specific principles [230], while [292] explores how the combination of supervised fine-tuning and RLHF impacts hallucinations. Wasserstein Generative Adversarial Networks (GANs) provide the conceptual basis for [17] which introduces Adversarial Feature Hallucination Networks (AFHN). AFHN synthesizes fake features for new classes by using labeled samples as conditional context. The framework uses a classification regularizer for feature discriminability and an anti-collapse regularizer that boosts the diversity of the synthesized features.
- Synthetic Data & Weak Supervision, where studies automatically generated hallucinated data or weak labels for training. For instance, in [68] hallucinated tags are prepended to the model inputs so that it can learn from annotated examples to control hallucination levels while [81] uses BART and cross-lingual models with synthetic hallucinated datasets for token-level hallucination detection. Similarly, Petite Unsupervised Research and Revision (PURR) involves fine-tuning a compact model on synthetic data comprised of corrupted claims and their denoised versions [235] while TrueTeacher uses labels generated by a teacher LLM to train a student model on factual consistency [311].
- Preference-Based Optimization and Alignment: In [114] a two-stage framework first combines supervised fine-tuning using curated legal QA data and Hard Sample-aware Iterative Direct Preference Optimization (HIPO) to ensure factuality by leveraging signals based on human preferences while in [270] a lightweight classifier is finetuned on contrastive pairs (hallucinated vs. non-hallucinated outputs). Similarly, mFACT—a metric for factual consistency—is derived from training classifiers in different target languages [79], while Contrastive Preference Optimization (CPO) combines a standard negative-log likelihood loss with a contrastive loss to finetune a model on a dataset consisting of triplets (source, hallucinated translation, corrected translation) [206]. UPRISE employs a retriever model that is trained using signals from an LLM to select optimal prompts for zero-shot tasks, allowing the retriever to directly internalize alignment signals from the LLM [322]. Finally, behavioral tuning uses label data (dialogue history, knowledge sources, and corresponding responses) to improve alignment [84].
- Knowledge-Enhanced Adaptation: Techniques like HALO injects Wikidata entity triplets or summaries via fine-tuning [140] while Joint Entity and Summary Generation employs a pre-trained Longformer model which is finetuned on the PubMed dataset, in order to mitigate hallucinations by supervised adaptation and data filtering [134]. The impact of injecting new knowledge in LLMs via supervised finetuning and the potential risk of hallucinations is also studied in [89].
- Hallucination Detection Classifiers: [142] involves fine-tuning a LLaMA-2-7B model to classify hallucination-prone queries using labeled data while in [129] a sample selection strategy

improves the efficiency of supervised fine-tuning by reducing annotation costs while preserving factuality through supervision.

Beyond the aforementioned categories, Supervised and Semi-supervised Learning has successfully been employed as an enabling method, complementing other hallucination mitigation approaches such as Contrastive Learning, Internal State Probing, Retrieval Augmented Generation and Prompt Engineering. For instance, it has been used to train models to optimize the aggregation of answers from manually annotated data [10], to train factuality classifiers [33,138,256,296], for the creation of synthetic datasets [111,241], and finally, as a component of refinement pipelines as shown in [71,108,148,211,302,307,313,315,316,327,328]. More specifically:

- Training of factuality classifiers: Supervised finetuning is used to train models on labeled text data in datasets such as HDMBENCH, TruthfulQA, and multilingual datasets demonstrating improvements in task-specific performance and factual alignment [33,138,211]. Additionally, training enables classifiers to detect properties such as honesty and lies within intermediate representations resulting in increased accuracy and separability of these concepts as shown in [148,256,296].
- Synthetic data creation: In the Fine-Grained Process Reward Model (FG-PRM), various hallucination types are injected into correct solutions of reasoning steps. The synthetic dataset thus created is used to train six Process Reward Models, each able to detect and mitigate a specific hallucination type [111] while techniques such as RAGTruth includes human-annotated labels indicating whether generated responses are grounded in retrieved content, which enables supervised training and evaluation of hallucination detection models [241]. Similarly, [97] addresses over-reliance on parametric knowledge by introducing an entity-based substitution framework that generates conflicting QA instances by replacing named entities in the context and answer.
- Refining pipelines: Supervised training is used to train a critic model using the base LLM's training data and synthetic negatives [71]. TOPICPREFIX is an augmentation technique that prepends topic entities from Wikipedia to improve contextual grounding and factuality [108] while the training of the Hypothesis Verification Model (HVM) on the FATE dataset aims to help the model recognize faithful and unfaithful text [302], while similar approaches discern between truthful and untruthful representations [313,315,316]. Self-training is used to train models on synthetic data with superior results compared to crowdsourced data [327] and finally, in WizardLM a LLaMA model is finetuned on generated instructions, thus resulting in better generalization [328].

5.1.2. Reinforcement Learning

Reinforcement Learning (RL) (Figure 7) is a fundamental technique where models are trained to perform desired actions through the optimization of reward signals [254]. Unlike supervised learning, RL does not require labeled data, as it learns strategies through trial-and-error by maximizing cumulative rewards based on a well-defined reward function [254]. For example, this reward function is computed in [107] using natural language inference (NLI) models that assess whether each sentence in the summary is entailed by the source document, with summaries that are more entailed leading to higher rewards. Maximizing the expected textual reward produces summaries that are fluent, grounded in the source and have less hallucinations [107]. Another effective approach fine-tunes LLMs using factuality-based reward signals such as FactScore (a reference-based metric that checks atomic claims against Wikipedia) and an additional score derived from the model's calibrated confidence over multiple samples as shown in [113]. Furthermore, research has shown that not only the generating component but also the reward modelling component can be improved using techniques such as process supervision where models are designed to evaluate and rank multi-step reasoning chains, thus improving the fidelity of reward signals and providing the training backbone for models like InstructGPT [178].

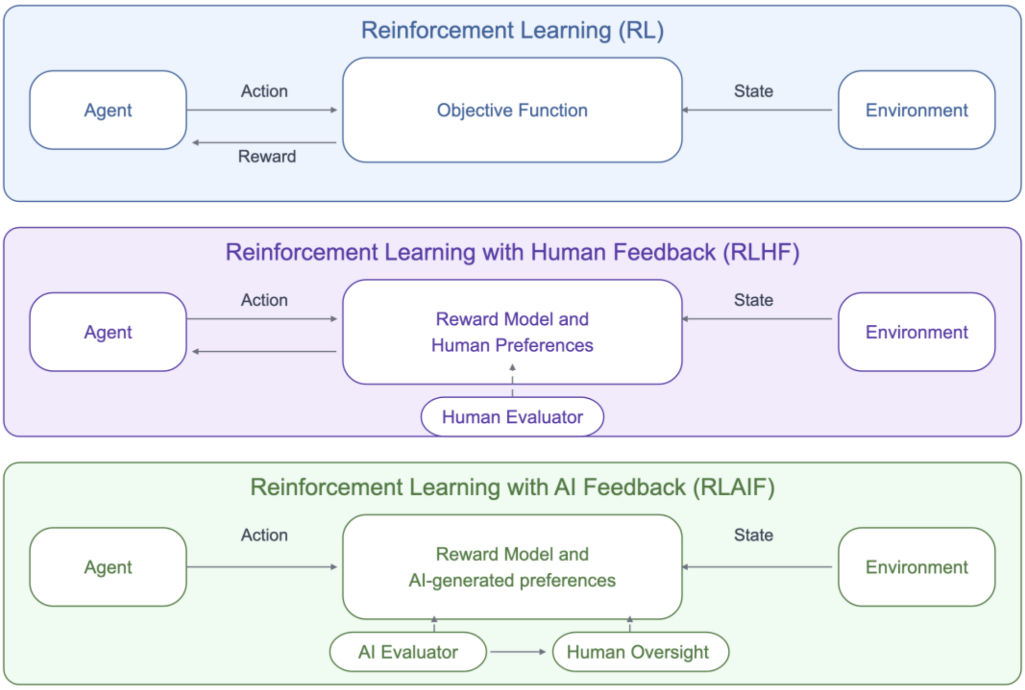

Reinforcement Learning (RL)
Agent
Action
Reward
Objective Function
State
Environment
Reinforcement Learning with Human Feedback (RLHF)
Agent
Action
Reward Model and Human Preferences
State
Environment
Human Evaluator
Reinforcement Learning with AI Feedback (RLAIF)
Agent
Action
Reward Model and AI-generated preferences
State
Environment
AI Evaluator
Human Oversight

Figure 7. Overview of reinforcement learning for hallucination mitigation, including RLHF and RLAIF.

Defining precise and scalable reward functions however, can be challenging with standard RL. This difficulty in directly specifying desirable outputs or defining a reward function led to the development of Reinforcement Learning from Human Feedback (RLHF) [74,307]. In RLHF, human evaluators refine the model's outputs through feedback, thereby enabling it to align with specific styles, constraints or user preferences. This process is particularly valuable for reducing hallucinations, as this feedback encourages the model not only to improve accuracy but also to recognize when it should acknowledge uncertainty rather than generating potentially false information [74,307]. The reward model subsequently provides the optimization signal for the language model, allowing it to better reflect human preferences, thus improving factuality, coherence, and adherence to ethical considerations [20,63,122,307]. For instance, Anthropic's Constitutional AI demonstrated an 85% reduction in harmful hallucinations [63] while OpenAI’s GPT-4 improves by 19% over GPT-3.5 on internal adversarial factuality benchmarks and shows large gains on datasets like TruthfulQA [122]. Recent research introduces reward-model-guided fine-tuning, where a self-evaluation signal encourages the model to acknowledge its limitations for queries outside its parametric knowledge. The approach leverages value alignment to optimize the model’s behavior, converting potential hallucinations into abstentions [23].

However, it's crucial to note that while RLHF is highly effective, its application is not always straightforward. Specifically, research has shown that as models underwent more RL training, certain types of context-dependent sycophancy increased, indicating that these models prioritize agreement with user beliefs over factual accuracy [305]. Additionally, the reliance on human feedback can be resource-intensive, time-consuming, and prone to personal biases and cultural biases as well as challenging to scale broadly [63,304], while inverse scaling issues might lead LLMs to express stronger political view or even refuse to shut down [85]. These issues have been addressed with Reinforcement Learning from AI Feedback (RLAIF) which extends the RL paradigm by employing AI systems to replace or augment human evaluators, thus automating the process and potentially scaling feedback generation, often with superior results to RLHF [63,261]. Specifically, RLAIF automates the feedback process by using another AI model to generate preference signals or critiques, which are then used to train the reward model for the LLM. RLAIF-trained models can match or even exceed RLHF performance across multiple evaluation dimensions, including helpfulness, honesty, and harmlessness, while significantly reducing annotation costs [261]. For instance, RLAIF produced a tenfold reduction in human labor for reward model training while models trained via RLAIF on critiques from GPT-4 outperformed RLHF-trained counterparts on benchmarks like MT-Bench and AlpacaEval [261]. In a similar line of research, Reinforcement Learning with Knowledge Feedback (RLKF) avoids human preference labels in favor of integrating knowledge feedback [175]. RLKF optimizes LLMs using a reward signal derived from "knowledge preference," where factual preferences are obtained via the automated DreamCatcher tool. The factual preference data are used to train a reward model, which then optimizes the LLM by using the PPO algorithm. This process assesses factual consistency and uses retrieved evidence as the reward, thus improving factual accuracy without the need for human intervention [175]. Finally, HaluSearch approaches response generation as a dual-process reasoning system, alternating between fast and slow thinking modes with the help of a dynamic switcher. In slow mode, it employs Monte Carlo Tree Search (MCTS) to explore multiple reasoning paths, scoring each intermediate step with a self-evaluation reward model. The reward signals guide exploration toward paths that are less likely to produce hallucinations, effectively performing step-level self-verification [301].

Both RLHF and RLAIF are part of the broader paradigm of Alignment Learning [20], which addresses the fundamental challenge of ensuring AI systems act in accordance with human values and intentions [20,63,115]. Despite the overhead in data collection, both RL as well as its variants RLHF and RLAIF often yield notable improvements in factual consistency and overall user satisfaction [307] while even more advanced variations, such as hierarchical or multi-agent RL [132], are currently being explored to further refine the alignment process. Furthermore, research has shown that Reinforcement Learning and its varieties, are the enabling paradigm for imbuing multi-agent LLM systems with meta-thinking capabilities (such as self-reflection and adaptive planning), allowing them to learn from interaction and long-term adaptation, thus creating more trustworthy agents [199,253] as we will discuss more analytically in the broader category of Agent-based Orchestration. While RL/RLHF/RLAIF can make use of AI agents that function as learners or decision-makers [122,254,307] we have decided to categorize all of them under Training and Learning Approaches because they fundamentally focus on optimizing a model's behavior through iterative learning. Defining a reward signal, training a policy to maximize it, fine-tuning the model to align with specific goals, as well as incorporating human preferences into the reward signal, collectively emphasize learning dynamics, thus making them quintessential training methodologies. Consequently, we reserve the category Agent-based Orchestration for frameworks that emphasize real-time decision-making, where AI agents use models for real-time interaction. We distinguish between this dual perspective based on whether the emphasis is on how the model learns or how it performs tasks in real-time and categorize it accordingly.

5.1.3. Contrastive Learning

Contrastive learning (Figure 8) is a training paradigm that sharpens a model's decision boundaries when it comes to differentiating between positive examples (factually sound statements) from negative examples (incorrect information). A major benefit of contrastive learning is its ability to use unlabeled or weakly labeled data and to efficiently learn representations from unstructured data as shown in techniques such as SimCLR [8] and Momentum Contrast (MoCo) [212]. Although initially targeting computer vision, these methods were later adapted to help mitigate hallucinations by generating synthetic "negative" examples, thus demonstrating that contrastive learning can significantly reduce the need for large and curated datasets [8,212].

Contrastive methods can improve content quality, such as in [152] where the loss is calculated over ranked candidate summaries and the model is penalized if a low-quality summary is selected over a better one (with higher log-probability). This loss is recombined with the standard MLE loss, thus resulting in the optimization of a combined, contrastive reward loss for factuality-ranked summaries. In contrastive learning, carefully constructed negative samples that closely resemble plausible but incorrect content (known as “hard negatives”) can be particularly beneficial, as they challenge the model to learn subtle distinctions [67]. For example, when training a model to generate summaries, a hard negative might be a summary that is fluent and seemingly correct, but contains a subtle factual error. Therefore, the effectiveness of contrastive learning depends significantly on the quality of these negative samples since treating semantically similar samples as false negatives can bias representation learning, a challenge addressed through debiased contrastive loss as shown in [67]. Although [67] does not directly target hallucinations, its proposed debiased contrastive loss has influenced research in hallucination mitigation that selects hard negatives for training models with stronger factual grounding such as [65] and [206].

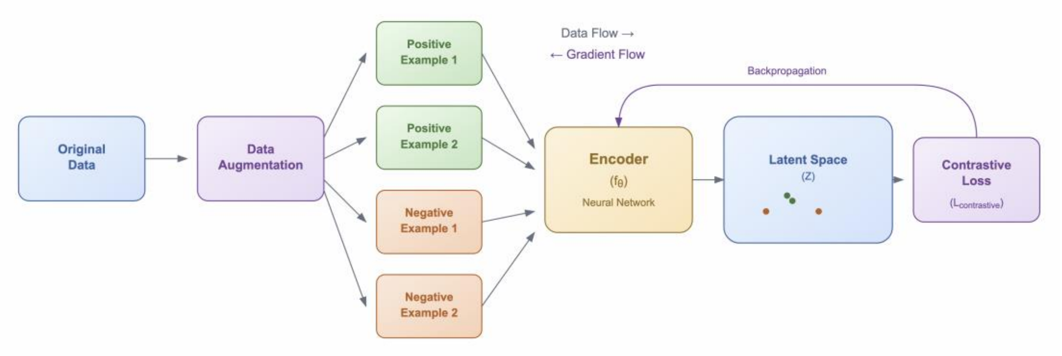


Figure 8. Illustration of the contrastive learning paradigm used to distinguish hallucinated from factual content through hard negative sampling and representation alignment.

Several studies have specifically explored the application of contrastive learning to address the problem of hallucinations in LLMs. For instance, MixCL reduces hallucinations in dialog tasks by training the model to distinguish between appropriate and inappropriate responses [65]. MixCL samples conflicting knowledge (negative sampling) and then employs mixed contrastive learning to reduce the generation probability of negative tokens. While this technique demonstrates scalability and efficiency, the authors also acknowledge the problem of catastrophic forgetting [65]. This issue is addressed in other research studies with techniques such as elastic weight consolidation or experience replay which preserves general knowledge by freezing core parameters while fine-tuning the model only on hallucination-prone tasks [208,327]. Other research studies such as SimCTG introduce a contrastive training objective and decoding method to enhance text generation quality by improving coherence and reducing degenerate outputs and repetitions [2] while Iter-AHMCL employs two guidance models (a positive model trained on low-hallucination data and a negative model trained on hallucinated samples) that provide contrastive signals. These signals are subsequently used to modify the representation layers of an LLM through an iterative contrastive learning process [161]. Other techniques include Contrastive Preference Optimization (CPO) where the standard negative log-likelihood loss is combined with a contrastive loss to explicitly increase the likelihood of preferred (non-hallucinated) translations [206]. Finally, TruthX explicitly allows for the editing of LLM parameters in 'truthful space' using contrastive gradients. By contrasting hallucinated against truthful outputs, TruthX shifts activations toward the latter without having to retrain the model [316]. These methods collectively demonstrate that contrastive learning shows promise for hallucination mitigation, yet its effectiveness isn't uniform as research suggests that strategies that use ground truth versus hallucinated could reduce detection accuracy [128].

While contrastive learning is typically a training approach that refines the representation knowledge of a model, a plethora of other strategies draw on its core intuition of contrasting desirable and undesirable outputs. For example, Decoding by Contrasting Retrieval (DeCoRe) employs a contrastive decoding scheme at inference time, which penalizes token predictions that align with a hallucination-prone version of the model [72]. Similarly, Delta employs a contrastive decoding approach that allows the model to compare output distributions derived from masked versus unmasked inputs and subsequently suppress hallucinated tokens during inference [76]. LLM

Factoscope uses a contrastive framework to distinguish factual from hallucinated outputs by analyzing token distributions across model layers [181]. Self-Consistent Chain-of-Thought Distillation (SCOTT) also draws inspiration from contrastive reasoning during knowledge distillation by filtering rationales from a teacher model and retaining only those that are more plausible when conditioned on correct answers [266]. Decoding by Contrasting Layers (DoLa) extends the contrastive paradigm to internal layer dynamics, contrasting early vs. late-layer activations to suppress hallucinated token distributions and leveraging the finding that factual knowledge often resides in deeper layers, while hallucinations arise from inconsistent intermediate representations [90]. Adversarial Feature Hallucination Networks (AFHN) combine standard adversarial alignment with a dedicated anti-collapse regularizer that maximizes the dissimilarity between hallucinated feature pairs, thus preventing mode-collapsed feature hallucinations [17]. In a different setting, contrastive pairs, comprised of hallucinated and non-hallucinated outputs, are used as labeled supervision for hallucination detection in neural machine translation, though without employing a contrastive loss [319]. Although none of the aforementioned methods except [17] and [319] involve training, they all demonstrate how the contrastive paradigm can enhance generation quality and we will discuss them in the sections "Decoding Strategies", "Internal State Probing", and "Knowledge Distillation", respectively.

5.1.4. Knowledge Distillation

Knowledge distillation (Figure 9) involves training a smaller and more efficient student model to emulate the outputs and learned representations of a larger, more knowledgeable teacher model thereby allowing the student model to benefit from the factual grounding and calibrated reasoning of the teacher [219,280,327]. Knowledge distillation (KD) has emerged as a promising approach to enhance the factual accuracy of LLMs, particularly in tasks susceptible to hallucinations, such as summarization and question answering.

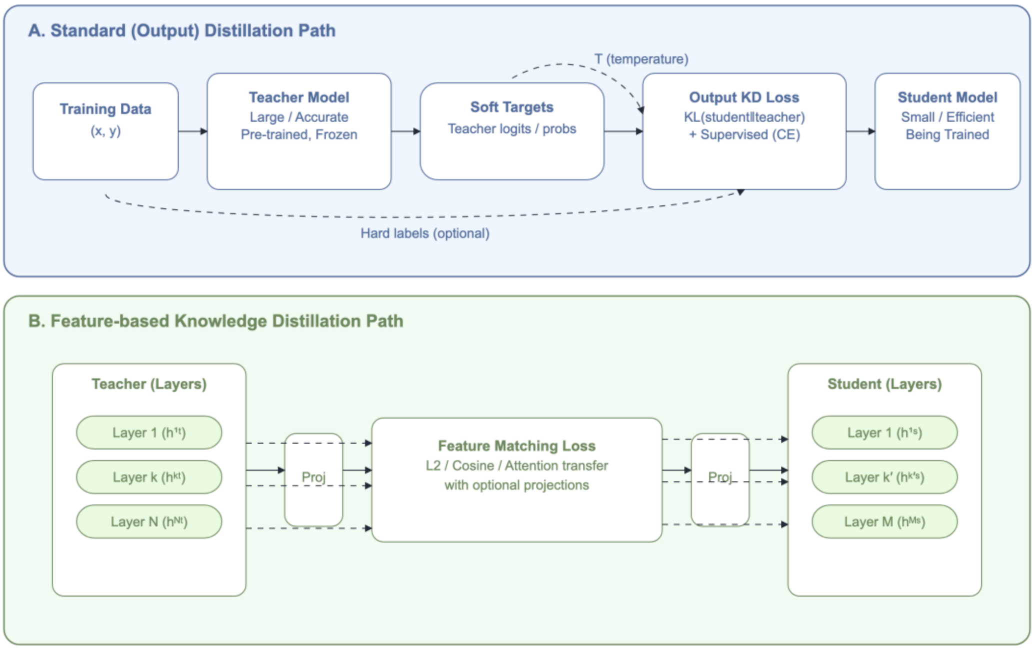


Figure 9. Knowledge distillation framework where a student model learns to emulate a teacher model's outputs to reduce hallucinations and improve factual consistency.

The efficacy of knowledge distillation in reducing hallucinations is demonstrated in research studies such as [219], where transferring knowledge from a high-capacity teacher model to a more compact student model demonstrates significant improvements in exact match accuracy and notable reduction in hallucination rates. In [280], a smoothed knowledge distillation method is proposed where the teacher model provides soft labels rather than deterministic labels significantly mitigating hallucinations, with improved factual consistency across summarization and QA benchmarks, although the authors caution that over-smoothing remains a trade-off potentially sacrificing output diversity for factual consistency [280]. Soft labels can be considered as probability distribution over outputs and they can be used to train the student model to be less overconfident and more grounded in its predictions. Furthering the above-mentioned researches, knowledge distillation is compared to self-training in using synthetic examples in retrieval-augmented QA systems and while both approaches proved beneficial for reducing hallucinations, Knowledge Distillation offered slightly more robust gains in factual accuracy [327]. Chain of Thought (CoT) Distillation and SCOTT independently extend the knowledge distillation paradigm by leveraging the CoT reasoning and self-evaluation capabilities of a larger model to train a smaller and more efficient model so that it can perform multi-step reasoning without needing to generate on long intermediate steps at inference time [201,266]. Similarly, in [258], an LLM generates silver explanations used to fine-tune a Small Language Model (SLM), again via CoT while a subsequent reinforcement learning stage rewards correct reasoning paths, thus promoting generalization. This distillation process is guided by retrieval-enhanced decoding, which helps SLMs adaptively modify their reasoning using the retrieved knowledge rather than merely mimicking LLM outputs [258].

5.1.5. Instruction Tuning

Instruction tuning (Figure 10) is a specialized form of supervised learning that fine-tunes language models on datasets of input-output pairs presented as natural language instructions [178]. Although instruction tuning uses prompts for formatting, it emphasizes training the model to behave differently and not merely prompting it at runtime, which distinguishes it as a method from in-context methods which we will cover in the section “In-context Prompting”. Instruction tuning optimizes the model to follow natural language instructions and be more responsive to carefully crafted prompts, which aim at guiding the model towards more factual and coherent outputs [239,264] or explicitly request factual grounding and discourage speculative responses [114,166,307]. Additionally, Instruction Tuning directly benefits from scaling [92,264] and not only does it enhance task generalization but is pivotal for mitigating hallucinations by aligning outputs with factual instructions [113,114,153] and enabling uncertainty-aware responses [239,271] through mechanisms such as:

- Factual alignment, such as in [114], where LLMs are finetuned using a dataset of legal instructions and responses, explicitly grounding outputs in legal knowledge, thus aligning their behavior to domain-specific prompts and reducing factual hallucinations. Curriculum-based Contrastive Learning-based Cross-lingual Chain-of-Thought (CCL-XCoT) combines curriculum-based cross-lingual contrastive learning with instruction fine-tuning to transfer factual knowledge from high-resource to low-resource languages. Furthermore, its Cross-lingual Chain of Thought (XCoT) strategy guides the model to reason in a high-resource language and then generate in the target language, thereby reducing context-related hallucinations [50].
- Consistency alignment, which is achieved in [153] during a two-stage supervised fine-tuning process: The first step uses instruction–response pairs while in the second step, pairs of semantically similar instructions, which are implemented via contrastive-style learning, are used to enforce aligned responses across instructions.
- Uncertainty awareness, where techniques such as R-Tuning are employed to incorporate uncertainty-aware data partitioning, guiding the model to abstain from answering questions outside its parametric knowledge [239,307].
- Data-centric grounding, where Self-Instruct introduces a scalable, semi-automated method for generating diverse data without human annotation [271]. It begins with a small set of instructions and uses a pre-trained LLM to generate new tasks and corresponding input-output

examples, which are then filtered and used to fine-tune the model, thus generating more aligned and grounded outputs.

Research has shown that instruction tuning can be significantly scaled by curating a high-quality collection of over 60 publicly available datasets covering diverse natural language processing (NLP) tasks [264]. The authors report substantial improvements in zero-shot and few-shot generalization using the Flan-PaLM model while showing that scaling instruction diversity improves factual grounding and reduces hallucinations across unseen tasks by exposing models to more diverse factual contexts [113,264] while CoT reasoning during training further enhances reasoning capabilities [264].

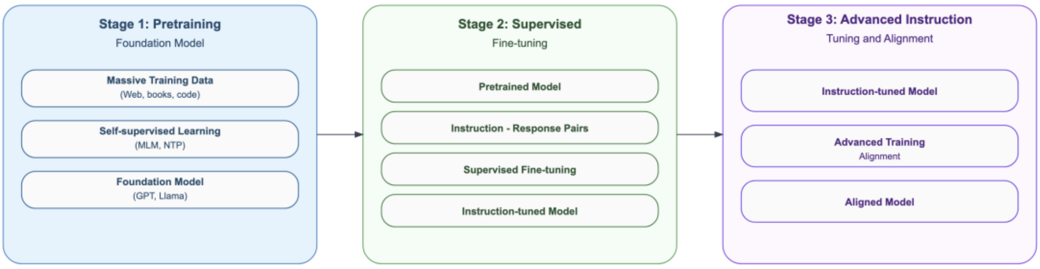


Figure 10. Instruction tuning paradigm showing how models are trained on natural language instructions to improve factual alignment and task generalization.

Despite its benefits, instruction tuning often exacerbates the issue of hallucinations in LLMs since fine-tuning on unseen examples can increase hallucination rates without safeguards [89] and negatively impact the calibration of a model's predictive uncertainty leading to overconfident but hallucinated outputs [314], a finding which is also corroborated by independent observations from OpenAI in the GPT-4 “Technical Report” [122]. To mitigate these risks, techniques explicitly guide a model to say "don't know" or generate refusal responses during the instruction tuning process in order to encourage the model to express uncertainty when appropriate [239,307], or make use of more robust decoding strategies and uncertainty estimation methods [314].

5.2. Architectural Modifications

Architectural modifications encompass structural changes and enhancements made to AI models and their inference processes to improve performance, efficiency, and generation quality. It includes advancements in attention mechanisms and refined decoding strategies. Modifications also involve incorporating external information sources through Retrieval Augmented Generation (RAG). The category further covers integrating different knowledge representation approaches as well as specialized architectural mechanisms for enhanced generation, all aimed at enhancing model capabilities, controllability, and factual accuracy. We clarify that several approaches bridge architectural and post-generation paradigms. In this section, methods are categorized by their primary operational stage: if they primarily modify model internals or inference with structured knowledge, we categorize them under “Architectural Modifications”; if they primarily operate on generated outputs for verification or attribution, we discuss them in “Post-Generation Quality Control”, even if the method involves minor architectural interventions.

5.2.1. Attention Mechanisms

The attention mechanism, first introduced in a groundbreaking paper on neural machine translation [216] and later revolutionized by the Transformer architecture [34], transformed natural language processing by allowing models to weight different parts of the input context dynamically, thereby focusing on the most relevant information at each step of generation. This granular control can directly reduce hallucinations by helping the model pinpoint critical pieces of factual data while research on multi-head attention has shown that heads specialized in factual recall can guide the model to learn multiple relationships in parallel [31].

A similar line of research has shown that inducing hallucinations is fundamentally tied to how the attention weights are altered and that perturbations in the input can influence the attention value maps, effectively steering the model toward specific outputs [184]. The attention mechanism also provides key insights into understanding instruction drift as attention decays over long conversational exchanges as is shown in [193]. Split-softmax addresses this issue by amplifying the model's attention to the system prompt at inference time by reweighting the attention distribution, and thus indirectly addressing hallucinations by framing them as a safety concern related to instruction drift [193]. Inference-Time Intervention (ITI) is another framework that identifies specific attention heads that contain desired knowledge through probing [218]. During the inference process, the activations of these chosen attention heads are directly altered while additional nonlinear probes extract more information from internal states via a multi-token intervention strategy, thus steering the model toward more factually accurate outputs [218]. The attention mechanism in [202] leverages the relational data processed by Graph Neural Networks (GNNs), thereby enabling it to focus on relevant graph parts and improve its contextual understanding. Sliding Generation divides input prompts into overlapping segments, thus implicitly guiding the attention mechanism to focus on smaller, more contextualized windows, while rephrasing verifies the internal consistency of generated content thereby reducing the risk of information loss at the boundaries [206]. Recent research introduces a head architecture that leverages token-to-token attention maps and lookback ratios as primary features to detect unsupported claims, thus allowing LLMs to self-assess their uncertainty through the attention mechanism [4].

Another line of research focuses on directly manipulating tokens. Techniques such as Adaptive Token Fusion (ATF) fuses tokens to alter the weight distribution in the attention mechanism, implicitly conditioning computation to operate over a more efficiently structured token space, thus resulting in a more focused and computationally efficient attention [172]. Truth Forest uses multi-dimensional orthogonal probes to detect “truth representations” in the hidden states, which are then shifted along directions representing maximum factuality, thus biasing the information that the attention mechanism processes. The Random Peek technique extends this intervention across a wider range of sequence positions, thereby ensuring the truthfulness enhancement is pervasive and the model is guided towards generating less hallucinatory outputs [313].

Several techniques also indirectly manipulate attention by conditioning the encoder or decoder during attention computation. For instance, attention patterns and their relationship to factual correctness are analyzed in [35], where the attention mechanism is modified indirectly via a decoding framework while in [72], hallucinations are coaxed through the masking of specific attention heads which specialize in retrieving relative content, effectively introducing a controlled perturbation which conditions the contrastive decoding process. In [9], the authors introduce an natural language understanding (NLU) model that parses meaning-representation pairs (MR pairs) from utterances and reconstructs the correct attribute values in references texts. Specifically, the NLU model consists of a self-attentive encoder and an attentive scorer, where the encoder produces vector representations of slot-value pairs and their paired utterances. The scorer then calculates semantic similarity between each slot-value pair and the utterance to identify the most salient words. Subsequently, the model undergoes self-training in order to recover texts with similar semantics [9]. Research has also shown that language-specific modules can co-ordinate both the encoder and decoder so that only modules corresponding to the intended input and target languages are activated [211]. Finally, attention maps can be used as useful diagnostic signals for factuality as shown in [18], where aggregated attention AggTruth is used to compute contextualized attention patterns across layers, thus detecting hallucinated outputs in RAG scenarios.

5.2.2. Decoding Strategies

In this section, we outline decoding strategies as a set of intervention methods in the generation process of LLMs, which dictate how models translate their learned representations into human-readable text by balancing factors such as fluency, coherence and factual accuracy. LLMs often generate hallucinated text because they struggle to adequately attend to the input context and may rely on outdated or conflicting prior knowledge [312]. Although standard decoding techniques such as greedy decoding, beam search, and nucleus sampling [291] offer distinct trade-offs between diversity, efficiency, and precision [292], models may still commit to incorrect tokens early in the decoding process and then strive to justify them, reflecting inherent issues in the decoding dynamics [147,268]. These strategies are frequently developed alongside other techniques including attention mechanisms [35,72,211], probability-based approaches [66,150,214,256] or RAG-based decoding [255,258] to further improve generation quality and interpretability. Empirical studies demonstrate that advanced decoding strategies can significantly mitigate hallucinations; for instance, Retrieval-Augmented Generation can boost a model's FactScore from 17.5% to 42.1%, and self-correction methods have improved truthfulness scores on benchmarks like TruthfulQA [292]. Building on these findings, we broadly categorize advanced decoding strategies thus:

- Probabilistic Refinement & Confidence-Based Adjustments, where techniques such as Context-aware decoding adjust token selection by prioritizing information aligned with relevant context so as to emphasize contextual information over its internal prior knowledge. This is achieved by amplifying the difference between output probabilities with and without the context, effectively downplaying prior knowledge when more relevant contextual details are available [66], [312]. Another direction involves entropy-based decoding adjustments, where the model’s cross-layer entropy or confidence values are used to penalize hallucination-prone outputs [150] and Conditional Pointwise Mutual Information (CPMI) which adjusts the score of the conditional entropy of the next-token distribution so as to prioritize tokens more aligned with the source [214]. [256] uses logits and their probabilities to refine the standard decoding process by interpreting and manipulating the outputs during generation, while Confident Decoding integrates predictive uncertainty into a beam search variant to reduce hallucinations, with epistemic uncertainty guiding token selection towards greater faithfulness [284]. Similarly, [220] modifies the beam search decoding process to prioritize outputs with lower uncertainty. This method leverages the connection between hallucinations and predictive uncertainty, demonstrating that higher predictive uncertainty correlates with a greater chance of hallucinations [220]. Selective abstention Learning (SEAL) introduces a selective abstention learning framework where an LLM is trained to output a special [REJ] token when predictions conflict with its parametric knowledge. During inference, its abstention-aware decoding leverages the [REJ] probability to penalize uncertain trajectories, thereby guiding generation toward more factual outputs [24]. Finally, Factual-nucleus sampling extends the concept of nucleus sampling [291] by adjusting the sampling randomness during decoding based on the sentence position, thereby significantly reducing factual errors [108].
- Beyond probabilistic refinements, some decoding strategies are inspired by contrastive learning to explicitly counter hallucinations. For instance, Decoding by Contrasting Retrieval Heads (DeCoRe) induces hallucinations through masking specific retrieval heads responsible for extracting contextual information and dynamically contrasting the outputs of the original base LLM and its hallucination-prone counterpart [72]. Delta mitigates hallucinations by randomly masking spans of the input prompt and then contrasting the output distributions generated for both the original and the masked prompts, thus effectively reducing the generation of hallucinated content [76]. Contrastive Decoding is an alternative to search-based decoding methods like nucleus or top-k sampling, which optimizes the difference between the log-likelihoods of an LLM and a SLM by introducing a plausibility constraint that filters out low-probability tokens [64]. Similarly, the Self-Highlighted Hesitation mechanism (SH2) uses contrastive decoding to manipulate the decision-making process at the token level by appending low-confidence tokens to the original context, thus leading the decoder to hesitate before

generation [277]. Spectral Editing of Activations (SEA) projects token representations into directions of maximum information, thus amplifying signals that correlate with factuality while suppressing those linked to hallucinated outputs [283]. Similarly, Induce-then-contrast (ICD) constructs a "factually weak LLM" by inducing hallucinations from the original LLM, via fine-tuning with non-factual samples. These induced hallucinations are subsequently leveraged as a penalty term, thus effectively downplaying untruthful predictions [26]. In Active Layer Contrastive Decoding (ActLCD), a reward-driven classifier uses a reinforcement learning policy to determine when to apply contrastive decoding between selected layers, effectively framing decoding as a Markov decision process [15]. Finally, Self-contrastive Decoding (SCD) reduces the influence of tokens which are over-represented in the model’s training data to directly affect the selection of tokens during text generation, thus reducing knowledge overshadowing [165].

- Verification & Critic-Guided Mechanisms: A number of decoding strategies work in tandem with verification and critic-guided mechanisms to further improve the generation capabilities of the decoder. Critic-driven Decoding combines the probabilistic output of an LLM with a "text critic" classifier which assesses the generated text and steers the decoding process away from the generation of hallucinations [71]. Self-consistency samples from a diverse set of reasoning paths and selects the most consistent answer, leveraging the idea that correct reasoning tends to converge on the same answer. Furthermore, the consistency among the sampled paths can serve as an uncertainty estimate which helps to identify and mitigate potential hallucinations [268]. In Think While Effectively Articulating Knowledge (TWEAK) the generated sequences at each decoding step, along with their potential future sequences, are treated as hypotheses, which are subsequently reranked by an NLI model or a Hypothesis Verification Model (HVM) according to the extent to which they are supported by the input facts [302]. In a similar line of research, mFACT integrates a novel faithfulness metric directly into the decoding process, thereby evaluating each candidate summary regarding its factual consistency with the source document. Candidates that fall below a predetermined mFACT threshold are then pruned, effectively guiding the generation towards more factually accurate outputs [79]. Finally, Reducing Hallucination in Open-domain Dialogues (RHO) generates a set of candidate responses using beam search and re-ranks them based on their factual consistency, which is determined by analyzing various trajectories over knowledge graph sub-graphs extracted from an external knowledge base [260].
- Internal Representation Intervention & Layer Analysis: Understanding how LLMs encode information regarding the possible replies to a query, particularly within their early internal states, is particularly useful for developing decoding strategies that mitigate hallucinations [290]. Intermediate outputs which are prone to hallucinations often exhibit diffuse activation patterns, where activations are spread across multiple competing concepts rather than being focused on relevant references. In-context sharpness metrics, proposed in [154], leverage this observation by enforcing sharper token activations to ensure that predictions are derived from high-confidence knowledge areas. Similarly, Inference-time Intervention (ITI) involves shifting activations during inference and applies these adjustments iteratively until the full response is generated [155] while DoLa leverages differences in logits from earlier vs. later layers promoting the factual knowledge encoded in higher layers as opposed to syntactically plausible but less factual contributions from lower layers [90]. Activation Decoding is another constrained decoding method that directly adjusts token probabilities based on entropy-derived activations without having to retrain the model [154]. Finally, LayerSkip, uses self-speculative decoding and trains models with layer dropout and early exit loss, enabling tokens to be predicted from earlier layers and verified by later layers, thus accelerating inference while mitigating hallucinations [174].
- RAG-based Decoding: RAG-based decoding strategies integrate external knowledge to enhance factual consistency and mitigate hallucinations [255,258]. For instance, REPLUG prepends a different retrieved document for every forward pass of the LLM and averages the probabilities

from these individual passes, thus allowing the model to produce more accurate outputs by synthesizing information from multiple relevant contexts simultaneously [255]. Similarly, Retrieval in Decoder (RID) dynamically adjusts the decoding process based on the outcomes of the retrieval, allowing the model to adapt its generation based on the confidence and relevance of the retrieved information [258].

5.2.3. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) (Figure 11) is a technique that modifies the architecture or inference pipeline by including a retrieval component. Given an initial prompt, this component fetches the most relevant documents from external sources and injects them into the prompt to enrich the prompt's context [259]. This allows LLMs to access information beyond their training data and ground their output in updated references and factual information, thus mitigating hallucinations for a number of varied tasks [39,40,259]. The majority of RAG implementations are pre-LLM approaches (refine-then-generate) where the retrieved documents enrich the prompt prior to reaching the LLM, while in post-LLM approaches (generate-then-refine) an intermediate response is first generated which guides the retrieval step, thus allowing the model to refine its output based on specifically relevant information [208,217,240,243,253]. The retrieval mechanism typically relies on techniques like dual encoders, semantic indexing, or vector similarity [77].

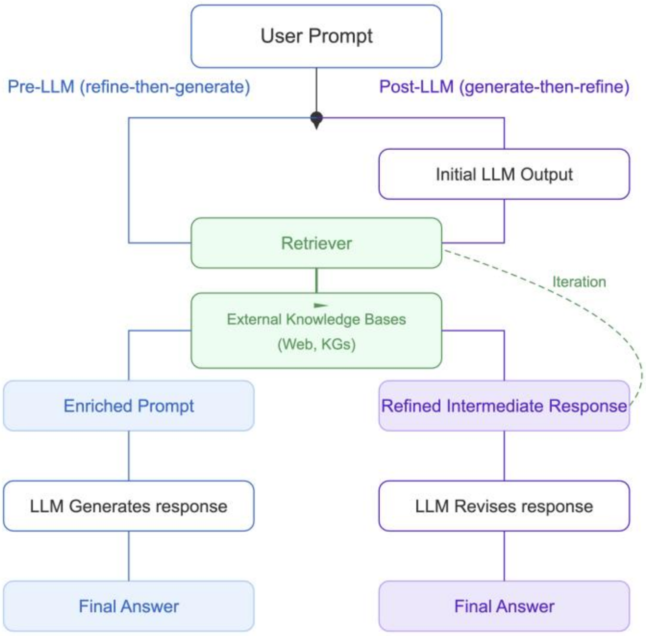


Figure 11. Simplified overview of Pre-LLM and Post-LLM retrieval-augmented generation (RAG) pipelines.

The effectiveness of the retrieval component within RAG systems is the focus of many papers such as [149] in which dense retrieval and sparse retrieval are compared and a new hybrid retrieval is proposed. Specifically, the authors introduce a hybrid retriever which combines the results of sparse and dense retrievers as well as sophisticated query expansion techniques and then uses weighted Reciprocal Rank Fusion to achieve superior relevance scores on the HaluBench dataset. Similarly, in HaluEval-Wild, dense and sparse retrievers are used to construct reference documents from the web, which are subsequently used to evaluate whether a model's output contains hallucinations [142]. Furthermore, in HalluciNot, RAG is used not for generation or evaluation, but as a retrieval mechanism to support hallucination detection. Specifically, HalluciNot retrieves documents from external knowledge sources, trains a classifier with these retrieved documents and finally verifies whether each generated span is supported by factual evidence [138].

While RAG has shown promise in mitigating hallucinations through external source grounding, research has shown that, paradoxically, hallucination rates may increase for a number of reasons:

- Dependency on Retrieval Quality: RAG-based methods rely heavily on the quality of retrieved documents and so inaccurate, outdated, noisy, or irrelevant retrievals may mislead the language model or introduce additional hallucinations [28,58,86,117,125,131,142,190,203]
- Complex Reasoning: Retrieval systems may often strive to engage in complex or multi-hop reasoning thus harming performance of natural language inference (NLI) models [28,117,131].
- Latency and Scalability: Lengthy prompts and iterative prompting improve factuality but at the cost of increased inference time and resource consumption, reducing scalability in real-world settings [58,86,131].
- Over-reliance on External Tools: Some systems require dependency on entity linkers, search engines, or fact-checkers, which can introduce errors or inconsistencies [10,117,131].
- Faithfulness and effective integration: Methods that prioritize factuality sometimes harm the fluency or coherence of generated text, leading to less natural outputs [131] while integrating it effectively into the generation process is another important factor [28].

To address these limitations, several enhanced RAG-based approaches have been proposed that aim to improve grounding, correction, and integration of retrieved evidence. LLM-AUGMENTER retrieves evidence from external sources, consolidates it using modules like entity linking and evidence chaining, and feeds the structured evidence into the LLM to ground its responses. Additionally, it iteratively improves outputs by generating feedback on factual inconsistencies and revising prompts until the response aligns with the retrieved knowledge [58]. Corrective Retrieval-Augmented Generation (CRAG) is another method that addresses hallucinations by leveraging Wikipedia and the Natural Questions (NQ) dataset, paired with passages retrieved via a BM25 retriever. The retrieval process incorporates both the original user query and segments of the LLM's output to identify relevant supporting content. This evidence is then fed into a correction model, which revises hallucinated or unsupported claims to produce more accurate responses [86]. In SELF-RAG is a self-reflective RAG framework where a critic model produces reflection tokens and a generator model outputs these tokens, thus enabling adaptive retrieval, self-evaluation, and critique-guided decoding [272]. Given that the parametric knowledge of LLMs is often outdated, FreshLLMs use search engine augmentation to detect and correct factual errors using retrieved evidence via Google Search, thereby combating factual inconsistencies and indirectly mitigating hallucinations [117] while in [258] the Mistral model is modified to integrate a retrieval module that dynamically fetches documents from a preprocessed and indexed Wikipedia corpus. The architecture involves query encoding, top-k document retrieval, and generation of a fact-grounded response. Finally, AutoRAG-LoRA uses Low Rank Adaptation (LoRA) adapters and KL-regularized feedback correction to mitigate hallucinations. Specifically, it integrates prompt rewriting, hybrid retrieval, and hallucination detection via both classifiers and self-evaluation to assign confidence scores and trigger selective adapter activation. When a hallucination is detected, a KL-divergence and contrastive loss loop fine-tunes only the adapters, thus steering generation toward less hallucinated outputs [38].

The success of RAG-based methods in mitigating hallucinations is partly predicated on the assumption that the retrieved content from the external sources is noise-free and of high quality, although this assumption does not always hold. Noisy retrieval is addressed in [190] via a two-stage approach which includes selectively filtering irrelevant documents using a trained classifier, followed by contextual fine-tuning which conditions the model to ignore misleading inputs. Beyond the issue of noisy retrieval, recent studies suggest that the timing of retrieval in RAG-based systems is just as critical for mitigating hallucinations, since unnecessary retrieval risks introducing irrelevant information while increasing both inference time and computational cost [203]. To address this issue, the authors propose Dynamic Retrieval Augmentation based on Hallucination Detection (DRAD), which adjusts the retrieval process in real time by incorporating a real-time hallucination detection (RHD) module and a self-correction mechanism using external knowledge (SEK). Probabilistic Tree-of-thought Reasoning (ProbTree) is another framework that combines retrieved knowledge (open-book QA) with parametric knowledge (closed-book QA) while addressing common issues in retrieval-augmented reasoning through a tree-based hierarchical reasoning structure and uncertainty estimation [231]. Recent research has improved the quality of chunking the indexed documents by using a semantic chunker during extraction, thereby ensuring that only meaningful and contextually relevant information is integrated for the generation process [93].

In addition to addressing the challenges of noisy retrieval, recent research has explored a variety of new frameworks to optimize RAG-based systems. For instance, the Read-Before-Generate (RBG) framework combines LLM with a reader module that assigns evidence probability scores to individual sentences, guiding the generation process toward factually grounded content. During inference, multiple retrieved documents are fused as input, while a specialized pre-training task encourages reliance on external evidence when producing answers [244]. REPLUG retrieves relevant documents from an external corpus via a lightweight, independent module, prepends them to the input prompt and uses the same LLM to encode the retrieved document [255]. Neural-retrieval-in-the-loop queries external knowledge sources based on complex multi-turn dialogue context. The authors enhance their approach with RAG-Turn, which allows for the synthesis of information across multiple documents, ensuring that the retrieved content remains relevant to each particular dialogue turn, thus preventing incorrect focus on single, potentially irrelevant topics, and indirectly helping in reducing factual inaccuracies [257]. Retrieval in Decoder (RID) performs retrieval and generation simultaneously since it integrates the retrieval directly into the decoding process, thus conditioning decoding granularity on the outcome of the retriever. Subsequently, decoding can be corrected as tokens are generated, preventing the accumulation of errors that can lead to factual inaccuracies [258]. Finally, Substring Extraction-based RAG (SEBRAG) leverages an extraction tool to retrieve a specific text span conditioned on three values: the document identifier, the position of the first word, and the position of the last word, which together define the exact substring to be returned. By targeting span responses and explicit abstention when no answer exists, SEBRAG reduces hallucinations in safety-critical QA tasks like aircraft maintenance [11].

RAG-based methods have been used in dataset creation and code generation. For instance, Analytical Annotation of Hallucinations (ANAH) is a dataset for analytical annotation of hallucinations in LLMs. A key step in this annotation process involves the retrieval of a reference fragment for each answer sentence. The retrieved reference is then used to judge the hallucination type and to correct hallucinated content, emphasizing the importance of these reference documents [28]. In [10], introduce ChitChatQA, a dataset that is used to assess the capabilities of LLMs to answer complex and diverse questions. For its construction, the authors developed a framework that invokes tools and integrates retrieved information from reliable external sources into its reasoning process. Finally, research has proven that RAG-based approaches can also help mitigate hallucinations in code generation. For instance, in [182], a specific retrieval library functioning as code repository is consulted for each code generation task. Relevant code snippets are retrieved based on similarity to the task description and used as prompts, resulting in consistent performance improvements across all evaluated LLMs.

In addition to the aforementioned approaches, an increasing number of studies apply RAG to retrieve specific information from Knowledge Graphs, so as to enrich the initial prompt's context-specificity. For instance, the Recaller module in [59] retrieves entities from a knowledge graph to guide generation through a two-stage highlighting mechanism: a coarse filter identifies potential hallucinations, and a fine-grained step masks and regenerates them. Graph-RAG [125] is comprised of HybridRAG which injects graph-related entities or relations into the final context, based on semantic proximity or relation paths, and FactRAG which evaluates the factual consistency of the generated text and filters hallucinated responses. Graph-based detection techniques can also be leveraged to extract structured triplets from LLM outputs. Following extraction, explicit relational reasoning is used to compare the extracted triplets against a reference knowledge graph to identify unsupported or fabricated facts [179].

While most RAG methods retrieve knowledge before generation, retrieval post-generation can also be used to validate and refine existing outputs as is shown in [208] where a post-hoc RAG pipeline explicitly extracts factual claims from LLM outputs and verifies them against knowledge graphs. Similarly, Retrofit Attribution using Research and Revision (RARR) implements a post-hoc correction method that focuses on correcting hallucinations after generation using external evidence by identifying unsupported claims, finding better supporting evidence, and revising the text accordingly [243]. Neural Path Hunter focuses on reducing hallucination in dialogue systems by grounding generated responses to known facts from a knowledge graph and while it does not use a typical RAG architecture, it traverses the knowledge graph, constructs a KG-entity memory and uses embedding-based retrieval during refinement [217]. These applications highlight the strong synergy between retrieval mechanisms and structured knowledge representations—particularly knowledge graphs—which not only support retrieval but also enable richer reasoning and verification. The next section explores Knowledge Representation Approaches in more detail, with a focus on their role in mitigating hallucinations.

5.2.4. Knowledge Representation Approaches

Knowledge representation approaches (Figure 12) incorporate structured and semi-structured knowledge, such as knowledge graphs [47,59,179,217], ontologies [144], or entity triplets into a language model's architecture to enhance reasoning and factual consistency [163]. These encoded knowledge representations emphasize explicit connections, such as hierarchical classifications and causal links, thereby enabling the verification of statements against known relationships. To ensure that the integration of such knowledge representations provides satisfactory results, it is necessary to maintain high-quality and up-to-date content, which is readily available and free from ambiguities [59,163,190].

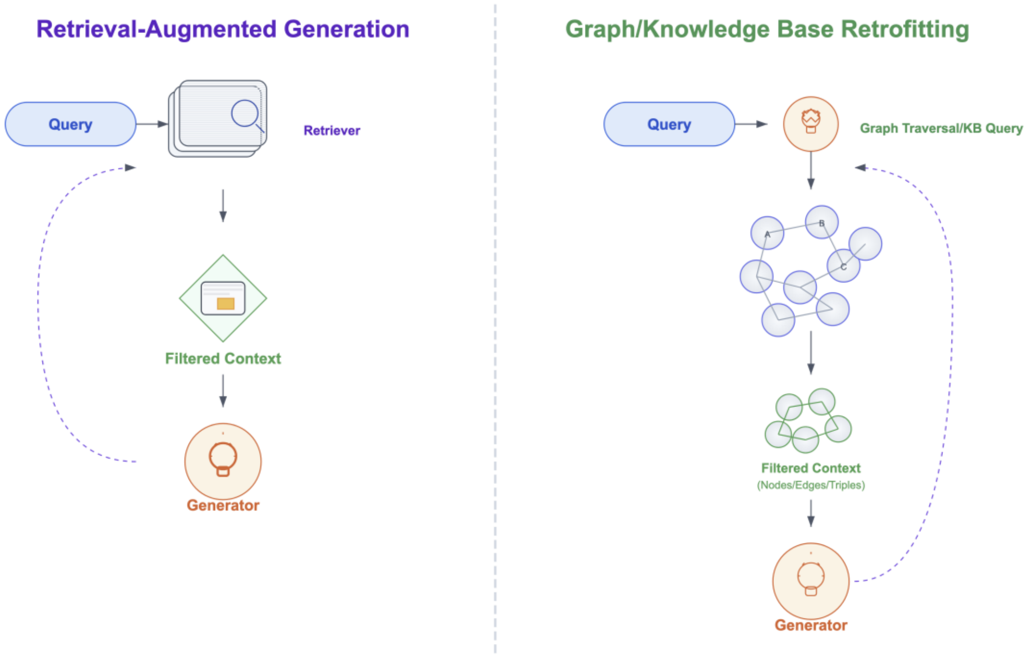


Figure 12. Graph / Knowledge Base injection vs. RAG diagram, effectively showing how these approaches process queries differently - RAG through document retrieval and filtering, while Knowledge Base systems traverse graph structures to extract relevant context.

Integration methods vary widely ranging from prompting during inference [297], to retrofitting external knowledge [59,125,208] and from modifying the attention mechanism [202] to directly injecting graph information into latent space [163,179]. For instance, the Recaller module retrieves entities and their one-hop neighbors from a knowledge graph and uses the retrieved knowledge to guide generation in two stages [59]. The first stage encompasses a coarse filter for the identification of potentially hallucinated spans, while the second step selectively masks these spans before regeneration, allowing the model to ground its output more precisely in retrieved knowledge. This intervention significantly improves factual accuracy while maintaining fluency though the authors caution that retrievers may lack sufficient relational depth for complex queries [59], which is an issue that is also independently corroborated in [190]. FLEEK follows a similar approach by extracting factual claims from text, converts them into a triplet format, and then generates questions for each triplet. These questions are submitted to a question answering system based on knowledge graphs to retrieve factual answers and to verify the correctness of the extracted facts [116].

Knowledge graphs also serve as external sources that enrich the context before generation as shown in [47,125,144,163,240]. Graph-integration techniques may include entity-relation extraction, subgraph extraction, or multi-hop reasoning through graph traversal [124,125]. These graphs are often flattened or linearized into hierarchical textual prompts or encoded representations to fit the input format expected by LLMs [124]. Specifically, in [125], a modular retrieval stage uses entity linking, subgraph construction, or semantic path expansion based on knowledge graphs to expand or re-rank retrieved passages based on graph-related signals like semantic proximity or relation paths, thereby improving factual grounding prior to decoding [125]. Similarly, in [144], DBpedia is used as a source to explore whether the internal knowledge of LLMs can substitute knowledge graphs. The authors conclude that, although LLMs encode factual information within their parametric representations, this implicit knowledge is limited and explicit, external knowledge still remains necessary [144]. Knowledge Graph Retrofitting is another framework that focuses on extracting only the important information that needs to be validated from intermediate responses and then retrofits the retrieved information from knowledge graphs to these responses, thus improving the model's reasoning process [208]. LinkQ is an open source NLI interface that directs the LLM to query Wikidata KG in order to ground its responses on factual data. The authors report that LinkQ outperforms GPT-4 in several scenarios although it struggles with complex questions [210]. ALIGNed-LLM projects entity embeddings from a pre-trained KGE model into the embedding space using a lightweight trainable projection layer. These aligned embeddings are concatenated with textual token embeddings, allowing the model to distinguish similar entities, reduce ambiguity, and generate responses that are structurally grounded in the KG [22]. Finally, G-retriever uses soft-prompting and addresses the context-length issue by formulating retrieval as a Prize-Collecting Steiner Tree optimization problem, where the goal is to extract a subgraph that balances the relevance of included nodes with the cost of their connections, while allowing less relevant nodes to be omitted. In this way, G-retriever optimizes both retrieval quality and input efficiency under limited context length [119].

A closely related line of research focuses on embedding graph-related information directly into the latent space of a model. For instance, Graph Attention Networks (GAT) and message passing mechanisms are leveraged in order to explore semantic relations between hallucinations in the latent space. This exploration is premised on the fact that hallucinations are not unstructured and on the principle of homophily, according to which entities that share similar characteristics are more likely to form connections with each other [179]. Graph Neural Networks (GNN) combat hallucinations by being effectively integrated to the attention mechanism, resulting in a hybrid model architecture that aggregates information from neighboring nodes to improve contextual understanding. [202]. Finally, Rho leverages the representation of linked entities and relation predicates to improve the dialog capabilities of models in open-domain dialogs. Specifically, the authors ground the parametric knowledge to retrieved embeddings (local knowledge) and also equip the attention mechanism with multi-hop reasoning (global knowledge), thus enabling the model to produce less hallucinated content [260].

5.2.5. Specialized Architectural Mechanisms for Enhanced Generation

This category encompasses advanced frameworks that we have found difficult to classify in the previous sections of "Architectural Modifications". A number of techniques—including diffusion-inspired mechanisms, Mixture-of-Experts frameworks [39], [247], dual-model architectures [95], LoRA adapter integration and specialized classifiers [39,138], token pruning and fusion [160,172], and Representation Engineering [256], a novel paradigm for training neural networks—have demonstrated significant potential in addressing hallucinations.

A Mixture of Experts (MoE) architecture, first introduced in [16] and later scaled for LLMs [224], partitions a model into expert modules, each specializing in particular task. A gating network dynamically selects a few relevant experts per input, enabling sparse activation for computational efficiency. MoE architectures help mitigate hallucinations by localizing knowledge and reducing interference between unrelated data domains although one of their disadvantages is that they increase parameter count. This concept is extended in [39], where Lamini-1 implements a Mixture of Memory Experts (MoME) architecture with millions of expert modules functioning as a structured factual memory. These modules allow explicit storage and retrieval of factual information during inference directly and internally via sparse expert routing, thus reducing reliance on internal weights while the combination of cross-attention routing, LoRA-style adapters, and accelerated kernels [39]. Similarly, the MoE architecture in [247] uses a number of specialized experts which are orchestrated via a comprehensive mechanism for input processing, dynamic expert selection, control of information flow based on decision-routing and majority voting, to effectively filter any erroneous responses while reducing computational overhead.

In addition to MoE architectures, dual-model architectures have also been used to effectively combat hallucinations as demonstrated in [95] where the primary model generates the initial text based on the input prompt, the secondary model monitors and analyzes the output from the primary model in real-time, checking for logical and factual inaccuracies while a feedback loop mechanism further enhances the self-monitoring process by allowing for iterative improvements based on detected errors. This interplay allows the system to cross-reference its outputs, thus improving the reliability of the generated text since any inconsistencies are immediately addressed [95].

A multi-task architecture designed with specialized components specifically for detecting various types of hallucinations is introduced in [138]. Hallucination Detection Method 2 (HDM-2) selects a pre-trained LLM as a backbone and adds distinct detectors such as, for instance, a classification head for common-knowledge hallucination detection. The individual components of the framework are trained LoRA adapters while shallow classifiers are used for context-based hallucination detection. These components allow the system to be tailored for specific types of factual verification [138]. Similar components are introduced in the RBG framework, which introduces an end-to-end architecture that jointly models answer generation and machine reading. This involves a dedicated reader module that produces evidence probability scores, which are then explicitly integrated into the final distribution prediction of the generator. The framework also includes a pre-training task for factual grounding, which encourages the model to rely more heavily on retrieved documents, aiming at enhancing the factual accuracy of the generated text [244]. is a Rank-adaptive LoRA fine-tuning (RaLFiT) is a method that focuses on parameter updates on the most truthfulness-relevant modules of an LLM. It first conducts probing to measure the truthfulness correlation of each transformer module and then allocates higher LoRA ranks to modules with stronger correlations. The model is fine-tuned using Direct Preference Optimization (DPO) on paired truthful and untruthful responses to enhance alignment with factual outputs [25].

In addition to architectural modifications and dual-model architectures, token management techniques such as pruning and fusion have also been used to target the mitigation of hallucinations. Pruning is a compression technique that modifies the architecture of a model by removing redundant weights, thereby forcing it to rely more heavily on the source document while reducing the chance of generating unsupported content as shown in [160] where a number of pruning methods (layer-wise magnitude, SparseGPT, Wanda) have been tested. Pruned models exhibit increased reliance on the source document, a higher lexical overlap between the generated summary and the original source material and less hallucinated content regardless of the pruning method employed although in some cases contextual nuance might be sacrificed [160]. Similarly, Adaptive Token Fusion (ATF) dynamically alters token processing during inference by adding an intermediate layer to the LLM's token processing pipeline, specifically placed after tokenization and embedding steps, thus directly contributing to enhanced generation. ATF selectively fuses redundant tokens based on contextual similarity, while prioritizing only contextually relevant tokens to be considered during inference. This semantic condensation decreases the likelihood of the model generating hallucinations while leading to fast inference times and reduced computational load [172].

Spectral Editing of Activations (SEA) is a training-free approach that uses spectral decomposition to edit the internal activations of a model, thus steering its behavior towards generating more desirable outputs. More specifically, SEA projects the model's input representations into directions that maximize covariance with "positive demonstrations" (e.g., truthful content) while minimizing covariance with "negative demonstrations" (e.g., hallucinated content). By decorrelating activations from "hallucinated" examples, SEA directly enhances the factual consistency and reliability of the generated content [283]. Representation Engineering (RepE) is a novel technique that focuses on representations as opposed to individual neurons as the primary unit of understanding, which significantly boosts model interpretability. The ability to identify how representations of concepts are preserved or altered across layers provides a direct path for controlling the generation of cognitive phenomena like honesty and lies while reducing hallucinations. RepE also uses Linear Artificial Tomography (LAT), which is essentially a diagnostic tool that enables targeted improvements in generative quality by aligning representations with desired behaviors, such as truthfulness or honesty [256]. Finally, TruthX is an inference-time intervention method that uses an auto-encoder to map internal representations into semantic and truthful latent spaces and contrastive learning to identify specific "truthful editing directions". During generation, it intervenes to re-orient the internal representations towards these directions, which effectively increases the truthfulness of the generated output and reduces the occurrence of hallucinations [316].

Finally, inspired by the denoising diffusion paradigm, recent research has explored architectural modifications collectively framed as latent diffusion, including improved attention mechanisms, hierarchical processing layers, and enhanced control over the propagation of latent representations within the model. For example, in [94], a modified Mistral model that incorporates such latent diffusion components shows reduced hallucination rates, improved contextual coherence, and greater predictive accuracy.

5.3. Input / Prompt Optimization

Input/Prompt Optimization refers to strategies for carefully crafting and refining the text provided to AI models to steer their behavior and output, often specifically to mitigate hallucinations. This includes comprehensive Prompt Engineering techniques like Structured or Iterative Reasoning Prompting (e.g., Chain of Thought) which can expose potential flaws in logic leading to hallucinations, and In-context Prompting using carefully synthesized prompts in order to guide the model away from hallucinations. It also involves Context Optimization where techniques such as RAG and meta-prompting optimize prompts and System Prompt Design to specifically synthesize persistent prompts that guide the generation process and alignment.

5.3.1. Prompt Engineering

Prompt engineering involves the strategic design of input prompts to constrain the generative space, leveraging LLMs' sensitivity to contextual cues [3,169,232]. This section focuses on general-purpose prompting strategies and their role in hallucination mitigation while other specialized forms of prompt engineering—such as in-context prompting, system prompt design, and meta-prompting—are discussed in their respective dedicated sections. Prompt engineering operates on the premise that LLMs are highly sensitive to suggestions or contextual cues: even subtle prompt modifications can narrow the generative space and shift the model’s output distribution [3], leading to marked changes in performance [169] and reducing the likelihood of hallucinations [232]. Prompts may take the form of explicit instructions or contextual information, with techniques ranging from simply rephrasing questions to introducing constraints or requesting the model to verify information or cite sources [308]. Prompt design can strongly influence model behavior, particularly in zero-shot or few-shot scenarios [51,53] while its effects on LLM outputs remain an active research area in NLP [5,56,100]. Even in studies focused on architectural modifications, such as pruning, researchers often test multiple prompt templates to control for prompt sensitivity and isolate causal effects on hallucination rates [160]. Nonetheless, prompt engineering remains a somewhat ad hoc process, requiring experimentation and domain expertise to optimize outcomes [326]. This has led researchers to explore systematic methods, such as algorithmic prompt search and generation [322], and to organize reusable prompting strategies into structured pattern catalogs [5].

In guiding content generation and self-correction, prompt engineering can also be used to instruct LLMs such as ChatGPT to create, filter, and vary hallucinated questions based on specific constraints, explicitly guiding the model's generation behavior, style and factual correctness across different sampling strategies [232]. Despite the detailed prompt design, the study reveals the key limitation of prompt engineering to overcome the model's tendency to default to familiar phrasings as a result of its entrenched patterns [232], a conclusion which is also supported by findings in [1,5,143]. In [145] the authors elicit candidate generations from language models to probe their internal beliefs using self-reflection prompts. By structuring prompts to generate statements first and then reflect on their truthfulness, the study shows how prompt decomposition can surface internal contradictions that may be indicative of hallucinations. However, the authors highlight a key limitation: self-reflection prompts can be unreliable, as models often evaluate their own outputs inconsistently or overestimate their correctness [145]. Their findings highlight that prompt-based introspection does not necessarily guarantee faithful self-evaluation, as is also shown in [95,201,302].

Prompt Engineering has been successfully used to enhance factual extraction and recall. In studies such as [14], research targets clinical information extraction from unstructured clinical notes using various levels of prompt detail. The authors demonstrate that prompts, carefully crafted with exhaustive researcher-generated search terms, achieved near-perfect agreement with a traditional string-search baseline. Although the results demonstrate the efficacy of carefully optimized prompts, the study also acknowledges a key limitation since detailed prompts required expertise and were time-consuming [14], [326]. Similar research targeting factual extraction and recall in knowledge graph entities has made use of task-specific prompts to evaluate LLM performance [144]. The authors formatted the inputs to match the structure of triples, thus aiming at consistent entity recognition, and explored few-shot prompting, where illustrative examples prime the model toward accurate tail prediction, especially in knowledge graph completion tasks. Despite well-crafted prompts, the authors note substantial performance degradation on long-tail entities, indicating that prompt engineering alone is insufficient to overcome LLMs' inherent knowledge gaps or biases toward over-represented entities [144,163]. This challenge is especially magnified in high-stakes, domain-specific applications such as legal research where LLMs might generate hallucinated legal cases, statutes, or citations with high confidence, as shown in [135].

Prompt engineering has also been used to craft prompts that are deliberately constructed to induce or expose hallucinations in GPT [3] or lies, including unrelated or tangential questions that reveal internal inconsistency patterns [148]. Varying a prompt's phrasing and using specially designed questions to probe the truthfulness of earlier outputs has proven to be an effective strategy to evaluate how the responses of a model change when it's challenged from unexpected angles, with research showing that lie-detection is sensitive to prompt phrasing and that its success depends to a large extent on how the questions are worded (see word sensitivity [169,326]), thus highlighting a key limitation in generalizing this method across prompt variants and domains [148,285]. The idiosyncrasies of prompt phrasing have also successfully been explored in negation-related tasks where cautionary instructions and in-context exemplars aim at guiding the model toward more accurate outputs [159]. However, the authors observe that when an LLM is asked to answer false premise prompts, any given contextual knowledge might lead it to hallucinate even more and that self-refinement might result in augmenting hallucinations instead of even partially mitigating them [159,269].

Beyond being used as a standalone primary intervention, prompt engineering also acts as a vital enabling component in a wide array of hallucination mitigation strategies which include (but are not limited to): supervised and semi-supervised learning pipelines, decoding strategies, RAG, reinforcement learning, and agent-based architectures. More specifically:

- In dataset creation and evaluation, prompt engineering has been used to generate and filter references used for inference and evaluation [37,51,142,143], systematically induce, detect, or elicit imitative falsehoods [26,100,112,126,132,148,179,315], and even create specific types of code hallucinations to test research methodologies [60].
- For confidence assessment and behavioral guidance, it has been used to elicit verbalized confidence, consistency, or uncertainty, and test or guide model behavior and alignment [48,84,86,87,144,145,159,230,232,235,267,277,290,313], reduce corpus-based social biases [270], extract and verify claims [251] as well as investigate failure cascades like hallucination snowballing [147].
- In knowledge integration scenarios it has been combined with retrieval modules or factual constraints [190,241,259,282], in agentic environments where prompts guide the generation of states, actions, and transitions [245], or the alignment process between queries and external knowledge bases [297], and even in the training process of a model where they are used to inject entity summaries and triplets [140]. Additionally, prompts have also been explored as explicit, language-based feedback signals in reinforcement learning settings, where natural language instructions are parsed and used to fine-tune policy decisions during training [305].

Some forms of prompt engineering go beyond shaping outputs or guiding behavior and instead aim to iteratively decompose prompts or elicit structured, step-by-step reasoning within the model. We will discuss such structured and iterative varieties of Prompt Engineering, such as Chain of Thought (CoT) [53] or Tree of Thoughts (ToT) [309], more extensively in the following section “Structured or Iterative Reasoning Prompting”. While prompt engineering is one of the most powerful methods for mitigating hallucinations, it also suffers from limitations and drawbacks such as:

- wording sensitivity where minor variations in wording can trigger different learned associations [169,326] or even amplify hallucinations in multi-turn interactions [147],
- over-specificity where a model fails to generalize beyond the exact pattern that a prompt encodes [53,299], thus limiting mitigation generalizability,
- scalability issues which arise from the number of intermediate tasks or their complexity [299],
- context dilution which demonstrates that prompts often fail when irrelevant context is retrieved, especially in RAG scenarios [190],
- bias amplification as a negative side effect of alignment objectives and preferences [63,305,307],
- false confidence which occurs when prompts eliciting confidence may yield overconfident incorrect answers [48,296], thus undermining trust in self-correction mechanisms,
- lack of standardized prompting workflows which makes prompt engineering a significant trial and error task not only for end-users but also for NLP experts [326], hindering reliable mitigation, and
- jailbreaking, where adversarial prompt techniques are used to manipulate a model into generating content beyond its capabilities, thus frequently resulting in severe hallucinated content or alignment violations [162,308].

5.3.2. Structured or Iterative Reasoning Prompting

Research has shown that prompt engineering can unlock reasoning abilities in LLMs, resulting in a category we refer to as “Structured or Iterative Reasoning Prompting” (Figure 13) [53,309]. This category encompasses a set of techniques that explicitly scaffold LLMs’ reasoning processes by encouraging them to externalize and systematically explore intermediate steps before committing to a final answer. Contrary to static prompts, structured and iterative reasoning aims to reduce hallucinations by decomposing tasks into verifiable steps and by enabling multi-path exploration or self-correction [268]. More specifically:

- Structured reasoning prompts modify the model's behavior in a single forward pass: the model follows the request to enumerate steps in one shot, as there is typically no separate module that determines how and when to make these steps or whether to call external tools.
- Iterative reasoning, on the other hand, can further improve the generative capabilities of a model, by guiding it to decompose a task into a series of steps, each of which builds upon, refines, and supports previous steps.

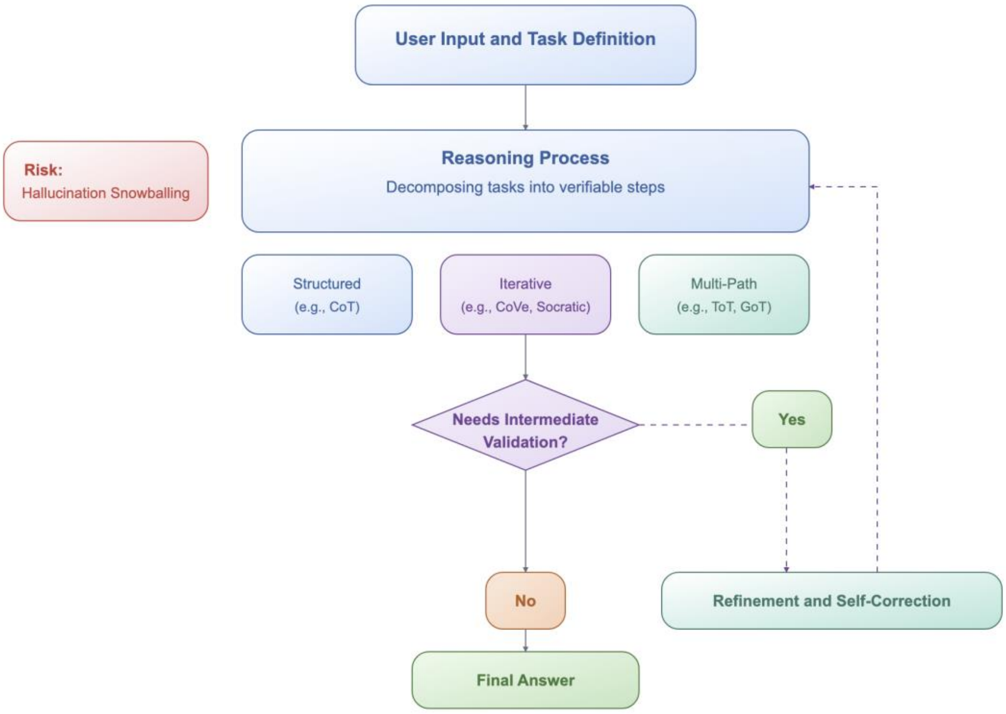

User Input and Task Definition
Risk:
Hallucination Snowballing
Reasoning Process
Decomposing tasks into verifiable steps
Structured
(e.g., CoT)
Iterative
(e.g., CoVe, Socratic)
Multi-Path
(e.g., ToT, GoT)
Needs Intermediate Validation?
Yes
No
Refinement and Self-Correction
Final Answer

Figure 13. Flowchart illustrating structured, iterative, and multi-path reasoning prompting, highlighting the role of intermediate validation in enabling refinement and self-correction before reaching the final answer.

Techniques that demonstrate the aforementioned approaches and have been successfully implemented toward mitigating hallucinations include Chain of Thought (CoT) prompting [53], Chain of Natural Language Inference (CoNLI) [51], and Chain of Verification (CoVe) [55]. Tree of Thoughts (ToT) further extends these approaches by branching the reasoning process into multiple potential solution paths as shown in [309], and further supported in a self-debugging context in [231] while Graph of Thoughts (GoT) organizes the intermediate steps into arbitrary graphs [123]. Additionally, providing the LLM with an informal logical description of the algorithm required to solve a given problem significantly reduced hallucination rates when using explanatory prompting and graph connectivity as demonstrated in [204]. By exploring these branches in parallel or sequentially, the model can compare, refine, or discard suboptimal lines of thought and arrive at a more reliable conclusion.

CoT-based reasoning, however, can also be misleading, suffering from logical inconsistencies that may not be deducible to the true reason for a certain prediction, as is shown in [169] or generate "absurd claims" and "wrong causal answers,"—particularly when engaged in complex logical tasks like causal reasoning, as highlighted in [49]. Another limitation is that iterative methods work better in larger models [92] while smaller ones may require distillation or fail to benefit [266]. Additionally, they incur higher latency/cost creating a trade-off between efficacy and efficiency [55,309]. Despite these drawbacks, the sequential or parallel exploration of multiple intermediate steps can be especially useful for complex, multi-step tasks such as:

- to iteratively decompose compound statements, generate sub-statements from the original output, check for logical inconsistencies, or synthesize logical proofs [10,51,53,176,228,231,269] or a controlled set of contrasting answers for expertise scoring [194].
- exploit the dialog capabilities of LLMs to detect logical inconsistencies by integrating deductive formal methods [75].
- use multiple CoT chains generated by an LLM to imbue smaller models with distilled knowledge and enhance comprehensive thinking [201,266]; systematically create datasets by injecting hallucinations used in training models [76] and mitigate them via reasoning [76,143].
- check and evaluate factual consistency between source and claims using iterative feedback [58,303], structured prompts [104,292,311], generate clarifying questions and correlate constraints between the query knowledge bases [297], or use retrieval and graph-based retrofitting approaches [208].
- verify or reason over intermediate steps so as to synthesize a final response [55,91], or use reasoning to introduce contextually relevant and up-to-date evidence [117] and rectify or refine this response across multiple steps [102,104,252].
- uncover why hallucinations occur by analyzing potential "risk factors" and attributing hallucination behavior to deficiencies in specific model capabilities like commonsense memorization, or memorization of disambiguation patterns [27], relational reasoning, and instruction following [237].

Despite increasing reasoning transparency, a key limitation of techniques that are based on Structured or Iterative Reasoning Prompting is that they can unintentionally amplify hallucinations, especially when early steps are flawed, thus resulting in hallucination snowballing [147]. More specifically, the authors design and compare direct prompts and zero-shot Chain of Thought (CoT) prompts, showing that prompt design directly affects both the trajectory and severity of hallucinated content.

While many structured prompting techniques follow fixed templates or scaffolds [53,55,309], others adopt more flexible, dialogic formats that simulate reasoning via iterative engagement. One such example is Socratic prompting, where the model is repeatedly questioned, challenged, or guided to refine its answers through multi-turn dialectical interaction [233], thereby reducing hallucinations through iterative self-correction and critique. A similar strategy is used in InterrogateLLM [158] which repeatedly reformulates the query before evaluating the final answer for potential hallucinations. Additionally, interrogation can take place using multi-turn interaction in an adversarial dual-model scenario, where generated claims and targeted questions are iteratively juxtaposed to discover factual errors and inconsistencies [187].

We clarify that “step-by-step” reasoning can appear in both Structured or Iterative Reasoning Prompting and Agent-based Orchestration, but our taxonomy distinguishes between them based on the locus of control. If the reasoning process is embedded directly in the user’s prompt to guide the model’s response (e.g., CoT), the control lies with the user, and we categorize it as Structured Prompting; if the reasoning is initiated and managed by an AI agent that plans, executes, and reflects over multiple steps, the control lies with the system, and we classify under Agent-based Orchestration.

5.3.3. In-Context Prompting

In-context prompting (Figure 14) is a technique that leverages the model's ability to understand and respond to prompts containing a few examples (otherwise known as “few-shot”), only one example (“one-shot”), just a textual description of the task with no examples (“zero-shot”), or instructions (instruction tuning), in an effort to guide the model to generate certain desired outputs [171,190]. Since Instruction Tuning is a specific category of “Supervised and Semi-supervised

Learning" that uses prompts to finetune a model, it is addressed in its own category in section 0, "Training and Learning Approaches. The effectiveness of In-context Prompting relies heavily on the quality of the prompts, since well-designed prompts can steer the model away from generating hallucinations by emphasizing factual accuracy and coherence, although research suggests that prompt sensitivity and the reliance on manual prompt crafting can be a limitation, making consistent hallucination mitigation challenging [171,190].

Few-shot prompting provides LLMs with a few examples (i.e., input-output pairs) within the prompt, allowing it to infer task instructions and patterns from context, rather than requiring explicit fine-tuning. Consequently, LLMs ground their responses in these explicit, relevant examples, thereby reducing the likelihood of incorrect or hallucinated content [287]. More specifically, hallucination mitigation can be attributed to:

- Pattern Reinforcement: By seeing multiple demonstrations, the model better aligns its response style and factual consistency with provided examples. For instance, Principle-Driven Self-Alignment provides 5 in-context exemplars alongside 16 human-written principles that guide the model by providing clear patterns for how it should comply with these principles, thus aligning its behavior and internal thoughts with the desired behavior demonstrated in the examples [230].
- Bias Reduction: Balanced example selection can minimize systematic biases, particularly in ambiguous queries [44,106] while few-shot examples have been used to calibrate GPT-3's responses, demonstrating how different sets of balanced vs. biased prompts significantly influence downstream performance [232].
- Contextual Precision: The model learns implicit constraints from the given examples, preventing it from generating unrelated or misleading information [190,312].

Few-shot prompting is especially effective when combined with structured reasoning techniques, such CoT prompting, which explicitly guides the model through intermediate reasoning steps [53]. CoT and few-shot examples can also be employed during the hallucination detection phase by guiding a model to identify ungrounded statements within responses, which are subsequently led to a separate mitigation agent that refines the output and reduces hallucinations [51,204]. Additionally, few-shot examples are integral to the Chain of Verification (CoVe) method, since they enable a model to perform self-correction by systematically checking its own output for factual inaccuracies [55], teach the model how to reason over retrieved evidence, sample multiple reasoning trajectories, and select the most consistent final answer across them [117,228,268], demonstrate how to verify and revise model outputs [70], guide a model's understanding regarding negation-specific tasks [159], or effectively test and observe the hallucination tendencies of existing LLMs [144].

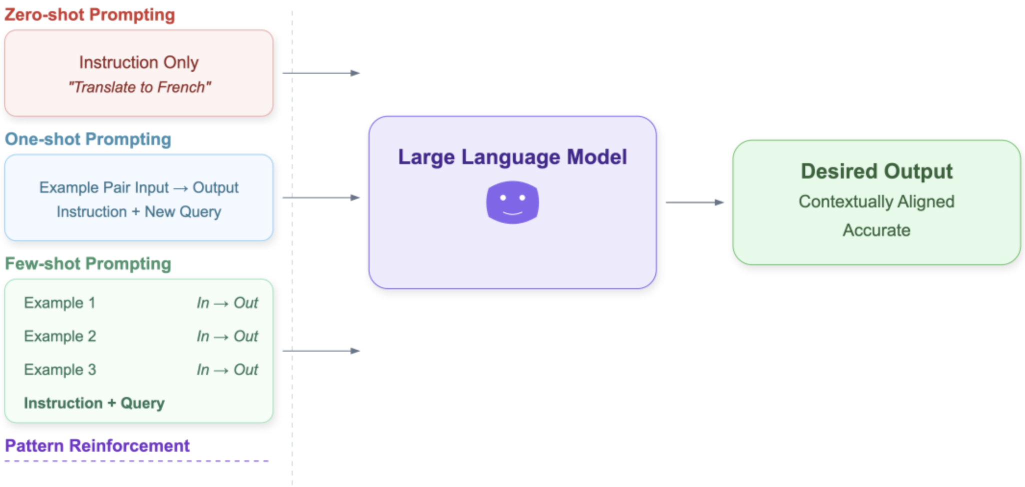


Figure 14. Illustration of in-context prompting strategies, including zero-shot, one-shot, and few-shot prompting.

Contrary to few-shot prompting, one-shot prompting provides a single example of the desired task format before the actual query, still aiming to lead the model to infer the expected response format while maintaining efficiency [190]. Compared to zero-shot prompting, one-shot prompting significantly reduces hallucination rates by anchoring the model to at least one relevant reference point, but it may still be insufficient for complex reasoning tasks, particularly in domains requiring step-by-step logical inference or factual consistency [44].

Zero-shot prompting represents the most basic form of in-context prompting, where the model is given only a natural language instruction with no explicit demonstrations and is expected to interpret and follow the instructions based solely on its pretrained knowledge [171]. While zero-shot prompting is computationally efficient and widely applicable, it is also more prone to hallucinations since the model lacks direct contextual grounding [171,190]. For instance, in [296], the authors attempted to use zero-shot prompting as a baseline to determine whether LLMs could explicitly state if a given statement was true or false, reporting that zero-shot prompting "completely failed, with accuracy levels not exceeding 52%". To counterbalance the absence of contextual grounding, zero-shot prompting often employs explicit constraints, such as instructing the model to "cite sources" or "only answer if certain" [144,173,315]. Self-Familiarity involves two distinct, sequential zero-shot prompts that generate a concise explanation for a given concept using only its internal knowledge and then instruct the model to reconstruct the original concept based solely on its generated explanation—thus gauging the model's familiarity with a concept and enabling a proactive prevention of hallucinations [329]. In InterrogateLLM, LLMs are led to generate multiple answers to the same question by systematically varying the prompt's context—providing the core mechanism for hallucination detection which is based on identifying inconsistencies across these varied generated responses [158]. In UPRISE, although the primary focus isn't on crafting prompts, natural language prompts are automated and utilized for better zero-shot performance [322].

5.3.4. Context Optimization

Context optimization provides the model with the most pertinent background information, references, and operational constraints needed to generate accurate and relevant responses. By prepending carefully crafted passages or tokens before the query, context optimization directs the model’s attention to the most relevant details, thereby reducing the likelihood of hallucination. Context selection can be implemented through external mechanisms (e.g. via retrieval systems [257,259]) or internal restructuring (e.g. via meta-prompting [198], pruning redundant inputs to focus on key information [190], or by employing a contrastive output distribution during decoding that amplifies context-aware output probabilities [312]). Context optimization can also incorporate domain knowledge or specifically curated datasets to condition the model on factual information, thus further minimizing inaccuracies and hallucinations as demonstrated in [308]. However, token limitations may necessitate techniques like hierarchical compression [59] to preserve salient facts or sliding windows [205] to prioritize grounded content.

Context optimization serves as an enabling technology for RAG approaches. In RAG, externally retrieved content from vector stores or other sources is fused into the prompt. While conventional in-context prompting relies on prompt design alone, we consider RAG an architectural modification (see section 0: Architectural Modifications), since it integrates external retrieval components into the inference pipeline. The existing prompt is augmented or conditioned on the retrieved content, thus reducing the model’s reliance on its parametric knowledge, which is prone to hallucination [149,257,259].

In addition to RAG-based approaches, dynamic prompt construction techniques include critic-driven decoding [71], structured context tagging [308], and knowledge-aware conditioning [208]. For example, in [125] the authors explicitly choose to enrich the prompt by using graph-derived signals while in [84] dialogue history and user queries are augmented with additional knowledge, thus guiding the model to prioritize relevant context and filter out irrelevant information, thereby enhancing factuality and response accuracy. Finally, meta-prompting is positioned as an advanced prompting technique that guides LLMs toward more accurate outputs by explicitly incorporating meta-level instructions within the prompt [198]. Contrary to RAG-based approaches which rely on externally retrieved content, meta-prompting uses these meta-level instructions to augment the context of the prompt internally. These instructions are distinct from persistent system-level prompts, and can be seen as task-agnostic structured scaffolds serving as reasoning and verification strategies, thus simplifying user interaction while enhancing generalization [198].

5.3.5. System Prompt Design

While both prompt engineering and system prompt design influence LLM outputs, it is important to briefly differentiate between them since they operate at different levels of control and serve distinct functions. Prompt engineering focuses on optimizing a particular query’s response through carefully structured user inputs that often involve instructions, constraints, examples, and continuous adjustments to phrasing in order to elicit a desired response [5,233], whereas system prompt design is used to create templates at the system level—invisible to the end-user—in order to establish behavioral and alignment guidelines that persist across multiple interactions and shape the model’s default reasoning style, factual adherence, and response consistency [226,307].

The aforementioned system prompts can often work by influencing the model's internal representations or attention patterns during generation, or by integrating specific knowledge and may include templates, scaffolds or constraints that shape the model’s responses and reasoning patterns; domain knowledge; ethical boundaries [63,85], or the tone and style by embedding a "role" or a "persona" that the model consistently inhabits [226]. Research has demonstrated that structured and constrained system prompts that consider factors such as sentence length, topical focus and persona behavior can steer ChatGPT 3.5 to generate differentiated, domain-specific dialogues, thereby enabling more targeted development and refinement of mitigation techniques [83]. Similarly, LaMDA models adapt their conversational behavior and content through 'domain grounding' or 'pre-conditioning'. This involves initializing the model with diverse conversational snippets, thus allowing it to effectively assume personas for specific applications [168]. A different line of research uses system prompt design to inject hallucinated examples and instructions for reasoning steps, along with specific rules and few-shot demonstrations, so as to improve hallucination detection and mitigation in models like Llama3-70B as shown in [111]. Techniques like Knowledge Consistent Alignment (KCA) leverage system-level prompts to reconcile inconsistencies between an LLM's intrinsic knowledge and external information. This is achieved by either appending reference knowledge snippets to prompts or by modifying responses into refusal formats, effectively training the model to decline answering when uncertain [166]. In the LinkQ framework, an LLM is guided through prompts to help users refine their questions and to collaboratively build queries using ground truth knowledge and graph data [210] while in [293], prompts explicitly instruct models to verify information, cross-reference with reliable sources, and actively correct misinformation. Finally, in [328] the creation of evolving instructions either in-depth (e.g., by adding constraints or reasoning steps) or in-breadth (e.g., mutating the instruction entirely) as well as the added elimination evolving step, which filters out invalid instructions, are collectively combined to increase the model’s ability to follow complex instructions.

Beyond explicit prompt engineering, research reveals that LLMs can model distinct behavioral characteristics, or 'personas', which are hypothesized to be learned and represented within the model's activation space from patterns in the pretraining data, and that the truthfulness of an answer can be predicted to a large extent from the model’s activations prior to generating this answer [226]. Contrastive prompting and activation differencing have allowed researchers to extract “persona vectors”, which are linear directions in a model’s activation space corresponding to specific behavioral traits such as hallucination propensity or sycophancy. Persona vectors support pre-finetuning data screening and post-hoc control, thus offering the ability to redirect these vectors along ethical and behavioral guidelines [225]. However, while system prompt design is a standard tool for aligning LLMs, the phenomenon of persona drift challenges the claim that system-level prompts persist across multiple interactions, as lengthy conversations may cause a model to lose its initially designated persona, and even adopt the simulated user's characteristics [194]. This observation suggests that the influence of the initial prompt decays over time, partly due to the attention mechanism which in this case destabilizes alignment [194]. Consequently, relying solely on prompts for stable persona-based model alignment may be unreliable. Additionally, due to its nature, system prompt design is inherently broad and might need to be combined with more granular prompt engineering or RLHF for maximum efficacy such as in [307], which leverages instruction-style prompts that encode general behavioral expectations to fine-tune an LLM using a combination of supervised learning and RLHF. While not structured as persistent system messages in the traditional sense, these instruction prompts function similarly by shaping the model's behavior globally during training.

While traditional system prompt design involves providing textual instructions at inference time to guide an LLM's behavior, more advanced techniques like prompt [299], prefix [229], or postfix tuning [287] extend this concept by appending learned continuous vectors to the system message in the first layer, in all layers before the system message, or in all layers after the system message, respectively. Although originally not developed for hallucination mitigation, these techniques are increasingly adopted for controlling model behavior—including hallucination reduction—via learned soft prompts or internal prefix vectors. One key limitation is that while learned prompt-based methods offer powerful ways to adapt model behavior without modifying weights, they also incur a loss of interpretability. Unlike human-readable system prompts or meta-prompts, these learned embeddings are opaque, making it difficult to audit or understand how they influence hallucination behavior [229,287].

5.4. Post-Generation Quality Control

Post Generation Quality Control encompasses a set of post-generation checks applied to text outputs, aiming to identify or correct inaccuracies after an intermediate or final text has been generated. Among these methodologies are:

- Self-verification and Consistency Checking: Involves internal assessments of output quality, ensuring logical flow, and maintaining factual coherence within the generated content.
- External Fact-checking and Source Attribution: Validates information against outside authoritative sources or asks the model to explicitly name its sources.
- Reliability Quantification: a broader subcategory that encompasses:
    - Uncertainty Estimation (quantifying the likelihood of claims) and
    - Confidence Scoring (assigning an overall reliability score to the output).
- Output Refinement: Involves further shaping and iteratively polishing the generated text.
- Response Validation: Strictly focuses on confirming that the output meets specific, pre-defined criteria and constraints.

5.4.1. Self-Verification and Consistency Checking

Self-verification (Figure 15) refers to the model's introspective evaluation of a single output's correctness using its own internal knowledge and reasoning. This includes re-parsing, self-questioning, or generating challenges to assess the factuality or coherence of its own response. Unlike consistency checking, which compares multiple outputs, self-verification focuses on whether the model can internally judge and refine a specific answer. While conceptually distinct, self-verification often overlaps with consistency-based methods, as both aim to detect hallucinations through internal evaluation.

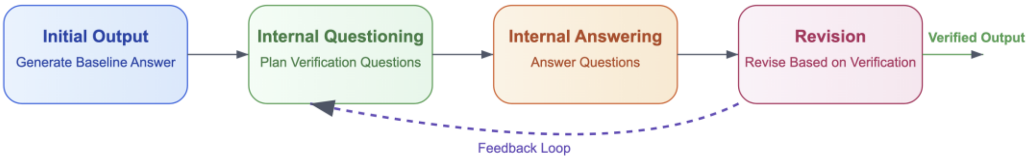


Figure 15. Simplified diagram of Self-verification.

A common approach involves multi-step introspection, as seen in CoVe [55], where the model generates a baseline answer, plans targeted verification questions, answers them, and revises its response accordingly—entirely without external tools. Similar iterative self-checking is found in [303], which frames self-verification as a loop of factual evaluation and internal correction while Socratic prompting [233] uses techniques like elenchus and dialectic to guide the model through a series of internal questions and answers. This process aims to enable the LLM to refine its own reasoning and expose internal inconsistencies in its generated information or understanding [233]. Similarly, in [147], introspective error recognition demonstrates how LLMs can recognize and acknowledge their own errors (e.g., identifying incorrect claims in explanations). Self-verification is achieved in [87] by directly querying the model about the references it generates in order to check if it possesses sufficient and consistent internal information about these cited references. In a similar vein, [170] investigates whether LLMs can self-verify their outputs and recognize when they are likely to be correct or incorrect. The authors introduce P(True), which measures the probability that a model's own sampled answer is correct, and P(IK), which estimates whether the model "knows" the answer before generating it. The authors also use calibration analysis, self-evaluation prompts, and value-head classifiers to differentiate confident, reliable outputs from uncertain or potentially hallucinatory ones. Finally, DrHall [80] is a framework for detecting and reducing factual hallucinations using metamorphic testing (MT). It reformulates original queries into semantically equivalent follow-up queries, inducing different internal execution paths and checking answer consistency to detect hallucinations, which are corrected using multi-path voting across diverse metamorphic relations.

A number of research papers directly integrate self-verification into the model architecture or training. For instance, the Self-Checks mechanism is explicitly built into the model architecture, providing internal consistency verification through rephrasing, iterative refinement, and cross-checking of generated outputs [207] while the SLM mechanism in [201] explicitly learns self-evaluation from the LLM to assess its outputs. Likewise, a dual-model setup in [95] enables one model to evaluate the other's output using token-level confidence, embedding coherence, and probabilistic anomaly detection—all without human or external feedback.

Implicit feedback or corrective mechanisms can also be used to help LLMs assess the quality of their outputs. In [121], a corrector model uses feedback loops to refine the generator's outputs based on task constraints or feedback signals (e.g., toxicity, functional correctness), functioning as an implicit verifier. Similarly, in [270] two mechanisms are introduced to help the model judge its own outputs: Self-Diagnosis, where a model identifies whether its own output contains undesirable biases, and Self-Debiasing, a decoding-time adjustment technique that reduces the likelihood of generating biased content without modifying model weights. The marginalization technique in [268] serves as a self-verification check by leveraging the intuition that a complex problem often has one correct answer derivable through multiple reasoning paths. Instead of generating a single solution, the method first samples a diverse set of reasoning paths from the language model itself. It then aggregates these paths, selecting the final answer that is most consistently supported across the different self-generated derivations.

Consistency checking assesses whether a model produces stable and coherent outputs across repeated or reformulated queries. It involves prompting the model multiple times and examining whether the responses align factually and logically. This contrasts with self-verification's focus on single-output introspection, instead emphasizing behavioral stability over time or phrasing. Despite this distinction, the boundaries often blur, as many techniques integrate elements of both internal judgment and output comparison.

One common strategy is to leverage self-contradictions across multiple generations within a LLM's own outputs. Such an approach is exemplified in the AutoHall framework [37], where the model is prompted to generate independent references for a specific claim. These references are then compared against the original reference, with the LLM itself determining if any contradictions exist between them. The presence of such conflicts indicates that the model's understanding of the claim is flawed, which in turn suggests that the original reference may be a hallucination [37]. InterrogateLLM adopts a similar philosophy by introducing a novel "zero-resource" method for detecting hallucinations by effectively having the LLM interrogate itself. Inspired by human interrogation techniques, the approach involves asking the model a series of questions without relying on any external knowledge or ground truth. The presence of inconsistencies or contradictions in the LLM's own responses to these internal probes then serves as an indicator of a hallucination in its generated answers [158]. SelfCheckGPT [274] proposes a similar method for hallucination detection that relies on response self-consistency. Instead of evaluating a single output in isolation, the model is prompted multiple times with the same input, generating a diverse set of outputs. The diverse responses obtained are then compared for semantic similarity, with greater variability across them being treated as a proxy for uncertainty or hallucinations. A more structured consistency-checking pipeline is presented in the CoNLI framework [51], which uses hierarchical verification. Each output sentence is treated as a hypothesis and is compared to the original source text using a natural language inference model. If the relationship is contradiction or neutral rather than entailment, the sentence is flagged as hallucinated. The remaining sentences are checked using named entity recognition to ensure all factual entities are traceable to the source, and only responses that pass both levels are considered consistent.

Repeated prompts can also be leveraged to test the behavioral consistency of a model as demonstrated in [148]. After a potentially false statement has been made, the model is presented with a set of unrelated follow-up questions, and the detection is then performed post hoc by analyzing the yes/no answers to these follow-up questions. This method leverages the observation that LLMs exhibit distinctive lie-related behavioral patterns, which can be uncovered through such consistency checks. A similar method is proposed in [195], which explores how reliably and consistently the model makes decisions under identical conditions in military crisis simulations. The model is prompted with the same scenario multiple times, and BERTScore is used to quantify semantic differences across its responses.

Finally, several methods at the boundary of self-verification and external fact-checking warrant clarification due to their hybrid nature. Specifically, CRITIC [70] performs an introspective reasoning loop but relies on external tools (e.g., unit tests, Python interpreters) for evidence, while DreamCatcher [175] primarily serves as an external verifier by comparing model outputs to Wikipedia references, although it also probes the model's internal knowledge. Additionally, Knowledge Consistent Alignment (KCA) [166] uses alignment data containing verified facts to detect inconsistencies, functioning as an external check despite its iterative, introspective process while RefChecker [251] explicitly operates in two modes: a zero-context setting for pure self-verification and a Noisy Context mode for external verification using retrieved knowledge. Finally, FLEEK [116] and KGR [208] exemplify hybrid pipelines where factual claims are first extracted from outputs and then validated against knowledge graphs, with the retrieved evidence sometimes feeding into a secondary generation loop.

5.4.2. External Fact-Checking

External fact-checking (Figure 16) is a fundamental, post hoc validation step that augments the model's output with verification against third-party or curated databases, websites, or other authoritative sources of information. This step is typically automated through APIs that search authoritative sources [28,33,102,104,108,113], consult knowledge graphs [163,208,210,217], or source repositories [182] to compare the model's statements with documented facts, and assign a veracity score or confidence measure. By providing an external anchor less prone to the idiosyncrasies of model training, these checks can effectively mitigate hallucinations. However, they rely on the availability and reliability of external sources, while borderline or contradictory findings may require a human-in-the-loop, as fully automating the interpretation of nuanced "facts" is often impractical [3,173]. Despite these issues and the fact that grounding does not guarantee absolute factual accuracy, external fact-checking can serve as a powerful complement to internally focused methods like self-verification, ensuring the model's outputs align with established factual records. While some hybrid methods (e.g., FLEEK [116], KGR [208], LinkQ [210]) reintegrate retrieved facts into a secondary generation loop, the majority of approaches in this category operate post hoc, leaving the model architecture unchanged. This is in direct contrast to the knowledge-integration strategies which we discussed in the section "Knowledge Representation Approaches", and which modify the model's inference process or latent representations to directly inject external knowledge.

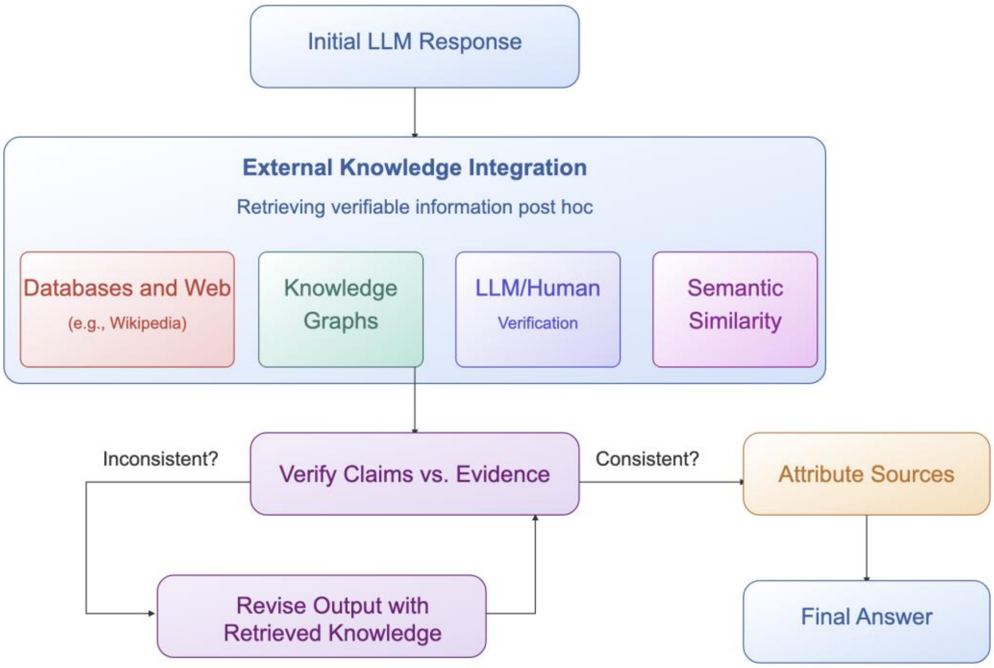


Figure 16. Flow diagram of the External Fact-Checking process, showing verification stages, intermediate validation, and feedback loops for iterative refinement of model outputs.

Many approaches integrate external knowledge directly into the LLM's process. ANAH [28], for instance, retrieves reference documents from databases like Wikipedia and Encyclopedia Britannica to validate LLM-generated sentences, while LLM-AUGMENTER [58] integrates evidence from web searches or task-specific databases into prompts, with its Knowledge Consolidator module performing retrieval, entity linking, and evidence chaining to ensure relevant and accurate information is used. Similarly, in the DRAD framework [203], the Self-correction based on External Knowledge (SEK) module activates a retrieval module when hallucinations are detected, using the retrieved knowledge to revise the output. In the Verify-and-Edit framework presented in [324], external knowledge from open-domain retrieval systems like DrQA and Google Search is introduced to correct reasoning chains. LaMDA's "Research" phase [168] iteratively queries external sources and integrates the retrieved information into the dialogue generation process. In PURR (Petite Unsupervised Research and Revision) external fact-checking is applied through denoising language model corruptions by incorporating relevant evidence, which rectifies factual errors in the generated claims, effectively acting as a fact-checking mechanism [235]. Similarly, the REFCHECKER presented in [251], in its "Noisy Context setting," utilizes retrieved knowledge to evaluate factual claims against external, relevant evidence. The Open-book QA module within the ProbTree framework also uses retrieved external knowledge to ensure factual accuracy, choosing the more confident answer between its internal knowledge and external information [231]. Finally, HERMAN verifies that specific entities, particularly quantity terms like dates and numbers, in a generated summary are supported by the original source text. This involves comparing the summary's quantity terms with those found in the original article to ensure factual consistency [250].

Several frameworks inject structured knowledge from graphs (KGs) as external ground truth. For instance, Knowledge Graph-based Retrofitting (KGR) [208] verifies factual statements by comparing them against factual knowledge in KGs, extracting claims, identifying entities, and retrieving relevant triples for verification. LinkQ [210] forces LLMs to construct and execute queries to query a KG, ensuring all data originates from ground truth, up-to-date information within the KG. Similarly, Neural Path Hunter follows a "generate-then-refine" strategy where an initial response is amended using information from the KG. A token-level "hallucination critic" identifies potential sources of factual errors in the generated text, ensuring that entities and relations in the response are supported by the KG [217]. Drowzee [91] grounds LLM outputs in a curated symbolic knowledge base derived from sources like Wikidata, using backward chaining and abductive reasoning over these entity-relation-entity triples to verify consistency and reduce hallucinations while Chain of Knowledge (CoK) in [52] specifically addresses hallucinated rationales by incorporating grounding information from heterogeneous external knowledge sources via SPARQL queries. In [33], model-based methods are leveraged to extract factual claims as relation tuples from generated text and verify them against ground-truth summaries. The authors also construct a large-scale dataset by cross-referencing facts from the Wikidata knowledge base and define a factual accuracy metric based on precision (factacc) using these extracted tuples to quantify the correctness of generated text.

A number of research papers employ the LLMs themselves, often with human oversight, for external fact-checking. In FactScore, an external retriever fetches evidence from Wikipedia, and a fact verification model checks atomic claims [113] while in [282], GPT-4 retrieves contextual information and serves as an independent factual reference, with textual entailment models comparing their answers against this context to detect contradictions. The approach presented in [292], also leverages GPT-4 to extract independent factual statements from LLM responses and then contrasts them to GPT-4's "rich world knowledge", thus essentially mimicking external verification. Similarly, TrueTeacher [311] involves an LLM annotating summaries for factual consistency by comparing them against original source documents. In Self-Checker [267], human annotators from Amazon Mechanical Turk perform external fact-checking for claim detection, evidence retrieval, and veracity prediction for the BINGCHECK dataset. Human labelers are also employed in [143] to annotate ChatGPT responses, often using search engines for verification and providing external knowledge like Wikipedia facts. Contrary to human-assisted annotation, the EVER framework uses an NLI-based verifier to compare model responses against Wikipedia evidence, using the factual feedback to train a reward model. This verifier checks whether the claims in the response are supported, refuted, or unverifiable with respect to the retrieved passages. The factual feedback is then used to train a reward model that guides answer generation through reinforcement learning, thus improving factual consistency with external sources [102]. Similarly, FacTool uses external fact-checking by extracting factual claims from LLM outputs and verifying them against retrieved evidence from external sources like Wikipedia. It decomposes the answer into atomic claims, then uses a retriever to fetch relevant documents and finally applies a fact-verification model to assess each claim separately [104]. Finally, in the MIXALIGN framework, an LLM maps the user's question constraints to knowledge bases. If this mapping is uncertain, MIXALIGN generates a series of follow-up questions for the user which are then used to filter candidate knowledge groundings and generate the final answer [297].

In addition to injecting knowledge from structured knowledge graphs or directly from other LLMs, a number of research papers focus on consistency and semantic similarity for external fact-checking. The method presented in [32] integrates fact-checking by cross-referencing LLM responses with structured knowledge bases to verify factual accuracy. This process, combined with anomaly detection and coherence analysis, helps identify and flag responses that deviate from known facts or exhibit internal inconsistencies [32], while cross-lingual semantic similarity (LaBSE, LASER), natural language inference models (XNLI), and quality estimation tools (COMET-QE) are used in [78] to measure whether a generated translation matches the source. DreamCatcher assesses the degree of hallucination in LLM generations by comparing them against a correct answer obtained from Wikipedia using token overlap and cosine similarity. This involves determining if the model possesses specific knowledge based on its generations or internal activations, effectively checking its outputs against known facts [175]. The method presented in [108] enhances factual accuracy of open-ended text generation by evaluating factual correctness against Wikipedia, using metrics like Named Entity Error Rate (NEER) and Entailment Ratio (EntailR) that require comparison to external reference material. FLEEK extracts structured fact triples from text and verifies each claim against evidence retrieved from curated knowledge graphs and web search results. It generates specific, type-aware questions for each fact and retrieves relevant answers from these sources. These answers are then used to assess whether the original claim is supported or refuted, using a comparison between the original and retrieved triplets [116]. Additionally, the Hallucination Detection Method 2 (HDM-2) framework in [138] leverages the parametric knowledge of LLMs in order to identify contradictions with "common knowledge," while including a span-level verification against known facts and retrieved context. In [140] the entity triplets and summaries injected into the model are sourced from an external knowledge base while the authors also use entailment to measure consistency between generated responses [140]. CRITIC interfaces with external tools like Google Search, Python interpreters, and toxicity APIs [70] while [182] retrieves code snippets from a local repository through similarity detection, which later serve as ground-truth-aligned evidence. The framework in [248] utilizes Hierarchical Semantic Piece (HSP) to extract multi-granularity semantic pieces from both the generated text and external "reference material". The method then uses a special fact verifier module to gauge the consistency between generated text and reference text [248], and finally, the Knowledge Consistent Alignment (KCA) uses external knowledge by having a model formulate multiple-choice questions based on external reference knowledge snippets [166].

Source attribution entails identifying and exposing the origin of the information used in generating the model’s response, whether it be from the training corpus, an external dataset, or a retrieved snippet. This method not only mitigates hallucinations by encouraging the model to rely on verifiable information, but it also promotes accountability, as each claim can be tied back to an external or internal record. For instance, the Hallucination Augmented Recitations (HAR) framework uses counterfactual open book QA hallucination-curated datasets, to ensure that the answer can only be grounded in the provided text. Furthermore, models fine-tuned with the HAR datasets are rewarded for attributing to the document, as the answers cannot be recalled from the model's parametric memory [126]. Similarly, in the "Research & Revision" phase of the Retrofit Attribution using Research and Revision (RARR) framework, an external "Document Corpus" is consulted for post-edits to the model output and for tracking sources that support the generated claims. The information retrieved from this corpus is then used to revise the original output, ensuring factual accuracy and providing an attribution report [242,243]. In both [126] and [243], however, challenges persist, such as how to handle partial matches, aggregated knowledge from multiple sources, or contradictory data.

5.4.3. Uncertainty Estimation & Confidence Scoring

In this category, we address uncertainty estimation and confidence scoring, two conceptually similar post-hoc methods for quantifying the reliability of the generated output of LLMs. Uncertainty estimation methods assess how confident a language model is in its predictions by measuring the reliability and consistency of its outputs across different scenarios or model states. These methods often involve architectural changes or prompt engineering to gauge how uncertain the model is in its own predictions. In contrast to uncertainty estimation, confidence scoring usually provides a single numerical value that reflects how confident a model is about its output. While both methods measure a model's sureness, they differ in their approach and application with uncertainty estimation being a more complex, probabilistic approach, whereas Confidence Scoring provides a single numerical value, suitable for system-level decisions.

Uncertainty Estimation

In the context of LLMs, uncertainty could refer to either aleatoric uncertainty, which characterizes genuinely ambiguous phrases or multiple valid interpretation of the same text, and epistemic uncertainty which arises from model limitations and can be addressed with more and better data [27]. Uncertainty estimation entails eliciting or quantifying how uncertain the model is in its own predictions, typically through methods like Monte Carlo dropout, Bayesian approximation, or specialized calibration layers. This approach acknowledges the probabilistic nature of deep learning models by providing a distribution of possible outputs or a range of plausible alternatives. By gauging uncertainty, the system can flag outputs that may be subsequently reviewed by an external agent or user. Uncertainty estimation methods are often hybrid methods where architectural modifications or prompt engineering are used to capture the model's true state of doubt.

- Entropy-based approaches: The Real-time Hallucination Detection method (RHD) in [203] also leverages entropy to detect output entities with low probability and high entropy, which are likely to be potential hallucinations. When RHD determines that the model is likely to generate unreliable text, it triggers a self-correction mechanism [203]. A similar mechanism Conditional Pointwise Mutual Information (CPMI) in [214], quantifies model uncertainty via token-level conditional entropy. Specifically, the method identifies hallucinated token generation as corresponding to high entropy states, where the model is most uncertain, thus confirming that uncertainty is a reliable signal [214]. INSIDE leverages the model's internal states to directly measure uncertainty with the EigenScore metric, which represents the differential entropy in the sentence embedding space while a feature clipping method mitigates overconfident generations [156]. Uncertainty in [165] is measured with Pointwise Mutual Information (PMI) which quantifies overshadowing likelihood by identifying low-confidence conditions that are likely to be overshadowed. In this case, shifts in model confidence under controlled perturbations are measured by using the probability difference between $p(y|x)$ and $p(y|x')$ which essentially serves as a method of uncertainty-based detection [165]. The concept of entropy is extended in a number of research papers to encompass semantics. For instance, in [82], the authors measure the model's uncertainty over the meaning of its answers rather than just variations in specific words by using "semantic entropy," an entropy-based uncertainty estimator which involves generating multiple answers, clustering them by semantic meaning, and then computing the entropy of these clusters to quantify the model's uncertainty [82]. Similarly, in [276] "semantic entropy probes (SEPs)" gauge the model's uncertainty from the hidden states of a single generation, which is a more efficient approach than sampling multiple responses, log probabilities or naive entropy according to the authors [276]. In an Epistemic Neural Network (ENN) extracts hidden layer features from LLaMA-2, feeding them into a small MLP ENN trained on next-token prediction, whose outputs are combined with DoLa contrastive decoding logits. This hybrid approach improves uncertainty estimation by allowing the model to down-weight low-confidence generations [249]. Finally, in [314], the conventional wisdom that hallucinations are typically associated with low confidence is challenged with the introduction of Certain Hallucinations Overriding Known Evidence (CHOKE). Specifically, the researches use and evaluate three uncertainty metrics (semantic entropy, Token probability, Top-2 token probability gap), showing that hallucinations can occur with high certainty, even when the model “knows” the correct answer [314].
- Sampling: Sampling is a classic method for uncertainty estimation as exemplified in [113], where a model's outputs are sampled multiple times and the variability in its output is used as a proxy for its uncertainty. Contrary to the confidence score being a simple, single-token probability, it

is a numerical value that is derived from the resampling process, and it acts as a signal for verification and a reward in its reinforcement learning stage [113]. Similarly, sampling is also used in [274] where increasing divergence between sampled responses is an indicator of hallucinated content and uncertainty which the authors measure with various methods such as BERTScore and NLI [274]. Finally, the framework presented in [48] leverages three main components: sampling strategies to generate multiple responses, different prompting techniques to elicit the model's uncertainty and aggregation techniques which are used to combine these multiple responses and their associated confidence scores to produce a final, calibrated confidence score [48].

- Monte Carlo methods: Fundamental uncertainty measures such as sequence log-probability and Monte-Carlo dropout dissimilarity are leveraged as key metrics to detect hallucinations and inform subsequent stages, such as refinement, detection, and re-ranking of the generated text, capturing the variability and confidence in its predictions [173]. In [245], reward estimation is based on the log probability of actions, effectively capturing how "confident" a model is about specific reasoning steps. While the authors do not explicitly term this as "uncertainty estimation," we believe that their approach overlaps significantly because the reward function evaluates the plausibility of reasoning steps. Specifically, the Monte Carlo Tree Search (MCTS) uses these rewards to guide exploration, prioritizing reasoning paths with higher estimated rewards as the reward mechanism reflects the degree of trust in the reasoning trace generated by the LLM [245].
- Explicit Verbalization: The core method in [239] trains models to explicitly verbalize epistemic uncertainty by identifying "uncertain" vs. "certain" data. It uses supervised (prediction-ground truth mismatch) and unsupervised (entropy over generations) methods and subsequently evaluates model performance with Expected Calibration Error (ECE) and Average Precision (AP), enhancing models' ability to express self-doubt about their knowledge. In a similar vein, SelfAware introduces a benchmark and method to assess a model's self-knowledge by detecting when they should verbalize uncertainty in response to unanswerable questions [88]. Similarly, [288] fine-tunes GPT-3 to produce what the authors call "verbalized probability", which is essentially a direct expression of the model's epistemic uncertainty. This teaches the model to be self-aware of its uncertainty, which is a "higher-order objective" that goes beyond the typical raw softmax-based confidence scoring. While we categorized [288] under Uncertainty Estimation, we do acknowledge that confidence scoring is a crucial component since it measures calibration of these scores using metrics such as mean square error (MSE) and mean absolute deviation (MAD). However, these scores are a direct outcome of the new verbalized probability method and not derived from existing logits [288].
- Semantic analysis: In [275], the authors propose semantic density to quantify uncertainty which measures similarity between a response and multiple completions in embedding space. Semantic density operates on a response-wise and not prompt-wise manner and doesn't require model retraining, addressing key limitations of earlier uncertainty quantification methods (like semantic entropy or P(True)). While the authors consistently use the term "confidence" as a counterpart to "uncertainty", and the final semantic density score is indeed a confidence indicator, yielding numerical scores in the range [0, 1] with provision for thresholding or filtering, we believe that this scoring is treated as the outcome of the proposed uncertainty metric [275]. Semantic analysis and logit-derived, token-level probabilistic measures are combined in [282] to calculate a confidence score for each atomic unit of an answer. This confidence is then integrated with textual entailment probabilities to produce a refined score to identify hallucinated spans. Although the authors use confidence scores that support thresholding, and consistently use the term "confidence," these confidence scores are used as part of a larger framework to detect model-generated hallucinations and thus we believe that confidence scoring is a means to an end, and that end is uncertainty estimation in [282].

- Training approaches: [280] explicitly links hard labels to overconfidence and proposes soft labels as a means of introducing uncertainty-aware supervision. This aligns with the theme of uncertainty estimation because their training objective is restructured to reflect model confidence calibration. They also evaluate overconfidence by plotting the NLL of incorrect answers, and argue that fine-tuning with soft labels reduces misplaced certainty—one of the major causes of hallucination [280]. The core contribution in [281], is a method to mitigate hallucinations by using "smoothed soft labels" as opposed to traditional hard labels that encourage overconfidence and disregard the inherent uncertainty in natural language. By introducing "uncertainty-aware supervision" through knowledge distillation, the student model learns from a more calibrated probability distribution. This approach aligns with the principle of maximum entropy and is designed to make models less overconfident and more reliable by improving factual grounding. This aligns with the theme of uncertainty estimation, because their training objective is restructured to reflect model confidence calibration. They also evaluate overconfidence by plotting the NLL of incorrect answers and argue that fine-tuning with soft labels reduces misplaced certainty—one of the major causes of hallucination [281].
- Composite methods: In [214], the authors utilize epistemic uncertainty to inform a modified beam search algorithm which prioritizes outputs with lower uncertainty, thus leading the model to reduce the generation of incorrect or nonexistent facts. In [96], a reference-free, uncertainty-based method uses a proxy model to calculate token and sentence-level hallucination scores based on uncertainty metrics. These metrics are then enhanced by focusing on keywords, propagating uncertainty through attention weights, and correcting token probabilities based on entity type and frequency to address over-confidence and under-confidence issues. Finally, in [4] the authors use the attention mechanism as a self-knowledge probe. Specifically, they design an uncertainty estimation head, which is essentially a lightweight attention head that relies on attention-derived features such as token-to-token attention maps and lookback ratios, serving as indicators of hallucination likelihood.

Confidence Scoring

Confidence scoring assigns a single numerical value reflecting the model’s conviction in the correctness of each token, phrase, or full response. Typically derived from the model’s underlying softmax probabilities, these scores provide a more lightweight and actionable measure. The practical impact is that it allows the deployment system to implement threshold-based decisions—such as refusing to provide an answer below a certain confidence level or requesting human oversight for borderline outputs. While confidence and uncertainty are two sides of the same coin, confidence scoring focuses on the practical application of a single, calibrated score for system-level decision-making, in contrast to the more intricate and probabilistic nature of uncertainty estimation.

In [95], the authors use “token-level confidence scoring” to assign a confidence score to each generated token based on the model's certainty, with lower scores indicating higher likelihood of hallucinations. This numerical score is then used in a feedback loop to dynamically adjust the output and refine the generation process [95]. Similarly, HILL explicitly incorporates various types of confidence scores, such as "Colored Ordinal CS," "Ordinal CS," and "Metric CS," as part of its prototype interfaces, while implementing a "response confidence threshold slider" that allows users to adjust the minimum confidence level for a response [145]. In [220], a novel approach called "Probabilistic Tree-of-thought Reasoning" conducts reasoning over a query tree while considering the confidence of both question decomposing and answering. The model selects the candidate “confident answers” from different QA modules, calculates confidence using log-likelihood over explanation sequences, and finally selects the most reliable answer [220]. Similarly, [231] details a method for generating and using confidence scores to evaluate and select answers from different sources. The confidence is explicitly calculated using log-likelihood over explanation sequences. This score is then used to select the most reliable source of truth between retrieved or parametric knowledge.

In Self-Highlighted Hesitation (SH2), the authors identify "low-confidence tokens" (i.e., those with lower predicted probabilities) and argue that these can be more informative for the generation process, as they can be used to force the model to "hesitate" and focus on factual information. The prediction probability is then scaled based on the difference in confidence between the original and "hesitated" inputs [277]. Similarly, [293] explores confidence shifts and their implications in multi-turn dialogues. Specifically, the authors gauge confidence using token probabilities, focusing on the impact of persuasive misinformation on confidence distribution.

Finally, in [284], the authors propose a confidence score, which is used to detect hallucination by measuring how much the model attends to the source and how much a word conveys source information. Specifically, it introduces a token-level confidence score, computed by combining an attention-based score with the probability from a base language model. This confidence score is used both during training—via a variational Bayes framework—and at inference, via reranking [284].

5.4.4. Output Refinement

Output refinement encompasses a range of post-processing techniques aimed at systematically improving the model’s initial response before final presentation. This process may include diverse strategies, from leveraging external knowledge to employing the model's inherent self-correction capabilities. Methods relying on external evidence are important, as they collect information beyond initial training data to enhance factual accuracy and coherence. These techniques involve retrieving information from external knowledge bases, integrating structured data, or incorporating various forms of external feedback. Specific examples include RAG or web searches, reasoning over structured knowledge sources like graphic neural networks (GNNs) and knowledge graphs (KGs), and external feedback mechanisms. Conversely, self-improvement methods represent a distinct paradigm where LLMs leverage their own internal capabilities, generated outputs, or self-assessment mechanisms. These approaches iteratively refine model outputs, often reducing or completely eliminating reliance on external data or extensive manual annotation. Such internal strategies involve iterative self-correction, self-regulation during generation, and the creation of self-generated data for further improvement. Additionally, some model-based techniques and tuning approaches refine outputs through internal architectural elements and learned metrics. Seen from this perspective, output refinement methods can be broadly categorized based on whether they primarily rely on external knowledge or internal, self-correcting mechanisms.

Methods that leverage external evidence encompass a set of approaches where LLMs collect information beyond their initial training data to refine and ensure the factual accuracy or coherence of their outputs. There exist a number of papers employing various such methods:

- RAG-based methods and Web searches, where external sources like documents or the web are directly used to retrieve information for output refinement. For instance, the Corrective Retrieval-Augmented Generation (CRAG) framework employs a lightweight retrieval evaluator and a decompose-then-recompose algorithm to assess the relevance of retrieved documents to a given query [69]. Similarly, EVER validates model outputs and iteratively rectifies hallucinations by revising intrinsic errors so that they align with factually verified content or re-formulates extrinsic hallucinations while warning users accordingly [102] while FAVA refines model outputs by performing fine-grained hallucination detection and editing at the span level. Specifically, it identifies specific segments of text, or "spans," that contain factual inaccuracies or subjective content and suggests edits by marking the incorrect span for deletion and providing a corrected span to replace it [112].
- Structured Knowledge Sources: These approaches integrate and reason over structured external data such as knowledge graphs or formal verification systems. For instance, [202] leverages the probabilistic inference capacity of Graph Neural Networks (GNN) to refine model outputs by processing relational data alongside textual information while [260] employs a re-ranking mechanism to refine and enhance conversational reasoning by leveraging walks over knowledge sub-graphs. Similarly, Neural Path Hunter (NPH) uses a generate-then-refine strategy that post-

processes generated dialogue responses by detecting hallucinated entity mentions and refining those mentions using a KG query to replace incorrect entities with faithful ones [217]. During the revision phase of FLEEK, a fact revision module suggests corrections for dubious fact triplets based on verified evidence from Knowledge Graphs or the Web [116] while [75] integrates deductive formal methods with the dialectic capabilities of inductive LLMs to detect hallucinations through logical tests.

- External Feedback and verification: These methods rely on external signals, human feedback, or verified external knowledge to guide the refinement process. Using the emergent abilities of CoT reasoning and few-shot prompting, CRITIC revises hallucinated or incorrect outputs based on external feedback that includes free-form question answering, mathematical reasoning, or toxicity reduction [70]. Chain of Knowledge (CoK) uses a three-stage process comprising reasoning preparation, dynamic knowledge adapting, and answer consolidation to refine model outputs. If a majority consensus is not reached, CoK corrects the rationales by integrating knowledge from identified domains, heterogeneous sources, including structured and unstructured data, to gather supporting knowledge [52]. Model outputs are refined in [324] through the Verify-and-Edit framework that specifically post-edits Chain of Thought (CoT) reasoning chains. The process begins with the language model generating an initial response and its corresponding CoT as an intermediate artifact. Subsequently, the framework generates "verifying questions" and retrieves relevant external knowledge to answer them. The original CoT and the newly retrieved external facts are then used to re-adjust the generated output, correcting any unverified or factually incorrect information [324]. The Self-correction based on External Knowledge (SEK) module presented in [203] is a key component of the DRAD framework designed to mitigate hallucinations in LLMs. When the Real-time Hallucination Detection (RHD) module identifies a potential hallucination, the SEK module formulates a query using the context around the detected error and retrieves relevant external knowledge from an external corpus. Finally, the LLM truncates its original output at the hallucination point and regenerates the content by leveraging the retrieved external knowledge, thereby correcting the factual inaccuracies [203].
- Filtering Based on External Grounding: These techniques filter outputs by comparing them against external documents or ground truth. For instance, the HAR (Hallucination Augmented Recitations) pipeline presented in [126], employs Factuality Filtering and Attribution Filtering to extract factual answers while simultaneously removing any question, document, and answer pairs where the answer is not properly grounded in the provided document [126]. HaluEval-Wild uses an adversarial filtering process, manual verification of hallucination-prone queries, and selection of challenging examples so as to refine the model outputs and ensure the dataset includes only relevant, challenging cases for evaluation [142].
- Agent-Based Interaction with External Context: These involve agents that interact with external environments, systems, or receive structured external feedback for refinement. For instance, the mitigation agent in [51] is designed to refine and improve the output by interpreting an Open Voice Network (OVON) JSON message generated by the second-level agent. This JSON message contains crucial information, including the estimated hallucination level and detailed reasons for potential hallucinations, which guides the third agent's refinement process [132].
- Model Tuning/Refinement with External Knowledge: Methods that explicitly use external knowledge during their training or refinement phase to improve model outputs. In [166], methods like refusal tuning, open-book tuning, and discard tuning are leveraged to refine the outputs of the model, thus ensuring consistency with external and intrinsic knowledge. The PURR model refines its outputs through a process akin to conditional denoising by learning to correct faux hallucinations—intentionally corrupted text that has been used to fine tune an LLM. The refinement happens as PURR denoises these corruptions by incorporating relevant evidence, resulting in more accurate and attributable outputs [235].

Complementary to methods that leverage external information, self-improvement methods represent a distinct paradigm where LLMs leverage their own internal capabilities or self-assessment mechanisms to iteratively refine model outputs, often reducing or completely eliminating reliance on external feedback or extensive manual annotation. There exist a number of papers employing various self-improvement techniques:

- Iterative self-correction, where approaches such as [252] leverage an adaptive, prompt-driven iterative framework for defect analysis, guided optimization, and response comparison using prompt-based voting. Output refinement is accomplished in [207] by employing Self-Checks where the model rephrases its own prompts or poses related questions to itself for internal consistency. Additionally, [273] uses in-context prompting to incorporate the model's self-generated feedback for iterative self-correction while [303] focuses on rewriting and improving answers to enhance factuality, consistency, and entailment through Self-Reflection. Furthermore, [269] utilizes a prompting-based framework for the LLM to identify and adjust self-contradictions within its generated text and finally, in the Tree of Thoughts (ToT) framework [309], structured exploration is guided by search algorithms like Breadth-First Search or Depth-First Search thus helping the model perform self-evaluation at various stages to refine its reasoning path.
- Self-Regulation during Generation/Decoding, where the model re-adjusts its own output or decision-making process in real-time during generation. For instance, the Self-highlighted Hesitation method (SH2) presented in [277] refines the model's output by iteratively recalibrating the token probabilities through hesitation and contrastive decoding, while the Hypothesis Verification Model (HVM) estimates faithfulness scores during decoding, refining the output at each step [302].
- Self-Generated Data for Improvement, where the LLM generates data or instructions which are subsequently used to finetune itself. For instance, the Self-Instruct framework bootstraps off the LLM’s own generations to create a diverse set of instructions for finetuning while in WizardLM such instructions are evolved and iteratively refined through elimination evolving to ensure a diverse dataset for instruction fine-tuning.
- Model-based techniques and tuning: LaMDA employs a generate-then-rerank pipeline that explicitly filters and ranks candidate responses based on safety and quality metrics. The discriminators are used to evaluate these attributes and the best-ranked response is selected for output [168]. Dehallucinator overwrites flagged translations by generating and scoring Monte Carlo dropout hypotheses, scoring them with a specific measure, and selecting the highest-scoring translation as the final candidate [189]. In [95], two models are responsible for generating the initial output and evaluating this output for inconsistencies using token-level confidence scoring and probabilistic anomaly detection. A feedback mechanism iteratively refines the output by flagging problematic sections and dynamically re-adjusting its internal parameters. Structured Comparative reasoning (SC2) combines approximate inference and pairwise comparison to select the most consistent structured representation from a number of intermediate representations [228]. In [48] techniques like sampling multiple responses and aggregating them for consistency aim to refine the model's output by filtering for coherence and reliability while the Verbose Cloning stage in [230] uses carefully constructed prompts and context distillation to refine outputs by making them more comprehensive and detailed, addressing issues with overly brief or indirect responses. The central idea in [121] is to iteratively refine sequences generated by a base model using a separately trained corrector model. This process is based on value-improving triplets of the form (input, hypothesis, correction) which are examples of mapping a hypothesis to a higher-valued correction thus resulting in significant improvements in math program synthesis, lexical constrained generation and toxicity removal [121]. Finally, the MoE architecture presented in [247] uses majority voting to filter out erroneous responses and refines outputs by combining expert contributions, ensuring only consensus-backed outputs are used and that a more accurate and polished response is generated.

5.4.5. Response Validation

Response validation evaluates the final or near-final output against specific, predefined criteria and constraints, either automatically through rule-based systems or semantically through learned models. This process may include checking for factual correctness [145], verifying mathematical reasoning [145], or matching against known templates in specialized fields. More advanced validators capable of parsing sentence-level logical coherence have also been proposed [248,282]. Given its modular nature, response validation can be easily updated or expanded as new types of errors are discovered, thus providing a structured safety net, catching errors too subtle for earlier training or architectural methods.

In neural machine translation, the "detect-then-rewrite" system presented in [78] generates multiple translation hypotheses using Monte Carlo dropout and then selects the best one based on specific criteria derived from attribution or similarity scores (attribution-based or similarity-based reranking). This process involves validating each hypothesis against predefined measures or criteria to determine the most valid output according to those metrics [78]. Hallucinations in machine translation are defined in [189] as pathological translations unfaithful to the source sequence. Dehallucinator flags suspect translations and validates them with metrics like COMET-QE to determine whether they are hallucinatory, thus ensuring that the output adheres to the constraint of faithfulness to the source sentence [189].

For the task of abstractive summarization, HERMAN works in a similar fashion by selecting the best summary from multiple candidates based on whether the quantities in the summary are validated against the source [250]. HILL is an external tool that provides a holistic set of indicators such as confidence score, sources, disclosure, and visual aids. It performs validation checks against these quality criteria and presents the results to assist human users in judging the output's validity, particularly with respect to hallucinations and factual correctness [145]. In code writing, CRITIC explicitly checks logical validity via execution (where the criterion is successful execution and correct output) and factuality via search (where the criterion is consistency with verifiable information) while clearly distinguishing the validation phase from the correction phase [70].

In addition to the aforementioned approaches, several research efforts investigate response validation at a more fine-grained level. For instance, in mathematical reasoning, the Fine-Grained Process Reward Model (FG-PRM) involves a verification task where candidate solutions are ranked and reward scores for correctness are assigned. This process directly confirms whether each reasoning step meets the specific, predefined criterion of mathematical correctness [111]. In Hierarchical Semantic Piece [248], the hallucinated segments are also detected and corrected through verifier-guided mechanisms. To detect hallucinations, the authors compute semantic similarity between sentence-level segments extracted from the generated text and their corresponding counterparts in the reference material, and specify a threshold below which the segment is flagged as a potential hallucination. Additionally, they employ a knowledge base (KB) lookup where entities and relations extracted from both the generated text and the reference material are queried against a predefined external knowledge. Once hallucinations are detected, they are mitigated by using a number of strategies such as paraphrasing, re-writing, or removing the hallucinated segments [248]. Finally, the framework presented in [282], decomposes answers into atomic units and evaluates their semantic alignment against a retrieved context. The framework combines textual entailment probabilities with token-level confidence scores to produce a refined score for each unit, indicating its hallucination probability, while a defined threshold on this refined score (e.g., < 0.5) is used to classify units as hallucinations [282].

5.5. Interpretability and Diagnostic Approaches

Interpretability and diagnostic approaches primarily focus on detection methods that help researchers understand why and where a model may be hallucinating. These include techniques such as internal state probing (which examines the model's internal variables and hidden representations), attribution-based diagnostics (which link outputs to specific inputs or internal steps), and neuron activation and layer analysis (which investigates activity in particular model components). These strategies—often collectively referred to as mechanistic interpretability approaches—aim to reverse-engineer deep neural networks and guide refinements in training or architecture as well as provide insights into model behavior based on its internal representations. Collectively, these approaches demystify hallucination triggers—whether through learned probes, activation patterns, or input attributions—thus providing a foundation for architectural refinements.

5.5.1. Internal State Probing

Internal state probing (Figure 17) involves analyzing a model's internal representations and hidden layers with diagnostic classifiers (probes) to better understand the information it encodes and how it processes inputs to produce outputs [321]. Research suggests that there exist specific patterns and correlations between internal states and hallucinations and that probing can reveal linguistic features or hallucinations [32,155], measure response consistency [140], or detect unlabeled anomalies [325]. Internal state probing is particularly valuable for evaluating the alignment between a model's internal states and its behavior, with techniques such as Principal Component Analysis (PCA) and linear classifiers being employed to demystify its otherwise opaque decision-making process [256]. However, internal state probing has limitations: probes may overfit to synthetic data, lack task generalizability or even introduce new patterns instead of revealing model knowledge [86,285].

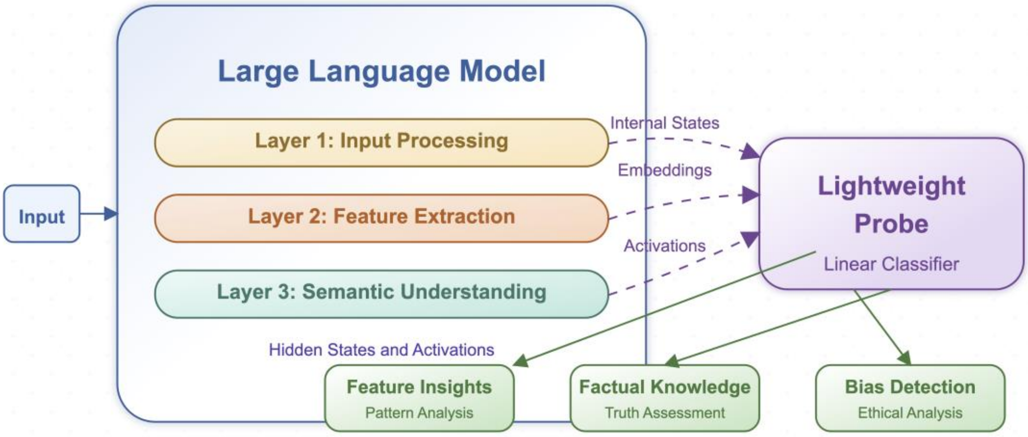


Figure 17. Visualization of internal state probing techniques used to analyze hidden layer representations and attention patterns in large language models.

While internal state probing may involve learning techniques, its fundamental purpose remains the interpretation and diagnosis of a model's internal representations. For the purposes of this discussion, we primarily categorize papers under "Internal State Probing" if their central contribution lies in the implementation of a probing classifier; otherwise, we consider "Neuron Activation and Layer Analysis" or "Attribution-based Diagnostics" as alternative categories. While we categorize methods by their primary diagnostic approach, we acknowledge that several works (such as [181] which blends probing and gradient-based attribution) defy easy categorization as they blend probing classifiers, activation analysis, and attribution methods.

In [32], OpenAI's ChatGPT and Google's Gemini have their internal states compared in terms of their susceptibility to hallucinations. The authors craft and refine hallucination-inducing prompts across diverse topics, then extract model states via attention-weight analysis and hidden layer activations. Their use of regression analysis and PCA demonstrates a strong correlation between specific internal state parameters and hallucination frequency. For practical applications, their approach included:

- Detection — employing anomaly detection and linguistic analysis.
- Mitigation — suggesting cross-referencing with structured knowledge bases [32].

Similarly, in [86], the authors ask if the diagnostic information necessary to detect hallucinations is already encoded in an LLM's self-attention output states or in its feed-forward layers. To this end, they train probing classifiers of increasing complexity across tasks such as abstractive summarization, knowledge-grounded dialogue, and data-to-text generation, aiming to detect span-level hallucinations in both sampled and synthetic datasets [86]. The authors acknowledge that narrowly trained probes generalize poorly across tasks and perform inconsistently between natural and synthetic hallucinations.

Extending the research in [86], the authors of [35] treat factual queries as constraint criteria and implement a SAT probe classifier, finding a strong positive correlation between the model's attention to constraint tokens and the factual accuracy of its output. Furthermore, they observe that focusing probe attention exclusively on constraint tokens can match or exceed the performance of next-token likelihood maximization, although they acknowledge that attention alone cannot account for all possible failure modes [35]. Building on the idea that the activation space of LLMs contains interpretable directions, Inference-time Intervention (ITI) employs linear and orthogonal probing, identifying a small number of attention heads with high linear probing accuracy and then manipulates their activations along truth-correlated directions, thereby mitigating hallucinations. [155]. The concept of ITI is extended in [218], where the proposed Non-Linear ITI (NL-ITI), employs nonlinear probes to extract richer information from internal states and implements a multi-token intervention strategy. By intervening on these activations, the model is steered towards generating more truthful and factually accurate outputs, thereby directly mitigating the production of false information or hallucinations [218]. Lookback Lens computes a "lookback ratio"—the proportion of attention given to input context tokens versus newly generated tokens—across all layers and heads, using these ratios as features. Finally, a logistic regression classifier is trained to detect hallucinated spans, and the same signal steers generation toward context-grounded outputs [188].

Shifting from token-level to sentence-level uncertainty, the INSIDE framework explores the dense semantic information of entire sentences, by analyzing the internal states of an LLM using the EigenScore metric. EigenScore quantifies the semantic consistency of multiple generated responses by computing the eigenvalues of the covariance matrix formed from their sentence embeddings—a high EigenScore suggests that the model's responses are semantically diverse, potentially signaling that hallucinated content is present. Hallucinations are addressed via an additional feature clipping technique, which truncates extreme activations, thereby reducing over-confident responses [156]. Semantic Entropy Probes (SEPs) are linear logistic regression models trained on pairs of hidden states and semantic entropy values. SEPs are computationally efficient and can be considered as proxies of faithfulness since they can be used for quantifying uncertainty [275].

By probing internal states in [227], the authors demonstrate that it is possible to predict whether an LLM's response will be truthful before it is generated, indicating the model's alignment with a truthful persona. This insight indirectly supports hallucination mitigation by finetuning an LLM on factual, truthful question-answer pairs, significantly improving its overall truthfulness on unseen topics as it reinforces its ability to align with a truthful persona [227]. In [294], the authors investigate how LLMS internally encode the truth value of statements by repeatedly reading, swapping and manipulating manipulate hidden-state vectors (residual-stream activations) to determine whether and where a "truth" signal appears. For instance, their "token-and-layer" patching heatmaps demonstrates that inserting residuals from a true prompt into a false one can flip the model's "TRUE/FALSE" logits [294]. Similarly, [296] demonstrates that a model's hidden states encode linearly separable signals indicating when it is "lying", independent of the final output logits. By training probes on intermediate representations, the authors reveal that the model "knows" it is lying even when the generated output claims otherwise. This internal probing reveals a dissociation between the model's latent knowledge and its surface-level generation [296].

LLM Factoscope decomposes model outputs into discrete factual units and locates the corresponding fact clusters within the hidden representations using gradient-based attribution. These clusters are subsequently probed using a small linear classifier to determine whether each fact is supported by the model's internal states, thereby distinguishing internally encoded from fabricated ones [181]. Similarly, to the fact clusters of the LLM Factoscope [181] and to the truth editing directions of TruthX [316], TruthForest detects "truth representations" within the model. While [290], [294] and [181] do not explicitly propose any hallucination mitigation techniques, Truth Forest extends [181] and [316] by employing orthogonal probes to intervene along truth-aligned directions, thereby biasing the attention mechanism toward processing truthful information. The authors further introduce Random Peek, a technique that applies this intervention across a wider range of sequence positions, thus ensuring that the truthfulness enhancement is pervasive and that the model is steered towards generating more accurate and less hallucinatory outputs [313]. PGFES (Psychology-Guided two-stage Fine-grained Editing and Sampling framework) presents a two-stage hallucination mitigation framework that first uses attention-augmented MLP probes to analyze intermediate activations and identify fine-grained “truthfulness directions” for different hallucination types. During inference, the model modifies hidden representations along these directions to generate multiple candidate outputs with reduced hallucination risk. A dynamic, similarity-weighted sampling mechanism is then used to integrate the most semantically consistent outputs into the final response [317]. Finally, Lightweight Query Checkpoint (LQC) extracts hidden states from intermediate layers of a smaller, non-instruction-tuned LLM and applies hybrid pooling to form query embeddings that capture both global context and key token signals. A binary classifier, trained with contrastive learning, determines whether a given query requires verification. If verification is needed, the system issues a clarification prompt rather than producing a potentially hallucinated answer, effectively reducing error propagation [180].

In addition to detecting where true statements reside in an LLM, internal state probing can also be used as a method to investigate whether LLMs encode beliefs [285]. This approach extracts embeddings from designated hidden layers and feeds them into a separate probe model. Here, the probe's objective is to infer the LLM's "beliefs" solely from these internal embeddings, without relying on the original input text [285]. Further research shows that the latent representations inherently encode (un)answerability and that there exists a linear representation subspace that separates answerable and unanswerable questions [290]. This information can be extracted using probing classifiers while the authors also explore the possibility of erasing it [290]. Finally, research on neural machine translation [319] employs several methods—including source perturbations (e.g. inserting a random token at the beginning of the source sentence), saliency analysis, and Layer-wise Relevance Propagation (LRP)—to analyze how specific tokens contribute to hallucinations. They also train a lightweight classifier on automatically generated perturbed samples, achieving strong performance in hallucination detection [319].

5.5.2. Neuron Activation and Layer Analysis

Complementing internal state probing, neuron activation and layer analysis (Figure 18) shifts the focus from classifier-based interpretations to directly observing activation patterns in an effort to understand how individual neurons and layers within a neural network respond to various inputs. Each neuron in a network typically captures a specific aspect of the input data, with earlier layers learning low-level features and later layers focusing on more abstract representations. By analyzing activations, researchers can determine which neurons or layers contribute most to a model's decisions, and identify meaningful neural activity associated with concepts such as honesty, sycophancy and other specific behaviors [256]. Techniques like activation map visualizations or attention heatmaps are widely used for this purpose, while layer-wise analysis helps elucidate the progression of feature abstraction throughout the model [57].

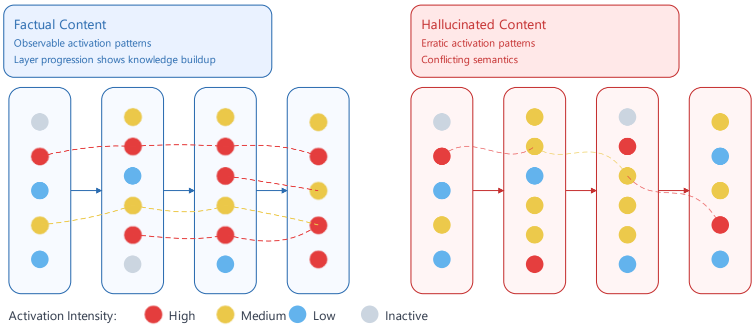

Factual Content
Observable activation patterns
Layer progression shows knowledge buildup
Hallucinated Content
Erratic activation patterns
Conflicting semantics
Activation Intensity:
High
Medium
Low
Inactive

Figure 18. A simplified overview of how neuron activations and progressive layer behaviors reveal the encoding of factual vs. hallucinated content.

In [57], the researchers investigate how internal activations quantify truthfulness by analyzing local intrinsic dimensions (LID) within these activations. LID measures the complexity of the latent space, which directly relates to how neurons activate in different layers. Specifically, they track how LID values evolve across different layers, showing a correlation between early layers and hallucinated outputs, as both have exhibit increasing LID values early on. Later layers, by contrast, show a decrease, reflecting the model's transition into a more complex and less structured manifold. The study compares LID values for human-written vs. hallucinated text, showing that human responses are more structured (lower LID), highlighting the divergence of LLMs from natural human-like generation [57]. Recent research introduces the concept of knowledge neurons, which are found in the feed-forward layers of transformers and are responsible for expressing specific factual knowledge [164]. Specifically, the authors propose a knowledge attribution method based on integrated gradients, which computes each neuron's contribution to predicting the correct entity in a cloze-style query. A refining strategy is then applied by comparing attributions across diverse prompts for the same fact, retaining only neurons that are consistently activated by that fact. Finally, these identified knowledge neurons can be selectively suppressed, amplified, or modified to probe or edit the parametric knowledge of the model without fine-tuning it [164].

In addition to being a decoding strategy, DoLa also exploits the increasing probability of factually correct answers being present in later layers compared to earlier ones, while syntactically plausible but incorrect answers remain stable throughout the network. To enhance factuality, DoLa dynamically selects what the authors call a premature layer by identifying the layer with the maximum Jensen-Shannon Divergence from the final layer's output, effectively contrasting the emergent factual knowledge in higher layers against less factual information in earlier ones [90]. Similar research employs KL-divergence analysis between intermediate and final layers to analyze hidden-state activation changes across layers and measure factuality trends in token predictions [150]. Hidden state activations across informative layers (e.g., the 26th-30th layers in LLaMA2) can also be used to identify "sharpness" in activations as an indicator of factual correctness, while cross-layer entropy is leveraged to detect factual vs. hallucinated responses, as shown in [154]. Entropy metrics and cross-attention heads are also leveraged in [289] for attention distribution analysis at the encoder-decoder interface. Additionally, Logit Lens and Tuned Lens are two methods that are explicitly designed to interpret intermediate representations and how they contribute to predictions by mapping residual streams to the vocabulary space. The authors observe how token probabilities dynamically change during inference, and particularly how correct answers manifest at different rates across layers [221].

When summarizing documents, pruning is one of the key methods often used to mitigate hallucinations, as it encourages models to rely more heavily on the source document. This increased reliance on the source leads to a higher lexical overlap between the generated summary and the original document, while research surprisingly finds that hallucinations are less prevalent in pruned LLMs compared to their original counterparts [160]. This result is further validated in TruthX [316], which independently shows that intermediate and final layers exhibit correlation with truthfulness, while Truth Forest utilizes the internal activation patterns of the LLM—particularly around attention heads and hidden layers—to construct orthogonal probes aligned with factual representations [313]. Similar research investigates how different parts of the model contribute to hallucinations by applying Layer-wise Relevance Propagation (LRP) to decompose the model's predictions into activation relevance scores across different layers [319], while neuron activations that hint at anomalous patterns, such as hallucinations or biases are analyzed in [325].

Adaptive Token Fusion (ATF) is a token management mechanism that it identifies tokens with high contextual similarity using a dynamic similarity assessment, and fuses them into a single, semantically representative token—reducing the overall token count without losing key meaning. This fusion layer sits after tokenization and embedding, functioning as a preprocessing stage before standard layers [172]. The perturbation of token embeddings and their effects on model outputs is explored in [184]. Specifically, the authors investigate how changes at the embedding and activation levels influence output, thus demonstrating how different layers and activations contribute to hallucinations [184]. Similarly, in [285], token embeddings pass through various computational layers, each of which transforms the embeddings based on information from previous tokens, with the goal of determining the depth at which truth-related information is represented [285].

We conclude this section with a number of papers that leverage statistical and mathematical methods—such as Hidden Markov Chains (HMC), Singular Value Decomposition (SVD), and Principal Component Analysis (PCA) to observe structures or hidden patterns in a model's latent space. Specifically, PoLLMgraph leverages probabilistic methods (such as Hidden Markov Chains) to extract temporal information and semantically bind state transition dynamics to a small set of manually labeled reference data. The authors conduct extensive tests and show that their PoLLMgraph successfully detects hallucinations across a variety of scenarios [227]. In a similar fashion, the authors of [283] use SVD to find principal directions in the latent space and directly edit neuron activations at specific layers [283]. Principal Component Analysis is leveraged in [294] to project neuron activations, thereby demonstrating that deeper layers separate true from false along near-linear axes that strengthen with scale [294].

5.5.3. Attribution-Based Diagnostics

Attribution-based diagnostics trace model outputs to specific inputs or internal steps, assigning importance scores to tokens or features. As diagnostic methods, they often overlap with internal state probing (e.g., gradient attribution and linear probes in [181]) and Neuron Activation and Layer Analysis (e.g., LRP across layers [319]). In this section, we examine those methods from an attributional perspective, and demonstrate how they reveal biases, spurious correlations, or contextually irrelevant dependencies, thus enabling targeted refinements.

Inspired by techniques like Grad-CAM, [183] quantifies the average gradients of input embeddings associated with each token to assign importance scores to specific inputs. This approach reveals which tokens primarily influence the LLM's internal state and subsequent hallucination estimation [183]. In ALTI+, the authors explicitly define hallucinations as translations detached from the source and quantitatively measure the contribution of each source token to each generated token, thus establishing a token-level, layer-wise attribution of how much the source influenced the output [78]. Similarly, [81] involves automatically identifying and quantifying content in the output that is not supported by the input text. By providing token-level hallucination labels, the method serves as a tool for diagnosing and interpreting model outputs, enabling the flagging of potential risks when the model is applied to new inputs [81]. Furthering this line of research, [157] extends well-known attribution methods like Gradient × Input and Input Erasure to provide "contrastive explanations" and identify "salient input tokens" that explain why a model predicts one token over another—thus linking outputs to specific inputs or internal steps and exemplifying how different input features contribute to specific decisions. Extending this research, [294] shows that causal interventions (i.e. adding or subtracting learned probed directions) can quantitatively force the model to label false statements as true and vice versa, demonstrating that the identified direction not only correlates with the truth prediction, but in fact causes it [294]. Additionally, uses causal mediation analysis and embedding-space projection, to compute layer-wise attribution features that trace hallucinations to specific attention heads and hidden states, distinguishing early-site vs. late-site causal contributors [196].

Constrained attention [35] and cross-attention analysis [154] can also function as attribution-based diagnostic tools by examining how LLMs prioritize specific "constraint tokens" within queries. Greater attention to these constraints predicts factual accuracy [35], while probing these internal attention patterns allows for the prediction of factual errors and assessment of factual constraints [154].

Research shows that statistical patterns and biases in pretraining data can cause some conditions to overshadow others, which is an attributional explanation of why certain hallucinations emerge [165]. Specifically, the authors use Pointwise Mutual Information (PMI) to pre-identify overshadowed conditions and diagnose how models prioritize information in generation. The pre-removal of conditions and tracking of probability shifts is similar to causal probing techniques, where researchers manipulate inputs to attribute model errors to specific biases [165]. In a similar study, pretraining biases related to memorization and corpus-frequency effects identified as a primary cause for hallucinations, while named entity substitution experiments further demonstrate that models use entities as memory indices [281]. The authors demonstrate that controlled interventions on NLI datasets can serve as a mechanistic probe, helping attribute false entailment predictions to memorization effects [238] rather than reasoning failures. In addition to the aforementioned training issues, potential "risk factors" may include commonsense memorization, relational reasoning, and instruction following as shown in [237]. Building on this, statistical patterns in a model's responses to a predefined set of unrelated follow-up questions after a suspected lie, can serve as a black-box diagnostic tool, as the model’s tendency to double down on lies or exhibit inconsistent behavior is itself a diagnostic and attributable signal [148]. Furthermore, [325] posits that there exist pivotal nodes responsible for encoding these spurious patterns and that these can be extracted from the model’s activations, thus demonstrating that anomalous sentences correlate with anomalous activation patterns.

We conclude this section with Faithful Finetuning (F2), which involves attribution by decomposing QA tasks into Internal Fact Retrieval and Fact-Grounded QA, thereby training the model to explicitly align responses with factual spans [209]. Using entity-based and attention-based heuristics, the authors are able to identify hallucination-prone spans to which they assign higher weights, thus attributing hallucination risk to specific entities. [154] broadens the use of entity- and attention-based heuristics to encompass entropy as a metric for diagnosing and detecting factual errors, as ROC-AUC evaluations of entropy as a hallucination indicator resemble attribution-based interpretability methods.

5.6. Agent-Based Orchestration

In agent-based orchestration we include frameworks comprising single or multiple large language models within multi-step loops, enabling iterative reasoning, tool usage, and dynamic retrieval. These architectures feature reflexive and self-reflective agents that employ reasoning to reflect on their own claims. They also encompass modular and multi-agent architectures where a number of LLM agents interact and collaborate to solve complex tasks.

5.6.1. Reflexive / Self-Reflective Agents

Reflexive and self-reflective agents (Figure 19) represent a class of artificial agents equipped with mechanisms to both respond to their environment in real time (reflexivity) and critically analyze their own behavior, goals, and internal states (self-reflection) [253,303]. As we have mentioned earlier, this ability to “critically analyze” can manifest in both CoT-based techniques as well as in self-reflection. As discussed in the section on Structured or Iterative Reasoning Prompting, we distinguish between these two cases based on how and when the step-by-step reasoning is applied, with emphasis on the locus of control. In Agent-based Orchestration, the step-by-step process is not directly dictated by the user prompt but is instead managed by an agent responsible for planning, executing, and reflecting on a sequence of actions to achieve a goal—either autonomously or through collaboration with other agents in a multi-agent system.

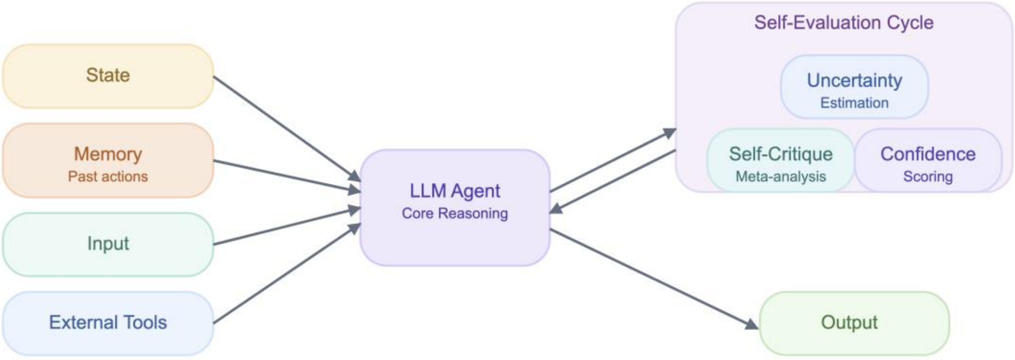


Figure 19. Diagram showing the architecture of a self-reflective agent that utilizes external tools and introspective feedback loops.

Self-reflection is considered one of the emergent abilities, typically observed only when a model reaches a certain scale (e.g., 70B parameters) [92]. In contrast, for smaller models, this introspective process can be detrimental—leading to the questioning of correct answers and the generation of hallucinated content [92,292]. Moreover, the effectiveness of self-reflection in mitigating hallucinations in LLMs is highly domain-sensitive—showing substantial benefits in open domains but only marginal improvements in specialized areas such as finance and science, due to a lack of self-corrective knowledge [92,292]. Reflexive agents are designed for rapid responses to external stimuli, whereas self-reflective agents often use confidence scoring or entropy-based uncertainty measures to trigger reflection [48,239,275]. They are frequently integrated into iterative reasoning frameworks, where they assess and refine their outputs in cycles [303], which allows them to reason about their decision-making processes, evaluate the outcomes of their actions, and adapt strategies based on past experiences, thereby improving accuracy and contextual relevance over time [70,253].

For instance, [199] explores meta-thinking, defined as self-reflection, self-evaluation, and self-regulation of reasoning processes. The implement techniques such as self-distillation, self-checking, meta-reward formulation, and internal error detection—centered on the idea that LLMs can become self-aware of their reasoning chains and improve them [199,303]. In a similar vein, the CRITIC framework introduced in [70] exemplifies a self-reflective agent where an LLM critiques, verifies, and refines its own outputs through interactions with external tools such as Google Search and code interpreters, orchestrating a self-corrective loop that relies on external fact-checking. Reflexion is another framework built around the principle of self-evaluation and verbal reflection, designed to iteratively improve LLM-based agents across decision-making, reasoning, and programming tasks. The Self-Reflection model (Msr) explicitly generates natural language critiques of the agent's past actions, stores them in memory, and uses them as feedback [253].

Inspired by the concept of self-reflection, Self-Refine models a single LLM playing multiple roles as it generates and reflects on its own output, improving it through structured, multi-round internal dialogue. Unlike frameworks that depend on search engines, Self-Refine is tool-independent, relying solely on the LLM's internal capacity for self-critique, and task-agnostic as no model training or domain-specific data are required. While this approach is limited by the fact that base models need to have sufficient few-shot modeling or instruction-following capabilities, the authors report that Self-Refine significantly outperforms one-shot baselines across seven diverse tasks [273]. Similarly, Graph of Thoughts (GoT) incorporates elements of self-evaluation and refinement of generated thoughts, drawing parallels to self-reflection frameworks. Though it is not framed as a full agentic architecture, the presence of modules like controllers, evaluators, and scoring functions suggests that it aligns conceptually with reflexive agent systems [123].

Despite their potential, the development of reflexive and self-reflective agents faces several challenges, including the computational overhead associated with meta-reasoning and the complexity of designing agents capable of balancing reflexivity with reflective deliberation [19]. Nevertheless, advancements in cognitive architectures and meta-learning are progressively addressing these challenges, enabling the development of agents with robust and adaptive intelligence suitable for high-stakes applications [61].

5.6.2. Modular and Multi-Agent Architectures

While self-reflective agents emphasize internal mechanisms for improvement, more complex scenarios often necessitate collaborative intelligence. Modular and multi-agent architectures (Figure 20) constitute a structured, meta-level approach to controlling large language models by embedding them within broader decision-making frameworks. These frameworks may include a number of specialized agents capable of dynamically invoking external tools—such as database query engines or fact-checking APIs—and orchestrate multiple steps of reasoning and retrieval to refine outputs prior to generating their final response. AI agents also maintain a dynamic memory of intermediate outcomes, reflect on them, and adjust their strategies in real time [10], [18], [151], [323]. Hallucinations are usually mitigated by cross-verifying the outputs of the various agents, as well as by feedback loops before the final output is generated. Additionally, the issue of misalignment between human values and training objectives can be addressed by training AI agents directly on human preferences over trajectory segments using reinforcement learning, as shown in [74] or via alternative RL approaches as demonstrated in [213]. This ensures that the agents learn to perform tasks that genuinely align with human intent, thereby reducing the likelihood of generating unintended behaviors—effectively mitigating hallucinated agent actions [74].

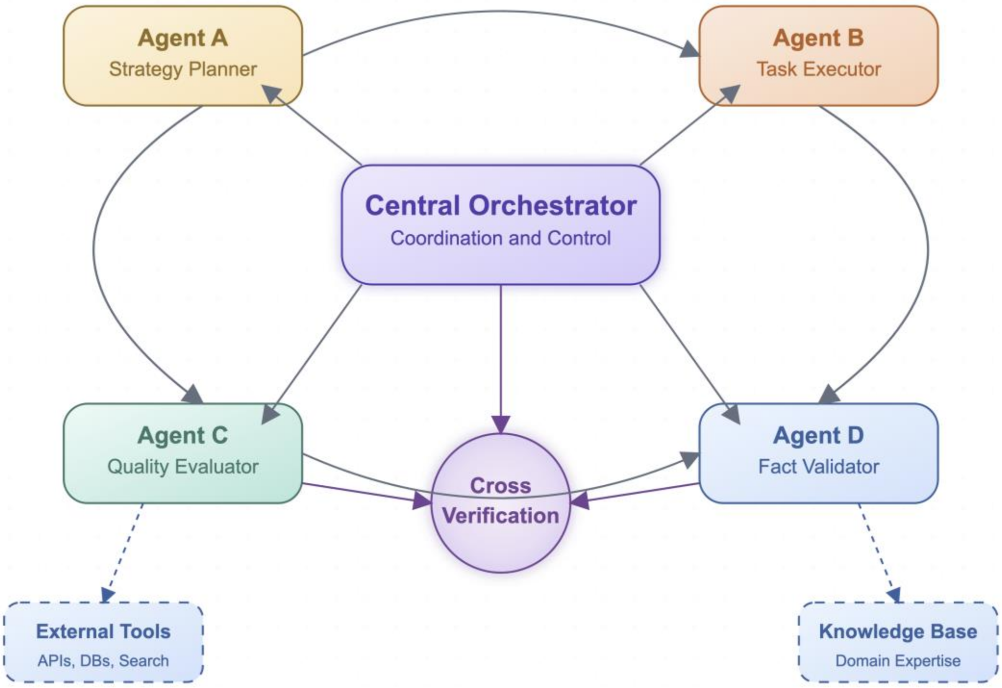


Figure 20. Illustration of multi-agent systems where specialized LLM agents—such as planners, executors, evaluators and validators—collaborate in orchestrated workflows. These architectures may incorporate retrievers, feedback mechanisms, and invocations of external tools.

In addition to self-reflective capabilities discussed in the previous section, [199] introduces multi-agent strategies such as supervisor-agent hierarchies, agent debates, and self-play setups. Specifically, the authors propose a modular decomposition of reasoning tasks where different agents (e.g., planner, evaluator, executor) interact through defined roles and communication protocols. These architectures support collaborative meta-reasoning and help overcome the limitations of single-agent feedback loops, especially for high-complexity tasks [199]. Decomposition is also employed in [10], where the Decompose-and-Query (D&Q) framework functions as an agentic system, guiding an LLM through a structured, multi-stage process of thought and action. Specifically, the LLM decomposes complex questions and queries external knowledge bases, while leveraging tools to gather reliable information. The authors argue that the iterative approach of D&Q is similar to ReAct—a prominent pattern for building AI agents that interleave reasoning steps with calls to external tools or environments—and that it fortifies the LLM against generating spurious content, thereby mitigating hallucinations [10]. Similarly, the multi-agent debate framework proposed in [151] leverages multiple LLM instances that independently generate answers which are subsequently enhanced through inter-agent consensus and iteratively refined via debate rounds. This method requires only black-box access and significantly reduces hallucinations especially in tasks like biographies and Massive Multitask Language Understanding (MMLU) while it outperforms models based on Self-reflection [253,273] across six tasks, including arithmetic, biographies, and chess. Furthermore, a multi-agent system is employed in [323] to reduce AI hallucinations in customer service interactions. Here, a Knowledge Retrieval Agent provides factual grounding to LLM Agents, a Fuzzy Logic Agent assesses the certainty and truthfulness of LLM-generated responses while a Validation Agent cross-references these outputs against reliable information, collectively ensuring higher reliability and factual consistency to mitigate hallucination risks [323]. In [132], the authors design a prompt-based pipeline that induces hallucinations using various methods, such as exploiting knowledge gaps, blending facts with fiction, or encouraging the model to be speculative. Two more agents, employing distinct LLMs and diverse strategies, are tasked with detecting false claims and clarifying ambiguities, while a fourth agent evaluates KPI score levels. The entire pipeline leverages the OVON framework (Open Voice Network), which is designed to enhance the reliability and explainability of AI-generated responses [132]. Finally, RAG-KG-Incremental Learning is a multi-agent hybrid framework that integrates RAG, knowledge graph reasoning, and incremental learning to mitigate hallucinations. The system employs multiple AI agents for tasks such as query understanding, data selection, text extraction, KG generation, and reasoning. Its knowledge graph stores newly extracted entities and relationships, enabling the model to continuously expand its knowledge without requiring full retraining [240].

Conceptually aligned with multi-agent architectures, the study in [187] instantiates the same model twice (Examiner and Examinee), using different prompt instructions in an interaction that mimics modular agents working in tandem on specific sub-tasks. Similarly, [267] implements a modular framework where a "policy agent" that dynamically orchestrates the sequence of actions based on the input and task, providing valuable insights into the overlap between external verification and orchestration-driven frameworks. Likewise, the use of agents for detection and mitigation in [51] suggests an orchestration-like setup, although the described agents do not appear to incorporate external tools, dynamic retrieval, or verbose reasoning. RAP (Reasoning via Planning) is an agentic framework in which the LLM serves dual roles: first, as a reasoning agent that generates actions and constructs reasoning traces (i.e. steps toward solving a problem) and second, as a world model that predicts future states based on the agent's actions. This agent-like behavior involves dynamic exploration and reasoning, with the LLM self-guiding through reasoning paths while balancing exploration and exploitation using Monte Carlo Tree Search (MCTS) [245].

Although AI agents have been successfully employed in mitigating hallucinations—either directly or indirectly—current AI agent face several limitations, as outlined in [19]. These include a narrow focus on accuracy, which leads to overly complex and costly agents, and to flawed conclusions about performance gains. Furthermore, cascading hallucinations may propagate in multi-agent scenarios [147], while over-reliance on external tools can be detrimental due to outdated or incomplete information [190,241]. Additionally, inadequate holdout sets in benchmarks promote overfitting and result in suboptimal agents that perform poorly in real-world applications. Moreover, the lack of standardization in evaluation practices leads to reproducibility issues and inflated accuracy estimates, thereby hindering a true understanding of agent capabilities. Finally, designing effective systems requires addressing challenges such as scalability, balancing autonomy and coordination, developing efficient communication protocols, and optimizing agent interactions for high-stakes applications. Continuous research—particularly on multi-agent reinforcement learning (MARL) promises to advance these architectures, enabling their deployment in increasingly complex scenarios [213].

6. Benchmarks for Evaluating Hallucinations

The evaluation of hallucinations presents inherent difficulties due to their context-dependent nature. As a consequence, researchers aim at combining automated metrics as well as task-specific benchmarks, so as to ensure a more comprehensive coverage of hallucinations ranging from blatant to nuanced. For instance, while RAG metrics excel in grounding, semantic entropy captures intrinsic uncertainty and while factuality benchmarks (FACTSCORE [106], FRANK [320]) enable fine-grained error typology analysis, TruthfulQA tests adversarial robustness. However, despite the plethora and variety of metrics, no single metric or benchmark fully encapsulates their complexity, thus rendering human assessments indispensable, particularly for tasks where context sensitivity and domain-specific expertise are essential [101]. Moreover, many popular dialogue benchmarks contain hallucinated gold responses (even as high as 60%) [223], potentially overstating model performance and highlighting the need for more reliable evaluation datasets. These findings suggest that human evaluation, despite being expensive and time-consuming, remains crucial for assessing the subjective quality and coherence of LLM outputs [101]. We have compiled a by no means exhaustive list of benchmarks and have categorized them into three categories: Factual Verification Benchmarks, Domain-Specific Benchmarks and Code Generation Benchmarks:

- Factual Verification Benchmarks: These benchmarks focus on assessing the factual accuracy of LLM outputs by comparing them against established ground truth.
    - ANAH is a bilingual dataset for fine-grained hallucination annotation in large language models, providing sentence-level annotations for hallucination type and correction [28]
    - BoolQ: A question answering dataset which focuses on yes/no questions, requiring models to understand the context and before deciding [43].
    - DiaHalu is introduced as the first dialogue-level hallucination evaluation benchmark for LLMs. This dataset aims to address the limitations of existing benchmarks that often focus solely on factual hallucinations by covering four multi-turn dialogue domains: knowledge-grounded, task-oriented, chit-chat, and reasoning [83].
    - FACTOR (Factuality Assessment Corpus for Text and Reasoning) is a benchmark designed to evaluate the factuality of language models, particularly their ability to perform multi-hop reasoning and retrieve supporting evidence [120]. This benchmark focuses on assessing whether a model's generated text aligns with verifiable facts, demanding not only information retrieval capabilities but also the ability to reason over multiple pieces of evidence to validate claims.
    - FACTSCORE [106] contributes a fine-grained, atomic-level metric designed to assess the factual precision of long-form text generation. Accompanied by a large benchmark of 6,500 model generations from 13 LLMs, FACTSCORE enables comparisons across models by distinguishing supported, unsupported, and unverifiable statements within a single output.
    - FELM (Factuality Evaluation of Large Language Models) is a benchmark dataset designed to evaluate the ability of factuality evaluators to detect errors in LLM-generated text. FELM spans five domains—world knowledge, science/tech, math, reasoning, and

writing/recommendation—and is primarily used to benchmark factuality detection methods for long-form LLM outputs [109].

- FEVER (Fact Extraction and Verification): FEVER is a dataset consisting of 185,445 claims that are classified as "Supported", "Refuted" or "NotEnoughInfo" [110]. The researchers argue that FEVER is a challenging testbed, since some claims require multi-hop reasoning and information retrieval, and can be used to assess the ability of models to not only retrieve relevant evidence but also to determine the veracity of claims based on that evidence.
- FEWL (Factuality Evaluation Without Labels): FEWL is a methodology for measuring and reducing hallucinations in large language models without relying on gold-standard answers [194]. This approach focuses on evaluating the factual consistency of generated text by using a combination of techniques, including self-consistency checking and external knowledge verification, to identify potential hallucinations.
- The FRANK benchmark introduces a typology of factual errors grounded in linguistic theory to assess abstractive summarization models. It annotates summaries from nine systems using fine-grained error categories, providing a dataset that enables the evaluation and comparison of factuality metrics [320].
- HADES (HAllucination DEtection dataset) is a reference-free annotated dataset specifically designed for hallucination detection in QA tasks without requiring ground-truth references, which is particularly useful for free-form text generation where such references may not be available. The creation of HADES involved perturbing text segments from English Wikipedia and then human-annotating them through a "model-in-the-loop" procedure to identify hallucinations [128].
- HalluEditBench is a dataset comprised of verified hallucinations across multiple domains and topics and measures editing performance across Efficacy, Generalization, Portability, Locality, and Robustness [46].
- HalluLens is a comprehensive benchmark specifically designed for evaluating hallucinations in large language models, distinguishing them from factuality benchmarks [139]. HalluLens includes both intrinsic and extrinsic hallucination tasks and dynamically generates test sets to reduce data leakage. The authors argue that factuality benchmarks like TruthfulQA [315] and HaluEval 2.0 [292] often conflate factual correctness with hallucination, although hallucinations are more about consistency with training data or user input rather than absolute truth. HalluLens, therefore, offers a more precise and task-aligned tool for hallucination detection.
- HALOGEN (Hallucinations of Generative Models) is a hallucination benchmark designed to evaluate LLM outputs across nine domains including programming, scientific attribution, summarization, biographies, historical events, and reasoning tasks. HALOGEN is used to measure hallucination frequency, refusal behavior, and utility scores for generative LLMs, providing insights into error types and their potential sources in pretraining data [141].
- HaluEval: HaluEval is a large-scale benchmark for the evaluation of hallucination tendencies of large language models across diverse tasks and domains [143]. It assesses models on their ability to generate factually accurate content and identify hallucinated information. The benchmark includes multiple datasets covering various factual verification scenarios, making it a comprehensive tool for evaluating hallucination detection and mitigation performance.
- HaluEval 2.0: HaluEval 2.0 is an enhanced version of the original HaluEval benchmark, containing 8,770 questions from such diverse domains as biomedicine, finance, science and education, offering wider coverage, more rigorous evaluation metrics for assessing factuality hallucinations in LLMs as well as challenging tasks to better capture subtle hallucination patterns and measure model robustness [292].

  - HaluEval-Wild: HaluEval-Wild is a benchmark specifically designed to evaluate hallucinations within dynamic, real-world user interactions as opposed to other benchmarks that focus on controlled NLP tasks like question answering or summarization [142]. It is curated using challenging, and adversarially filtered queries from the ShareGPT dataset. These queries are categorized into five types to enable fine-grained hallucination analysis.
  - HDMBench is a benchmark designed for hallucination detection across diverse knowledge-intensive tasks [138]. It includes span-level and sentence-level annotations, covering hallucinations grounded in both context and common knowledge. HDMBench enables the fine-grained evaluation of detection models, supporting future research on factuality by providing unified evaluation protocols and high-quality, manually validated annotations across multiple domains.
  - Head-to-Tail: Head-to-Tail delves into the nuances of factual recall by categorizing information based on popularity [144]. It consists of 18,000 question-answer pairs and segments knowledge into "head" (most popular), "torso" (moderately popular), and "tail" (least popular) categories, mirroring the distribution of information in knowledge graphs.
  - HotpotQA: HotpotQA evaluates multi-hop reasoning and information retrieval capabilities, requiring models to synthesize information from multiple documents to answer complex questions and it has proven to be particularly useful for evaluating factual consistency across multiple sources [146].
  - NQ (Natural Questions): NQ is a large-scale dataset of real questions asked by users on Google, paired with corresponding long-form answers from Wikipedia [215]. It tests the ability to retrieve and understand information from a large corpus.
  - RAGTruth is a dataset tailored for analyzing word-level hallucinations within standard RAG frameworks for LLM applications. It comprises nearly 18,000 naturally generated responses from various LLMs that can also be used to benchmark hallucination frequencies [241].
  - SelfCheckGPT is a zero-resource, black-box benchmark for hallucination detection. It assesses model consistency by sampling multiple responses and measuring their similarity, without needing external databases or model internals [274].
  - TriviaQA is a large-scale question-answering dataset that contains over 650,000 question-answer-evidence triplets that were created by combining trivia questions from various web sources with supporting evidence documents automatically gathered from Wikipedia and web search results [309]. The questions cover a diverse range of topics with many of those questions requiring multi-hop reasoning or the integration of information from multiple sources in order to generate a correct answer.
  - TruthfulQA: TruthfulQA is a benchmark designed to assess the capability of LLMs in distinguishing between truthful and false statements, particularly those crafted to be adversarial or misleading [315].
  - The UHGEval benchmark offers a large-scale, Chinese-language dataset for evaluating hallucinations under unconstrained generation settings. Unlike prior constrained or synthetic benchmarks, UHGEval captures naturally occurring hallucinations from five LLMs and applies a rigorous annotation pipeline, making it a more realistic and fine-grained resource for factuality evaluation [318].
- Domain-Specific Benchmarks: These benchmarks target specific domains, testing the model's knowledge and reasoning abilities within those areas.
  - PubMedQA: This benchmark focuses on medical question answering, evaluating the accuracy and reliability of LLMs in the medical domain [234].
  - SciBench: This benchmark verifies scientific reasoning and claim consistency, assessing the ability of LLMs to understand and apply scientific principles [265].

  - LegalBench: This benchmark examines legal reasoning and interpretation, evaluating the performance of LLMs on legal tasks [177].
- Code Generation Benchmarks (e.g., HumanEval, Codex): These benchmarks assess the ability of LLMs to generate correct and functional code, which requires both factual accuracy and logical reasoning [99].

These benchmarks, summarized in Appendix B, critically examine hallucination phenomena across multiple dimensions: factual verification, contextual coherence, domain-specific reliability, and cross-modal consistency. Arguably, the most sophisticated benchmarking strategies integrate interdisciplinary perspectives, drawing from epistemology, cognitive science, and computational linguistics in order to create assessment frameworks that capture the multifaceted nature of hallucinations.

7. Practical Implications

Benchmarks quantify what to measure (faithfulness, evidentiary grounding, uncertainty, calibration), but high-stakes deployment needs operational guardrails. Drawing on the evidence surveyed in this review, we offer a concise, practice-oriented guideline for researchers and practitioners deploying hallucination-mitigation methods in high-stakes settings (e.g., healthcare, medicine, defense). The steps below are heuristics, not mandates: they set principled defaults that must be adapted to each application's risk profile, domain norms, data governance, and operational constraints. Because every system has its own idiosyncrasies, treat these recommendations as a starting point to be validated with local evaluation, safety cases, and human-in-the-loop checks. The guidance distills recurring patterns from our research and maps benchmark dimensions (faithfulness, grounding, uncertainty) to concrete design decisions [106,109,110,128,142,143,315].

- Make evidence the default: Prefer retrieval-augmented and evidence-linked generation, preferably from well curated knowledge bases; surface source spans for non-trivial claims. Favor designs that bind decoding to retrieved context and revision passes [58,242,244,255,258].
- Calibrate—and abstain when needed: Expose uncertainty (entropy/semantic entropy, PMI-based signals, internal-state metrics) and combine with self-consistency; enforce a strict abstain/deferral policy below thresholds [23,82,156,165,268,276].
- Constrain decoding with verification in the loop: Use conservative decoding and inference-time checks; rerank with faithfulness signals before release [79,90,154,218,302].
- Add post-hoc fact-checking by default: Run external verification on claims and record provenance; separate self-verification from evidence-based checks to avoid false reassurance [70,116,208,210,217,251].
- Keep humans in the loop with role clarity: Use Self-Reflection and external fact-checking pipelines to route low-confidence or conflicting outputs to a designated reviewer; require dual sign-off for irreversible actions when sources disagree.
- Evaluate with task-relevant metrics: Report faithfulness and grounding alongside utility; don't rely on surface metrics alone [58,242,244,255,258,320].
- Log everything for audits: Persist prompts, retrieved docs, model/version, verifier scores, and decision outcomes to enable incident analysis and rollback with pipelines such as [241–243].

8. Challenges

The mitigation of hallucinations in large language models represents a computational and theoretical challenge at the intersection of deep learning, cognitive science, and epistemology. Current research confronts fundamental questions about the nature of AI-generated knowledge, the mechanisms of uncertainty representation, and the epistemological boundaries between generative creativity and factual reliability. Key obstacles include limited generalization across domains, insufficiently interpretable internal mechanisms, and the scarcity of standardized benchmarks for consistent evaluation. Research must also address efficient verification under latency constraints, and methods resilient to adversarial or distribution-shifted inputs. Promising avenues include hybrid retrieval–generation frameworks, self-reflective reasoning agents, and knowledge-grounded fine-tuning that balances factual accuracy with creative generation. Moreover, ethical considerations demand sophisticated approaches that balance the potential for creative knowledge generation with stringent requirements for factual accuracy, particularly in high-stakes domains.

9. Conclusions

The exploration of hallucinations in large language models reveals a complex, dynamic research landscape characterized by rapid technological evolution and profound epistemological implications. This taxonomy represents more than a mere categorization of mitigation strategies; it serves as a critical analytical framework for understanding the fundamental challenges in generating reliable, contextually appropriate machine-generated knowledge. The research underscores the necessity of multidimensional, interdisciplinary approaches to hallucination mitigation. No single methodology provides a comprehensive solution, highlighting the inherent complexity of developing artificially intelligent systems capable of nuanced, contextually grounded communication. Future research trajectories must prioritize developing interpretable, adaptable mitigation techniques that can generalize across diverse linguistic and computational contexts, ultimately moving towards more robust and epistemologically sophisticated language models.

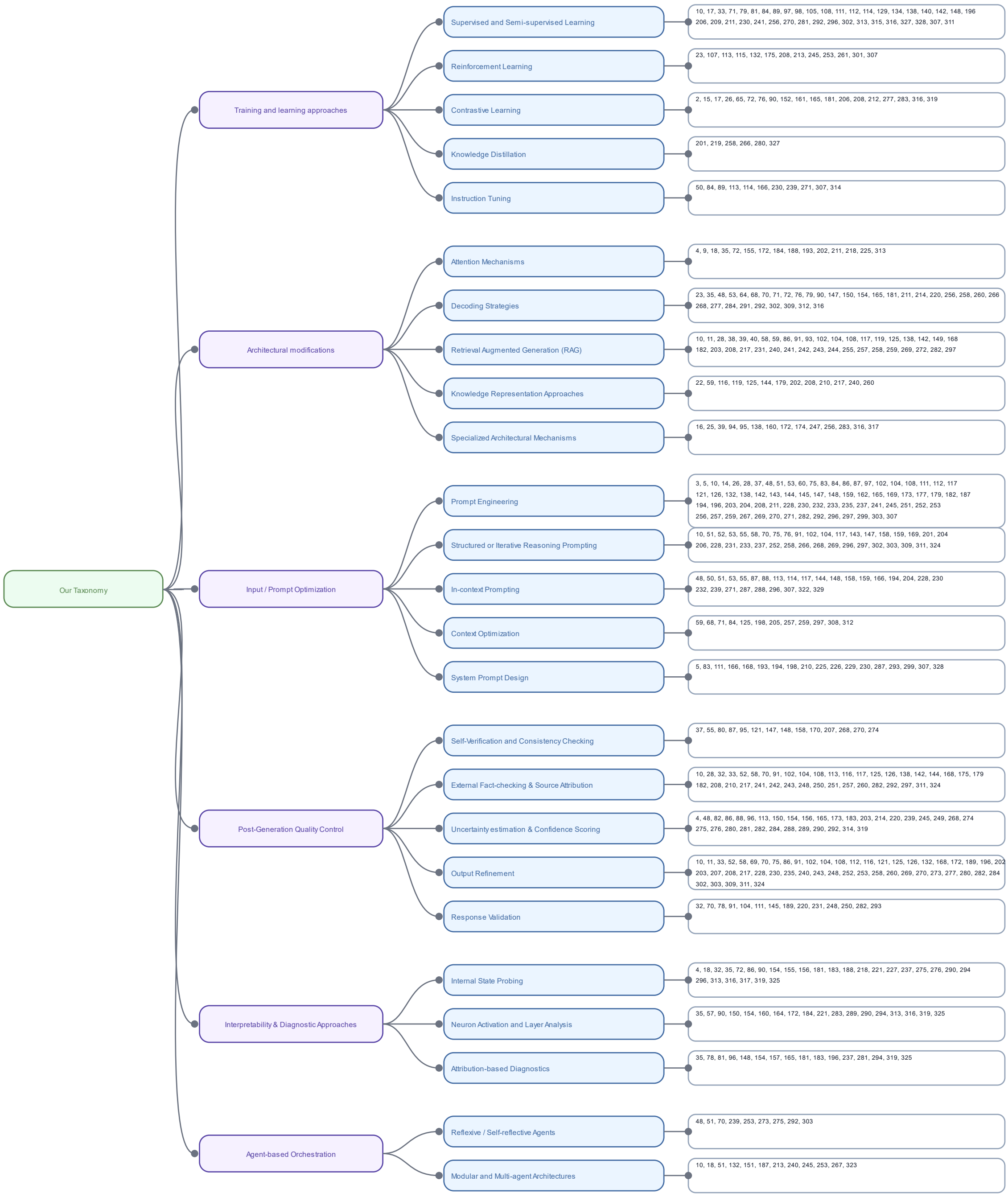
Author Contributions: All authors have contributed to the review presented. Conceptualization, writing original draft preparation, writing review and editing, visualization, I.K.; supervision, E.A., K.D. and C.B.; All authors have read and agreed to the published version of the manuscript.

Funding: This review article has not received external funding.

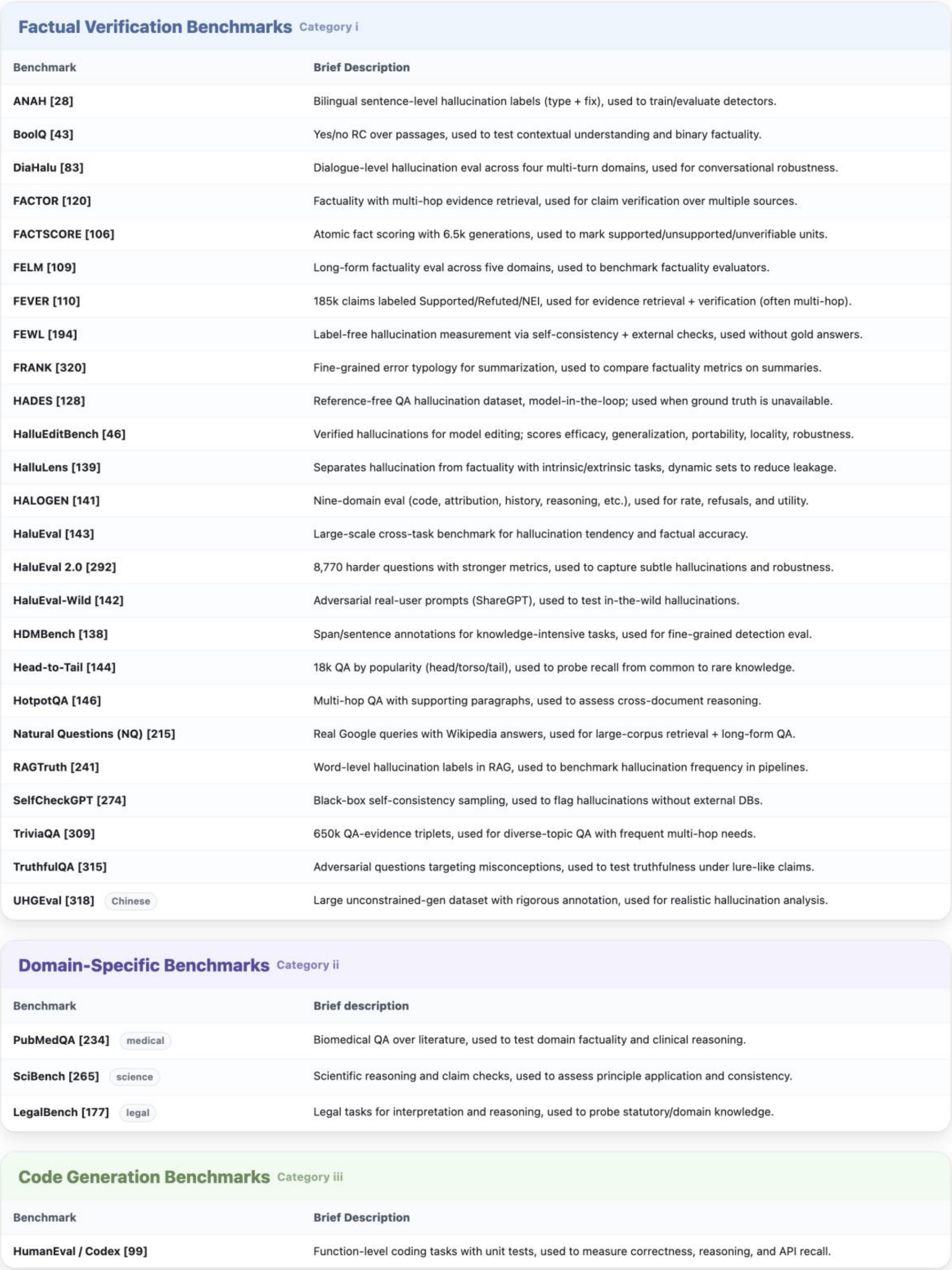
Data Availability Statement: Data are contained within the review article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Our Taxonomy Presented in Categories, Subcategories and Associated Papers



Appendix B. Summary Table of Benchmarks Used in Hallucination Detection and Mitigation



Factual Verification Benchmarks Category i

Benchmark	Brief Description
ANAH [28]	Bilingual sentence-level hallucination labels (type + fix), used to train/evaluate detectors.
BoolQ [43]	Yes/no RC over passages, used to test contextual understanding and binary factuality.
DiaHalu [83]	Dialogue-level hallucination eval across four multi-turn domains, used for conversational robustness.
FACTOR [120]	Factuality with multi-hop evidence retrieval, used for claim verification over multiple sources.
FACTSCORE [106]	Atomic fact scoring with 6.5k generations, used to mark supported/unsupported/unverifiable units.
FELM [109]	Long-form factuality eval across five domains, used to benchmark factuality evaluators.
FEVER [110]	185k claims labeled Supported/Refuted/NEI, used for evidence retrieval + verification (often multi-hop).
FEWL [194]	Label-free hallucination measurement via self-consistency + external checks, used without gold answers.
FRANK [320]	Fine-grained error typology for summarization, used to compare factuality metrics on summaries.
HADES [128]	Reference-free QA hallucination dataset, model-in-the-loop; used when ground truth is unavailable.
HalluEditBench [46]	Verified hallucinations for model editing; scores efficacy, generalization, portability, locality, robustness.
HalluLens [139]	Separates hallucination from factuality with intrinsic/extrinsic tasks, dynamic sets to reduce leakage.
HALOGEN [141]	Nine-domain eval (code, attribution, history, reasoning, etc.), used for rate, refusals, and utility.
HaluEval [143]	Large-scale cross-task benchmark for hallucination tendency and factual accuracy.
HaluEval 2.0 [292]	8,770 harder questions with stronger metrics, used to capture subtle hallucinations and robustness.
HaluEval-Wild [142]	Adversarial real-user prompts (ShareGPT), used to test in-the-wild hallucinations.
HDMBench [138]	Span/sentence annotations for knowledge-intensive tasks, used for fine-grained detection eval.
Head-to-Tail [144]	18k QA by popularity (head/torso/tail), used to probe recall from common to rare knowledge.
HotpotQA [146]	Multi-hop QA with supporting paragraphs, used to assess cross-document reasoning.
Natural Questions (NQ) [215]	Real Google queries with Wikipedia answers, used for large-corpus retrieval + long-form QA.
RAGTruth [241]	Word-level hallucination labels in RAG, used to benchmark hallucination frequency in pipelines.
SelfCheckGPT [274]	Black-box self-consistency sampling, used to flag hallucinations without external DBs.
TriviaQA [309]	650k QA-evidence triplets, used for diverse-topic QA with frequent multi-hop needs.
TruthfulQA [315]	Adversarial questions targeting misconceptions, used to test truthfulness under lure-like claims.
UHGEval [318] Chinese	Large unconstrained-gen dataset with rigorous annotation, used for realistic hallucination analysis.

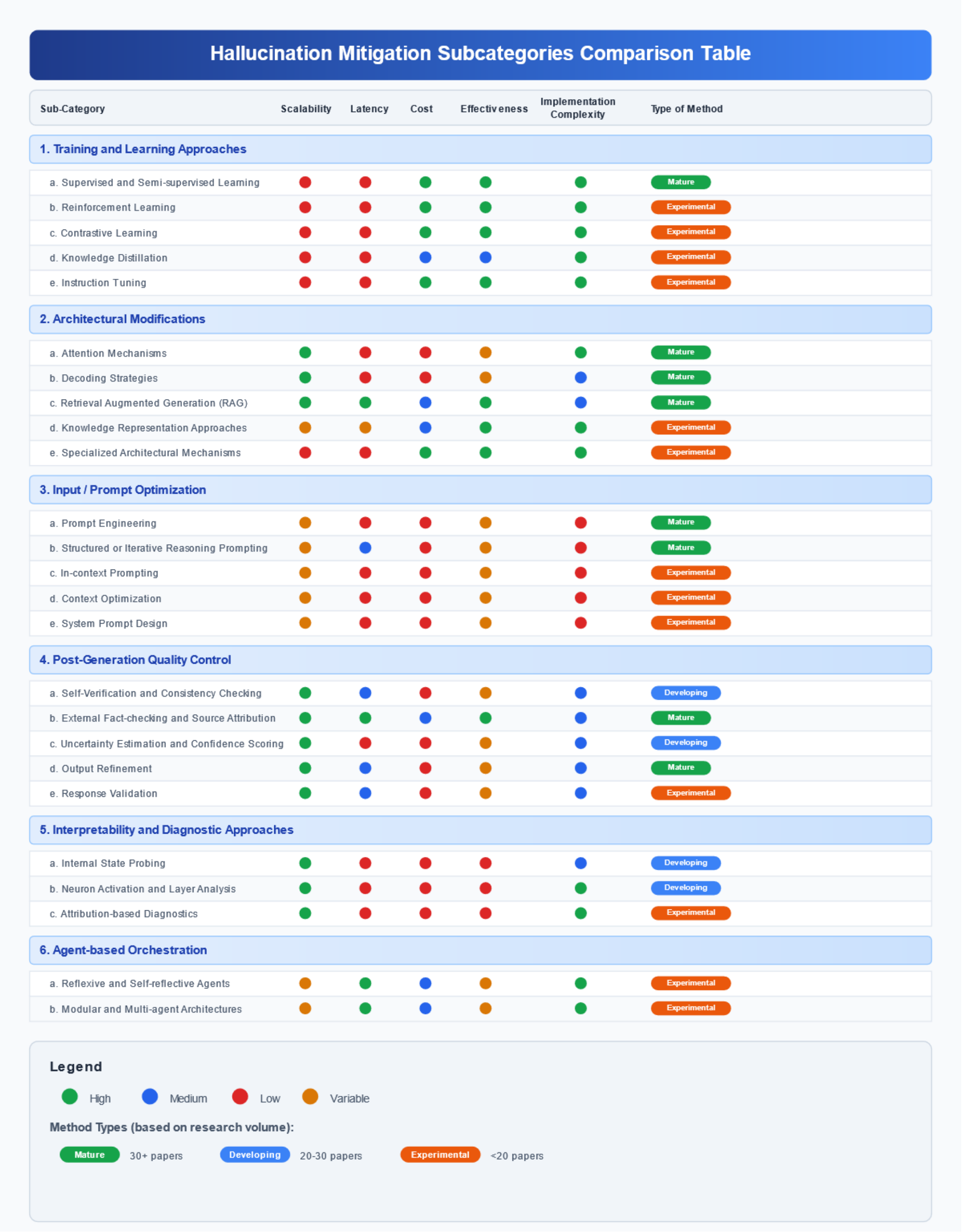
Domain-Specific Benchmarks Category ii

Benchmark	Brief description
PubMedQA [234] medical	Biomedical QA over literature, used to test domain factuality and clinical reasoning.
SciBench [265] science	Scientific reasoning and claim checks, used to assess principle application and consistency.
LegalBench [177] legal	Legal tasks for interpretation and reasoning, used to probe statutory/domain knowledge.

Code Generation Benchmarks Category iii

Benchmark	Brief Description
HumanEval / Codex [99]	Function-level coding tasks with unit tests, used to measure correctness, reasoning, and API recall.

Appendix C. Hallucination Mitigation Subcategories Comparison Table



Hallucination Mitigation Subcategories Comparison Table

Sub-Category	Scalability	Latency	Cost	Effectiveness	Implementation Complexity	Type of Method
1. Training and Learning Approaches						
a. Supervised and Semi-supervised Learning	🔴	🔴	🟢	🟢	🟢	Mature
b. Reinforcement Learning	🔴	🔴	🟢	🟢	🟢	Experimental
c. Contrastive Learning	🔴	🔴	🟢	🟢	🟢	Experimental
d. Knowledge Distillation	🔴	🔴	🔵	🔵	🟢	Experimental
e. Instruction Tuning	🔴	🔴	🟢	🟢	🟢	Experimental
2. Architectural Modifications						
a. Attention Mechanisms	🟢	🔴	🔴	🟠	🟢	Mature
b. Decoding Strategies	🟢	🔴	🔴	🟠	🔵	Mature
c. Retrieval Augmented Generation (RAG)	🟢	🟢	🔵	🟢	🔵	Mature
d. Knowledge Representation Approaches	🟠	🟠	🔵	🟢	🟢	Experimental
e. Specialized Architectural Mechanisms	🔴	🔴	🟢	🟢	🟢	Experimental
3. Input / Prompt Optimization						
a. Prompt Engineering	🟠	🔴	🔴	🟠	🔴	Mature
b. Structured or Iterative Reasoning Prompting	🟠	🔵	🔴	🟠	🔴	Mature
c. In-context Prompting	🟠	🔴	🔴	🟠	🔴	Experimental
d. Context Optimization	🟠	🔴	🔴	🟠	🔴	Experimental
e. System Prompt Design	🟠	🔴	🔴	🟠	🔴	Experimental
4. Post-Generation Quality Control						
a. Self-Verification and Consistency Checking	🟢	🔵	🔴	🟠	🔵	Developing
b. External Fact-checking and Source Attribution	🟢	🟢	🔵	🟢	🔵	Mature
c. Uncertainty Estimation and Confidence Scoring	🟢	🔴	🔴	🟠	🔵	Developing
d. Output Refinement	🟢	🔵	🔴	🟠	🔵	Mature
e. Response Validation	🟢	🔵	🔴	🟠	🔵	Experimental
5. Interpretability and Diagnostic Approaches						
a. Internal State Probing	🟢	🔴	🔴	🔴	🔵	Developing
b. Neuron Activation and Layer Analysis	🟢	🔴	🔴	🔴	🟢	Developing
c. Attribution-based Diagnostics	🟢	🔴	🔴	🔴	🟢	Experimental
6. Agent-based Orchestration						
a. Reflexive and Self-reflective Agents	🟠	🟢	🔵	🟠	🟢	Experimental
b. Modular and Multi-agent Architectures	🟠	🟢	🔵	🟠	🟢	Experimental

Legend

🟢 High 🔵 Medium 🔴 Low 🟠 Variable

Method Types (based on research volume):

Mature 30+ papers Developing 20-30 papers Experimental <20 papers

Glossary

A

ActLCD — Active Layer-Contrastive Decoding — Decoding that contrasts intermediate layers to steer token selection toward more factual continuations.

Activation Decoding — Constrained decoding that adjusts next-token probabilities using activation/uncertainty signals to suppress hallucinations.

AFHN — Adversarial Feature Hallucination Networks — Adversarial training to produce features and examples that stress models and reduce hallucinations.

AggTruth — Attention aggregation across heads/layers to flag unsupported spans and improve factual consistency checks.

ALIGNed-LLM — Aligns external knowledge (e.g., KG/entity embeddings) with the model's representation space to ground generations.

ALTI+ — Attribution method that quantifies how much each input token contributes to generated tokens for interpretability/factuality analysis.

ANAH — Bilingual hallucination dataset with sentence-level annotations and suggested corrections.

ATF — Adaptive Token Fusion — Merges redundant/similar tokens early to retain meaning while reducing noise and hallucination risk.

AutoHall — Automatic pipeline to synthesize, detect, and evaluate hallucinations for training and benchmarking.

AutoRAG-LoRA — Lightweight LoRA adaptation to better couple retrieval and generation in RAG systems.

B

BERTScore — Semantic similarity metric using contextual embeddings to evaluate generated text.

BLEU — N-gram overlap metric; useful for surface similarity but not a direct measure of factuality.

BoolQ — Yes/no question-answering dataset often used in factuality experiments.

C

CCL-XCoT — Cross-lingual transfer of Chain of Thought traces to improve reasoning and reduce hallucinations across languages.

CD — Contrastive Decoding — Penalizes tokens favored by a weaker/contrast model to filter implausible continuations.

Chain of Knowledge (CoK) — Grounds reasoning by explicitly incorporating external knowledge into intermediate steps.

Chain of NLI (CoNLI) — Cascaded entailment checks over partial outputs to prune unsupported content.

Chain of Thought (CoT) — Prompting that elicits step-by-step reasoning before the final answer.

Chain of Verification (CoVe) — Contextualized word embeddings pre-trained on translation, used for stronger semantic representations.

COMET-QE — Reference-free MT quality estimation used as a proxy signal for consistency.

Conditional Pointwise Mutual Information (CPMI) — Decoding re-scoring that rewards tokens better supported by the source/context.

Confident Decoding — Incorporates uncertainty estimates into beam/nucleus procedures to favor low-uncertainty continuations.

CPM — Conditional Entropy Mechanism — Uses token-level entropy to detect and avoid uncertain/hallucination-prone outputs.

CPO — Contrastive Preference Optimization — Preference optimization that uses contrastive signals to align outputs with faithful behavior.

CRAG — Corrective Retrieval-Augmented Generation — Adds corrective/revision steps atop RAG to fix unsupported claims.

CRITIC — A verify-and-edit framework where a "critic" process checks claims against evidence and proposes fixes.

Critic-driven Decoding — Decoding guided by a trained critic/verifier that down-weights unsupported next tokens.

D

D&Q — Decompose-and-Query — Decomposes a question into sub-questions and retrieves evidence for each before answering.
DeCoRe — Decoding by Contrasting Retrieval Heads — Contrasts retrieval-conditioned signals to suppress ungrounded tokens.
Dehallucinator — Detect-then-rewrite approach that edits hallucinated spans into grounded alternatives.
Delta — Compares outputs under masked vs. full context to detect and penalize hallucination-prone continuations.
DiaHalu — Dialogue-level hallucination benchmark covering multiple multi-turn domains.
DoLa — Decoding by Contrasting Layers — Uses differences between early vs. late layer logits to promote factual signals.
DPO — Direct Preference Optimization — RL-free preference tuning that directly optimizes for chosen responses.
DRAD — Decoding with Retrieval-Augmented Drafts — Uses retrieved drafts/evidence to guide decoding away from unsupported text.
DreamCatcher — Detects and corrects hallucinations by cross-checking outputs against external evidence/tools.
DrHall — Lightweight, fast hallucination detection targeted at real-time scenarios.

E

EigenScore — Uncertainty/factuality signal derived from the spectrum of hidden-state representations.
EntailR — Entailment-based verifier used to check whether generated claims follow from retrieved evidence.
EVER — Evidence-based verification/rectification that validates claims and proposes fixes during/after generation.

F

F2 — Faith Finetuning — Direct finetuning objective to increase faithfulness of generations.
Faithful Finetuning — Finetuning strategies that explicitly optimize for verifiable, source-supported outputs.
FacTool — Tool-augmented factuality checking that extracts claims and verifies them against sources.
FactPEGASUS — Summarization variant emphasizing factual consistency with the source document.
FactRAG — RAG design focused on retrieving and citing evidence that supports each claim.
FACTOR — Benchmark emphasizing multi-hop factuality and evidence aggregation.
FAVA — Corrupt-and-denoise training pipeline to teach models to correct fabricated content.
FELM — Benchmark for evaluating factuality evaluators on long-form outputs.
FEVER — Large-scale fact verification dataset (Supported/Refuted/Not Enough Info).
FG-PRM — Fine-Grained Process Reward Model — Process-level reward modeling for stepwise supervision of reasoning.
FRANK — Fine-grained factual error taxonomy and benchmark for summarization.
FreshLLMs — Uses live retrieval/search refresh to reduce outdated or stale knowledge.
FActScore / FACTSCORE — Atomic, claim-level factuality scoring/benchmark for long-form text.

G

GAN — Generative Adversarial Network — Adversarial training framework used to stress and correct model behaviors.
GAT — Graph Attention Network — Graph neural network with attention; used to propagate grounded evidence.
GNN — Graph Neural Network — Neural architectures over graphs for structured reasoning/grounding.
GoT — Graph-of-Thoughts — Represents reasoning as a graph of states/operations to explore multiple paths.
Grad-CAM — Gradient-based localization on intermediate features for interpretability of decisions.
Gradient × Input — Simple attribution method multiplying gradients by inputs to estimate token importance.
Graph-RAG — RAG that leverages knowledge graphs/graph structure for retrieval and grounding.
G-Retriever — Graph-aware retriever designed to recall evidence that reduces hallucinations.

H

HADES — Reference-free hallucination detection dataset for QA.
HALO — Estimation + reduction framework for hallucinations in open-source LLMs.
HALOGEN — Structure-aware reasoning/verification pipeline to reduce unsupported claims.

HalluciNot — Retrieval-assisted span verification to detect and mitigate hallucinations.
HaluBench — Benchmark suite for evaluating hallucinations across tasks or RAG settings.
HaluEval — Large-scale hallucination evaluation benchmark.
HaluEval-Wild — "In-the-wild" hallucination evaluation using web-scale references.
HaluSearch — Retrieval-in-the-loop detection/mitigation pipeline that searches evidence while generating.
HAR — Hallucination Augmented Recitations — Produces recitations/snippets that anchor generation to evidence.
HDM-2 — Hallucination Detection Method 2 — Modular multi-detector system targeting specific hallucination types.
HERMAN — Checks entities/quantities in outputs against source to avoid numerical/entity errors.
HILL — Human-factors-oriented hallucination identification framework/benchmark.
Hidden Markov Chains — Sequential state models used to analyze latent dynamics associated with hallucinations.
HIPO — Hard-sample-aware iterative preference optimization to improve robustness.
HSP — Hierarchical Semantic Piece — Hierarchical text segmentation/representation to stabilize retrieval and grounding.
HybridRAG — Combines multiple retrieval sources/strategies (e.g., dense + sparse + KG) for stronger grounding.
HumanEval — Code generation benchmark often used in hallucination-sensitive program synthesis.
HVM — Hypothesis Verification Model — Classifier/verifier that filters candidates by textual entailment with evidence.

I

ICD — Induce-then-Contrast Decoding — Induces errors with a weaker model and contrasts to discourage hallucinated tokens.
INSIDE — Internal-state-based uncertainty estimation with interventions to reduce overconfidence.
Input Erasure — Attribution by removing/ablating input spans to see their effect on outputs.
InterrogateLLM — Detects hallucinations via inconsistency across multiple answers/contexts.
Iter-AHMCL — Iterative decoding with hallucination-aware contrastive learning to refine outputs.
ITI — Inference-Time Intervention — Nudges specific heads/activations along truth-aligned directions during decoding.

J

Joint Entity and Summary Generation — Summarization that jointly predicts entities and the abstract to reduce unsupported content.

K

KB — Knowledge Base — External repository of facts used for grounding/verification.
KCA — Knowledge-Consistent Alignment — Aligns model outputs with retrieved knowledge via structured prompting/objectives.
KG — Knowledge Graph — Graph-structured facts used for retrieval, verification, and attribution.
KGR — Knowledge Graph Retrofitting — Injects/retrofits KG-verified facts into outputs or intermediate representations.
KL-divergence — Divergence measure used in calibration/regularization and to compare layer distributions.
Knowledge Overshadowing — When parametric priors dominate over context, causing the model to ignore given evidence.

L

LaBSE — Multilingual sentence encoder used for cross-lingual matching/verification.
LASER — Language-agnostic sentence embeddings for multilingual retrieval/entailment.
LAT — Linear Artificial Tomography — Linear probes/edits to reveal and steer latent concept directions.
LayerSkip — Self-speculative decoding with early exits/verification by later layers.
LID — Local Intrinsic Dimension — Dimensionality measure of hidden states linked to uncertainty/truthfulness.
LinkQ — Forces explicit knowledge-graph queries to ground answers.
LLM — Large Language Model — Transformer-based model trained for next-token prediction and generation.

LLM Factoscope — Probing/visualization of hidden-state clusters to distinguish factual vs fabricated content.
LLM-AUGMENTER — Orchestrates retrieval/tools around an LLM to improve grounding and reduce errors.
Logit Lens — Projects intermediate residual streams to the vocabulary space to inspect token preferences.
Lookback Lens — Attention-only method that checks whether outputs attend to relevant context.
LoRA — Low-rank adapters for efficient finetuning, commonly used in factuality/hallucination pipelines.
LQC — Lightweight Query Checkpoint — Predicts when a query needs verification or retrieval before answering.
LRP — Layer-wise Relevance Propagation — Decomposes predictions to attribute token-level contributions.

M

MARL — Multi-Agent Reinforcement Learning — Multiple agents coordinate/critique each other to improve reliability.
MC — Monte Carlo — Stochastic sampling used for uncertainty estimation and search.
MCTS — Monte Carlo Tree Search — Guided tree exploration used in deliberate, plan-and-verify reasoning.
METEOR — MT metric leveraging synonymy/stemming; not a direct factuality measure.
mFACT — Decoding-integrated factuality signal to prune low-faithfulness candidates.
MixCL — Mixed contrastive learning (with hard negatives) to reduce dialog hallucinations.
MoCo — Momentum contrast representation learning used to build stronger encoders.
MoE — Mixture-of-Experts — Sparse expert routing to localize knowledge and reduce interference.

N

NEER — Neural evidence-based evaluation/repair methods that use entailment or retrieved evidence to improve outputs.
Neural Path Hunter — Analyzes reasoning paths/graphs to locate error-prone segments for correction.
Neural-retrieval-in-the-loop — Integrates a trainable retriever during inference to stabilize grounding.
NL-ITI — Non-linear version of ITI with richer probes and multi-token interventions.
NLU — Natural Language Understanding — Models/components (e.g., NLI, QA) used as verifiers or critics.
Nucleus Sampling — Top-p decoding that samples from the smallest set whose cumulative probability exceeds p.

O

OVON — Open-Vocabulary Object Navigation; task setting where language directs navigation to open-set objects, used in agent/LLM evaluations.

P

PCA — Principal Component Analysis — Projects activations to principal subspaces to analyze truth/lie separability.
PGFES — Psychology-guided two-stage editing and sampling along “truthfulness” directions in latent space.
Persona drift — When a model’s stated persona/stance shifts across sessions or contexts.
PoLLMgraph — Probabilistic/graph model over latent states to track hallucination dynamics.
PMI — Pointwise Mutual Information — Signal for overshadowing/low-confidence conditions during decoding.
Principle Engraving — Representation-editing to imprint desired principles into activations.
Principle-Driven Self-Alignment — Self-alignment method that derives rules/principles and tunes behavior accordingly.
ProbTree — Probabilistic Tree-of-Thought — ToT reasoning with probabilistic selection/evaluation of branches.
PURR — Trains on corrupted vs. corrected claims to produce a compact, factuality-aware model.
TOPICPREFIX — Prompt/prefix-tuning scheme to stabilize topic adherence and reduce drift.

Q

Q^2 — Factual consistency measure comparing outputs to retrieved references.

R

R-Tuning — Instructioning/tuning models to abstain or say “I don’t know” when unsure.
RAG — Retrieval-Augmented Generation — Augments generation with document retrieval for grounding.
RAG-KG-IL — RAG integrated with knowledge-graph and incremental-learning components.
RAG-Turn — Turn-aware retrieval for multi-turn tasks.
RAGTruth — Human-annotated data for evaluating/teaching RAG factuality.

RAP — Reasoning viA Planning — Planning-style reasoning that structures problem solving before answering.
RARR — Retrieve-and-Revise pipeline that edits outputs to add citations and fix unsupported claims.
RBG — Read-Before-Generate — Reads/retrieves first, then conditions generation on the evidence.
REPLUG — Prepends retrieved text and averages probabilities across retrieval passes to ground decoding.
RepE — Representation Engineering — Editing/steering latent directions to improve honesty/faithfulness.
RefChecker — Reference-based fine-grained hallucination checker and diagnostic benchmark.
Reflexion — Self-critique loop where the model reflects on errors and retries.
RID — Retrieval-In-Decoder — Retrieval integrated directly into the decoder loop.
RHO — Reranks candidates by factual consistency with retrieved knowledge or graph evidence.
RHD — Real-time Hallucination Detection — Online detection and optional self-correction during generation.
RLCD — Reinforcement Learning with Contrastive Decoding — RL variant that pairs contrastive objectives with decoding.
RLHF — Reinforcement Learning from Human Feedback — Uses human preference signals to align model behavior.
RLAIF — Reinforcement Learning from AI Feedback — Uses AI-generated preference signals to scale alignment.
RLKF — Reinforcement-Learning-based Knowledge Filtering that favors context-grounded generation.
ROUGE — Overlap-based summarization metric (e.g., ROUGE-L).
RaLFiT — Reinforcement-learning-style fine-tuning aimed at improving truthfulness/factuality.

S

SC2 — Structured Comparative Reasoning — Compares structured alternatives and selects the most consistent one.
SCOTT — Self-Consistent Chain-of-Thought Distillation — Samples multiple CoTs and distills the consistent answer.
SCD — Self-Contrastive Decoding — Penalizes over-represented priors to counter knowledge overshadowing.
SEA — Spectral Editing of Activations — Projects activations along truth-aligned directions while suppressing misleading ones.
SEAL — Selective Abstention Learning — Teaches models to abstain (e.g., emit a reject token) when uncertain.
SEBRAG — Structured Evidence-Based RAG — RAG variant that structures evidence and grounding steps.
SEK — Evidence selection/structuring module used to verify or revise outputs.
SEPs — Semantic Entropy Probes — Fast probes that estimate uncertainty from hidden states.
Self-Checker — Pipeline that extracts and verifies claims using tools or retrieval.
Self-Checks — Generic self-verification passes (consistency checks, regeneration, or critique).
Self-Consistency — Samples multiple reasoning paths and selects the majority-consistent result.
Self-Familiarity — Calibrates outputs based on what the model “knows it knows” vs. uncertain areas.
Self-Refine — Iterative refine-and-feedback loop where the model improves its own draft.
Self-Reflection — The model reflects on its reasoning and revises responses accordingly.
SELF-RAG — Self-reflective RAG where a critic guides retrieval and edits drafts.
SelfCheckGPT — Consistency-based hallucination detector using multiple sampled outputs.
SH2 — Self-Highlighted Hesitation — Injects hesitation/abstention mechanisms at uncertain steps.
SimCLR — Contrastive representation learning framework used to build stronger encoders.
SimCTG — Contrastive text generation that constrains decoding to avoid degenerate outputs.
Singular Value Decomposition (SVD) — Matrix factorization used to analyze or edit latent directions.
Socratic Prompting — Uses guided questions to elicit intermediate reasoning and evidence.

T

ToT — Tree-of-Thought — Branch-and-evaluate reasoning over a tree of intermediate states.
TOPICPREFIX — Prompt/prefix-tuning that encodes topics to stabilize context adherence.
TrueTeacher — Teacher-style training that builds a factual evaluator and uses it to guide student outputs.
Truth Forest — Learns orthogonal “truth” representations and intervenes along those directions.
TruthfulQA — Benchmark evaluating resistance to common falsehoods.
TruthX — Latent editing method that nudges activations toward truthful directions.
Tuned Lens — Learns linear mappings from hidden states to logits to study/steer layer-wise predictions.

TWEAK — Think While Effectively Articulating Knowledge — Hypothesis-and-NLI-guided reranking that prefers supported continuations.

U

UHGEval — Hallucination evaluation benchmark for unconstrained generation in Chinese and related settings.

UPRISE — Uses LLM signals to train a retriever that selects stronger prompts/evidence.

V

Verbose Cloning — Prompting/aggregation technique that elicits explicit, fully-specified answers to reduce ambiguity.

X

XCoT — Cross-lingual Chain-of-Thought prompting/transfer.

XNLI — Cross-lingual NLI benchmark commonly used for entailment-based verification.

References

1. S. M. T. I. Tonmoy et al., “A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.01313
2. Y. Su, T. Lan, Y. Wang, D. Yogatama, L. Kong, and N. Collier, “A Contrastive Framework for Neural Text Generation,” Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.06417
3. T. R. McIntosh et al., “A Culturally Sensitive Test to Evaluate Nuanced GPT Hallucination,” IEEE Access, vol. 12, pp. 51555–51572, 2024, doi: 10.1109/ACCESS.2024.3386138.
4. A. Shelmanov et al., “A Head to Predict and a Head to Question: Pre-trained Uncertainty Quantification Heads for Hallucination Detection in LLM Outputs,” May 2025, [Online]. Available: http://arxiv.org/abs/2505.08200
5. J. White et al., “A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT,” Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.11382
6. Z. Yin, “A review of methods for alleviating hallucination issues in large language models,” Applied and Computational Engineering, vol. 76, no. 1, pp. 258–266, Jul. 2024, doi: 10.54254/2755-2721/76/20240608.
7. B. AlKhamissi, M. Li, A. Celikyilmaz, M. Diab, and M. Ghazvininejad, “A Review on Language Models as Knowledge Bases,” Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.06031
8. T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.05709
9. F. Nie, J.-G. Yao, J. Wang, R. Pan, and C.-Y. Lin, “A Simple Recipe towards Reducing Hallucination in Neural Surface Realisation,” Association for Computational Linguistics, 2019. [Online]. Available: https://aclanthology.org/P19-1256.pdf
10. H. Cao, Z. An, J. Feng, K. Xu, L. Chen, and D. Zhao, “A Step Closer to Comprehensive Answers: Constrained Multi-Stage Question Decomposition with Large Language Models,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.07491
11. Q. Signé, M. Boughanem, J. G. Moreno, and T. Belkacem, “A Substring Extraction-Based RAG Method for Minimising Hallucinations in Aircraft Maintenance Question Answering,” in Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR), New York, NY, USA: ACM, Jul. 2025, pp. 513–521. doi: 10.1145/3731120.3744624.
12. V. Rawte, A. Sheth, and A. Das, “A Survey of Hallucination in Large Foundation Models,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.05922
13. L. Huang et al., “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” ACM Transactions on Information Systems, Nov. 2024, doi: 10.1145/3703155.
14. S. v. Shah, “Accuracy, Consistency, and Hallucination of Large Language Models When Analyzing Unstructured Clinical Notes in Electronic Medical Records,” JAMA Network Open. American Medical Association, p. e2425953, 2024. doi: 10.1001/jamanetworkopen.2024.25953.
15. H. Zhang, H. Chen, M. Chen, and T. Zhang, “Active Layer-Contrastive Decoding Reduces Hallucination in Large Language Model Generation,” Jun. 2025, [Online]. Available: http://arxiv.org/abs/2505.23657
16. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive Mixtures of Local Experts,” 1991. Accessed: Aug. 07, 2025. [Online]. Available: https://2024.sci-hub.se/1867/e922caa86bf169b2dbb314f150dbdadb/jacobs1991.pdf
17. K. Li, Y. Zhang, K. Li, and Y. Fu, “Adversarial Feature Hallucination Networks for Few-Shot Learning,” 2020. [Online]. Available: http://arxiv.org/abs/2003.13193
18. P. Matys et al., “AggTruth: Contextual Hallucination Detection using Aggregated Attention Scores in LLMs,” Jun. 2025, doi: 10.1007/978-3-031-97570-7_18.

19. S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir, and A. Narayanan, “AI Agents That Matter,” Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.01502
20. J. Ji et al., “AI Alignment: A Comprehensive Survey,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.19852
21. N. Maleki, B. Padmanabhan, and K. Dutta, “AI Hallucinations: A Misnomer Worth Clarifying,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.06796
22. N. A. Z. Nishat, A. Coletta, L. Bellomarini, K. Amouzouvi, J. Lehmann, and S. Vahdati, “Aligning Knowledge Graphs and Language Models for Factual Accuracy,” Jul. 2025, [Online]. Available: http://arxiv.org/abs/2507.13411
23. Y. Yang, E. Chern, X. Qiu, G. Neubig, and P. Liu, “Alignment for Honesty,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2312.07000
24. L. Huang et al., “Alleviating Hallucinations from Knowledge Misalignment in Large Language Models via Selective Abstention Learning,” 2025. Accessed: Aug. 04, 2025. [Online]. Available: https://aclanthology.org/2025.acl-long.1199.pdf
25. J. Li, Z. Mao, and Q. Wang, “Alleviating Hallucinations in Large Language Models via Truthfulness-driven Rank-adaptive LoRA,” Jul. 2025. Accessed: Aug. 04, 2025. [Online]. Available: https://aclanthology.org/2025.findings-acl.103.pdf
26. Y. Zhang, L. Cui, W. Bi, and S. Shi, “Alleviating Hallucinations of Large Language Models through Induced Hallucinations,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.15710
27. Y. Li, Z. Li, K. Hung, W. Wang, H. Xie, and Y. Li, “Ambiguity processing in Large Language Models: Detection, resolution, and the path to hallucination,” Natural Language Processing Journal, p. 100173, Jul. 2025, doi: 10.1016/j.nlp.2025.100173.
28. Z. Ji, Y. Gu, W. Zhang, C. Lyu, D. Lin, and K. Chen, “ANAH: Analytical Annotation of Hallucinations in Large Language Models,” May 2024, Available: http://arxiv.org/abs/2405.20315
29. S. Ramprasad, E. Ferracane, and Z. C. Lipton, “Analyzing LLM Behavior in Dialogue Summarization: Unveiling Circumstantial Hallucination Trends,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.03487
30. A. Chaturvedi, S. Bhar, S. Saha, U. Garain, and N. Asher, “Analyzing Semantic Faithfulness of Language Models via Input Intervention on Question Answering under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license,” Computational Linguistics, vol. 50, no. 1, 2024, doi: 10.1162/coli.
31. P. Michel, O. Levy, and G. Neubig, “Are Sixteen Heads Really Better than One?” Nov. 2019, [Online]. Available: http://arxiv.org/abs/1905.10650
32. P. Zablocki and Z. Gajewska, “Assessing Hallucination Risks in Large Language Models Through Internal State Analysis.” Jul. 17, 2024. doi: 10.22541/au.172124175.55788724/v1.
33. B. Goodrich et al., “Assessing the Factual Accuracy of Generated Text,” KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Aug. 2019, doi: 10.1145/3292500.3330955.
34. A. Vaswani et al., “Attention Is All You Need,” Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762
35. M. Yuksekgonul et al., “Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.15098
36. Y. Zhao, Z. Liu, Y. Zheng, and K.-Y. Lam, “Attribution Techniques for Mitigating Hallucination in RAG-based Question-Answering Systems: A Survey.” Jun. 19, 2025. doi: 10.36227/techrxiv.175036754.46793269/v1.
37. Z. Cao, Y. Yang, and H. Zhao, “AutoHall: Automated Hallucination Dataset Generation for Large Language Models,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2310.00259
38. K. Dwivedi and P. P. Mishra, “AutoRAG-LoRA: Hallucination-Triggered Knowledge Retuning via Lightweight Adapters,” Jul. 2025, [Online]. Available: http://arxiv.org/abs/2507.10586
39. J. Li et al., “Banishing LLM Hallucinations Requires Rethinking Generalization,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.17642
40. J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking Large Language Models in Retrieval-Augmented Generation,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.01431
41. T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.09675
42. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A Method for Automatic Evaluation of Machine Translation.” in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, USA, Jul. 2002, pp. 311–318. doi: 10.3115/1073083.1073135. https://aclanthology.org/P02-1040.pdf

43. C. Clark et al., "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions." [Online]. Available: https://arxiv.org/abs/1905.10044
44. T. Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate Before Use: Improving Few-Shot Performance of Language Models," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2102.09690
45. A. T. Kalai and S. S. Vempala, "Calibrated Language Models Must Hallucinate," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.14648
46. B. Huang, C. Chen, X. Xu, A. Payani, and K. Shu, "Can Knowledge Editing Really Correct Hallucinations?" Mar. 2025, [Online]. Available: http://arxiv.org/abs/2410.16251
47. G. Agrawal, T. Kumarage, Z. Alghamdi, and H. Liu, "Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2311.07914
48. M. Xiong et al., "Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.13063
49. E. Kıcıman, R. Ness, A. Sharma, and C. Tan, "Causal Reasoning and Large Language Models: Opening a New Frontier for Causality," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2305.00050
50. W. Zheng, R. K.-W. Lee, Z. Liu, K. Wu, A. Aw, and B. Zou, "CCL-XCoT: An Efficient Cross-Lingual Knowledge Transfer Method for Mitigating Hallucination Generation," Jul. 2025, [Online]. Available: http://arxiv.org/abs/2507.14239
51. D. Lei et al., "Chain of Natural Language Inference for Reducing Large Language Model Ungrounded Hallucinations," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.03951
52. X. Li et al., "Chain-of-Knowledge: Grounding Large Language Models via Dynamic Knowledge Adapting over Heterogeneous Sources," May 2023, [Online]. Available: http://arxiv.org/abs/2305.13269
53. J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 2022, [Online]. Available: http://arxiv.org/abs/2201.11903
54. J. Cheng et al., "Chain-of-Thought Prompting Obscures Hallucination Cues in Large Language Models: An Empirical Evaluation," Jun. 2025, [Online]. Available: http://arxiv.org/abs/2506.17088
55. S. Dhuliawala et al., "Chain-of-Verification Reduces Hallucination in Large Language Models," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.11495
56. J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and Applications of Large Language Models," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.10169
57. F. Yin, J. Srinivasa, and K.-W. Chang, "Characterizing Truthfulness in Large Language Model Generations with Local Intrinsic Dimension," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.18048
58. B. Peng et al., "Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.12813
59. Q. Lv, J. Wang, H. Chen, B. Li, Y. Zhang, and F. Wu, "Coarse-to-Fine Highlighting: Reducing Knowledge Hallucination in Large Language Models," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.15116
60. V. Agarwal, Y. Pei, S. Alamir, and X. Liu, "CodeMirage: Hallucinations in Code Generated by Large Language Models," Aug. 2024, [Online]. Available: http://arxiv.org/abs/2408.08333
61. K. Thórisson and H. Helgasson, "Cognitive Architectures and Autonomy: A Comparative Review," Journal of Artificial General Intelligence, vol. 3, no. 2, pp. 1–30, May 2012, doi: 10.2478/v10229-011-0015-3.
62. H. Ye, T. Liu, A. Zhang, W. Hua, and W. Jia, "Cognitive Mirage: A Review of Hallucinations in Large Language Models," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.06794
63. Y. Bai et al., "Constitutional AI: Harmlessness from AI Feedback," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.08073
64. X. L. Li et al., "Contrastive Decoding: Open-ended Text Generation as Optimization," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.15097
65. W. Sun, Z. Shi, S. Gao, P. Ren, M. de Rijke, and Z. Ren, "Contrastive Learning Reduces Hallucination in Conversations," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.10400
66. Z. Xu, "Context-aware Decoding Reduces Hallucination in Query-focused Summarization," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.14335
67. J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive Learning with Hard Negative Samples," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.04592
68. K. Filippova, "Controlled Hallucinations: Learning to Generate Faithfully from Noisy Data," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.05873
69. S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective Retrieval Augmented Generation," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.15884
70. Z. Gou et al., "CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing," May 2023, [Online]. Available: http://arxiv.org/abs/2305.11738
71. M. Lango and O. Dušek, "Critic-Driven Decoding for Mitigating Hallucinations in Data-to-text Generation," 2023. [Online]. Available: https://arxiv.org/abs/2310.16964

72. A. P. Gema et al., “DeCoRe: Decoding by Contrasting Retrieval Heads to Mitigate Hallucinations,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.18860
73. K. Lee et al., “Deduplicating Training Data Makes Language Models Better,” Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.06499
74. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03741
75. S. Jha, S. K. Jha, P. Lincoln, N. D. Bastian, A. Velasquez, and S. Neema, “Dehallucinating Large Language Models Using Formal Methods Guided Iterative Prompting,” in Proceedings - 2023 IEEE International Conference on Assured Autonomy, ICAA 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 149–152. doi: 10.1109/ICAA58325.2023.00029.
76. C. P. Huang and H.-Y. Chen, “Delta -- Contrastive Decoding Mitigates Text Hallucinations in Large Language Models,” Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.05825
77. V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” 2020. [Online]. Available: https://arxiv.org/abs/2004.04906
78. D. Dale, E. Voita, L. Barrault, and M. R. Costa-jussà, “Detecting and Mitigating Hallucinations in Machine Translation: Model Internal Workings Alone Do Well, Sentence Similarity Even Better,” Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.08597
79. Y. Qiu, Y. Ziser, A. Korhonen, E. M. Ponti, and S. B. Cohen, “Detecting and Mitigating Hallucinations in Multilingual Summarisation,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.13632
80. W. Wu, Y. Cao, N. Yi, R. Ou, and Z. Zheng, “Detecting and Reducing the Factual Hallucinations of Large Language Models with Metamorphic Testing,” Proceedings of the ACM on Software Engineering, vol. 2, no. FSE, pp. 1432 – 1453, 2025, [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3715784
81. C. Zhou et al., “Detecting Hallucinated Content in Conditional Neural Sequence Generation,” 2021. [Online]. Available: https://arxiv.org/abs/2011.02593
82. S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, “Detecting hallucinations in large language models using semantic entropy,” Nature, vol. 630, no. 8017, pp. 625–630, Jun. 2024, doi: 10.1038/s41586-024-07421-0.
83. K. Chen, Q. Chen, J. Zhou, Y. He, and L. He, “DiaHalu: A Dialogue-level Hallucination Evaluation Benchmark for Large Language Models,” Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.00896
84. E. Razumovskaia et al., “Dial BEINFO for Faithfulness: Improving Factuality of Information-Seeking Dialogue via Behavioural Fine-Tuning,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.09800
85. E. Perez et al., “Discovering Language Model Behaviors with Model-Written Evaluations,” Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.09251
86. S. CH-Wang, B. van Durme, J. Eisner, and C. Kedzie, “Do Androids Know They’re Only Dreaming of Electric Sheep?” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.17249
87. A. Agrawal, M. Suzgun, L. Mackey, and A. T. Kalai, “Do Language Models Know When They’re Hallucinating References?” May 2023, [Online]. Available: http://arxiv.org/abs/2305.18248
88. Z. Yin, Q. Sun, Q. Guo, J. Wu, X. Qiu, and X. Huang, “Do Large Language Models Know What They Don’t Know?” 2023. [Online]. Available: https://github.com/yinzhangyue/SelfAware
89. Z. Gekhman et al., “Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?” May 2024, [Online]. Available: http://arxiv.org/abs/2405.05904
90. Y.-S. Chuang, Y. Xie, H. Luo, Y. Kim, J. Glass, and P. He, “DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models,” Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.03883
91. N. Li, Y. Li, Y. Liu, L. Shi, K. Wang, and H. Wang, “Drowzee: Metamorphic Testing for Fact-Conflicting Hallucination Detection in Large Language Models,” May 2024, doi: 10.1145/3689776.
92. J. Wei et al., “Emergent Abilities of Large Language Models,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.07682
93. A. M. Garcia-Carmona, M.-L. Prieto, E. Puertas, and J.-J. Beunza, “Enhanced medical data extraction: leveraging LLMs for accurate retrieval of patient information from medical reports,” JMIR AI, Nov. 2024, doi: 10.2196/68776.
94. C. Wang, Y. Zhao, Y. Liu, and H. Zhu, “Enhancing Latent Diffusion in Large Language Models for High-Quality Implicit Neural Representations with Reduced Hallucinations,” 2024. Accessed: Jun. 29, 2025. [Online]. Available: https://osf.io/preprints/osf/9utwy_v1
95. S. Behore, L. Dumont, and J. Venkataraman, “Enhancing Reliability in Large Language Models: Self-Detection of Hallucinations With Spontaneous Self-Checks.” Sep. 09, 2024. doi: 10.22541/au.172591474.45065639/v1
96. T. Zhang et al., “Enhancing Uncertainty-Based Hallucination Detection with Stronger Focus,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.13230
97. S. Longpre et al., “Entity-Based Knowledge Conflicts in Question Answering,” 2021. Accessed: Aug. 03, 2025. [Online]. Available: https://arxiv.org/abs/2109.05052

98. F. Nan et al., “Entity-level Factual Consistency of Abstractive Text Summarization,” Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.09130
99. M. Chen et al., “Evaluating Large Language Models Trained on Code,” Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03374
100. Q. Cheng et al., “Evaluating Hallucinations in Chinese Large Language Models,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.03368
101. W. Kryściński, B. McCann, C. Xiong, and R. Socher, “Evaluating the Factual Consistency of Abstractive Text Summarization,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.12840
102. H. Kang, J. Ni, and H. Yao, “Ever: Mitigating Hallucination in Large Language Models through Real-Time Verification and Rectification,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.09114
103. F. Liu et al., “Exploring and Evaluating Hallucinations in LLM-Powered Code Generation,” Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.00971
104. I.-C. Chern et al., “FacTool: Factuality Detection in Generative AI -- A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.13528
105. D. Wan and M. Bansal, “FactPEGASUS: Factuality-Aware Pre-training and Fine-tuning for Abstractive Summarization,” May 2022, [Online]. Available: http://arxiv.org/abs/2205.07830
106. S. Min et al., “FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.14251
107. P. Roit et al., “Factually Consistent Summarization via Reinforcement Learning with Textual Entailment Feedback,” May 2023, [Online]. Available: http://arxiv.org/abs/2306.00186
108. N. Lee et al., “Factuality Enhanced Language Models for Open-Ended Text Generation,” Jun. 2022, [Online]. Available: http://arxiv.org/abs/2206.04624
109. S. Chen et al., “FELM: Benchmarking Factuality Evaluation of Large Language Models,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2310.00741
110. J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “FEVER: a large-scale dataset for Fact Extraction and VERification,” Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.05355
111. R. Li, Z. Luo, and X. Du, “FG-PRM: Fine-grained Hallucination Detection and Mitigation in Language Model Mathematical Reasoning,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.06304
112. A. Mishra et al., “Fine-grained Hallucination Detection and Editing for Language Models,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.06855
113. K. Tian, E. Mitchell, H. Yao, C. D. Manning, and C. Finn, “Fine-tuning Language Models for Factuality,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.08401
114. Y. Hu, L. Gan, W. Xiao, K. Kuang, and F. Wu, “Fine-tuning Large Language Models for Improving Factuality in Legal Question Answering,” Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.06521
115. S.-C. Lin et al., “FLAME: Factuality-Aware Alignment for Large Language Models,” May 2024, [Online]. Available: http://arxiv.org/abs/2405.01525
116. F. F. Bayat et al., “FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.17119
117. T. Vu et al., “FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.03214
118. F. Leiser, S. Eckhardt, M. Knaeble, A. Maedche, G. Schwabe, and A. Sunyaev, “From ChatGPT to FactGPT: A Participatory Design Study to Mitigate the Effects of Large Language Model Hallucinations on Users,” in ACM International Conference Proceeding Series, Association for Computing Machinery, Sep. 2023, pp. 81–90. doi: 10.1145/3603555.3603565.
119. X. He et al., “G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering,” May 2024, [Online]. Available: http://arxiv.org/abs/2402.07630
120. D. Muhlgay et al., “Generating Benchmarks for Factuality Evaluation of Language Models,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.06908
121. S. Welleck et al., “Generating Sequences by Learning to Self-Correct,” Oct. 2022, [Online]. Available: http://arxiv.org/abs/2211.00053
122. OpenAI et al., “GPT-4 Technical Report,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.08774
123. M. Besta et al., “Graph of Thoughts: Solving Elaborate Problems with Large Language Models,” Aug. 2023, doi: 10.1609/aaai.v38i16.29720.
124. S. Sherif, D. Saad, S. Silva, and V. Gomes, “Graph-Enhanced RAG: A Survey of Methods, Architectures, and Performance,” 2025. [Online]. Available: https://www.researchgate.net/publication/393193258
125. M. Barry et al., “GraphRAG: Leveraging Graph-Based Efficiency to Minimize Hallucinations in LLM-Driven RAG for Finance Data,” 2025. [Online]. Available: https://aclanthology.org/2025.genaik-1.6.pdf
126. A. Köksal, R. Aksitov, and C.-C. Chang, “Hallucination Augmented Recitations for Language Models,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.07424

127. Y. Chen et al., "Hallucination Detection: Robustly Discerning Reliable Answers in Large Language Models," in International Conference on Information and Knowledge Management, Proceedings, Association for Computing Machinery, Oct. 2023, pp. 245–255. Available: https://arxiv.org/pdf/2407.04121
128. J. Luo, T. Li, D. Wu, M. Jenkin, S. Liu, and G. Dudek, "Hallucination Detection and Hallucination Mitigation: An Investigation," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.08358
129. Y. Xia et al., "Hallucination Diversity-Aware Active Learning for Text Summarization," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.01588
130. Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is Inevitable: An Innate Limitation of Large Language Models," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.11817
131. W. Zhang and J. Zhang, "Hallucination Mitigation for Retrieval-Augmented Large Language Models: A Review," Mathematics, vol. 13, no. 5. Multidisciplinary Digital Publishing Institute (MDPI), Mar. 01, 2025. doi: 10.3390/math13050856.
132. D. Gosmar and D. A. Dahl, "Hallucination Mitigation using Agentic AI Natural Language-Based Frameworks," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.13946
133. Z. Bai et al., "Hallucination of Multimodal Large Language Models: A Survey," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.18930
134. T. Rehman, R. Mandal, A. Agarwal, and D. K. Sanyal, "Hallucination Reduction in Long Input Text Summarization," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.16781
135. V. Magesh, F. Surani, M. Dahl, M. Suzgun, C. D. Manning, and D. E. Ho, "Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools," May 2024, [Online]. Available: http://arxiv.org/abs/2405.20362
136. G. P. Reddy, Y. v. Pavan Kumar, and K. P. Prakash, "Hallucinations in Large Language Models (LLMs)," in 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences, eStream 2024 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/eStream61684.2024.10542617.
137. G. Perković, A. Drobnjak, and I. Botički, "Hallucinations in LLMs: Understanding and Addressing Challenges," in 2024 47th ICT and Electronics Convention, MIPRO 2024 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 2084–2088. doi: 10.1109/MIPRO60963.2024.10569238.
138. B. Paudel, A. Lyzhov, P. Joshi, and P. Anand, "HalluciNot: Hallucination Detection Through Context and Common Knowledge Verification," Apr. 2025, [Online]. Available: http://arxiv.org/abs/2504.07069
139. Y. Bang et al., "HalluLens: LLM Hallucination Benchmark," Apr. 2025, [Online]. Available: http://arxiv.org/abs/2504.17550
140. M. Elaraby et al., "Halo: Estimation and Reduction of Hallucinations in Open-Source Weak Large Language Models," Aug. 2023, [Online]. Available: http://arxiv.org/abs/2308.11764
141. A. Ravichander, S. Ghela, D. Wadden, and Y. Choi, "HALoGEN: Fantastic LLM Hallucinations and Where to Find Them," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.08292
142. Z. Zhu, Y. Yang, and Z. Sun, "HaluEval-Wild: Evaluating Hallucinations of Language Models in the Wild," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.04307
143. J. Li, X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models," May 2023, [Online]. Available: http://arxiv.org/abs/2305.11747
144. K. Sun, Y. E. Xu, H. Zha, Y. Liu, and X. L. Dong, "Head-to-Tail: How Knowledgeable are Large Language Models (LLMs)? A.K.A. Will LLMs Replace Knowledge Graphs?", Aug. 2023, [Online]. Available: http://arxiv.org/abs/2308.10168
145. F. Leiser et al., "HILL: A Hallucination Identifier for Large Language Models," in Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, May 2024. doi: 10.1145/3613904.3642428.
146. Z. Yang et al., "HOTPOTQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," Association for Computational Linguistics. [Online]. Available: https://arxiv.org/abs/1809.09600
147. M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith, "How Language Model Hallucinations Can Snowball," May 2023, [Online]. Available: http://arxiv.org/abs/2305.13534
148. L. Pacchiardi et al., "How to Catch an AI Liar: Lie Detection in Black-Box LLMs by Asking Unrelated Questions," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.15840
149. C. S. Mala, G. Gezici, and F. Giannotti, "Hybrid Retrieval for Hallucination Mitigation in Large Language Models: A Comparative Analysis," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2504.05324
150. J. Wu et al., "Improve Decoding Factuality by Token-wise Cross Layer Entropy of Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2502.03199
151. Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving Factuality and Reasoning in Language Models through Multiagent Debate," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14325

152. I.-C. Chern, Z. Wang, S. Das, B. Sharma, P. Liu, and G. Neubig, “Improving Factuality of Abstractive Summarization via Contrastive Reward Learning,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.04507
153. Y. Zhao et al., “Improving the Robustness of Large Language Models via Consistency Alignment,” Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.14221
154. S. Chen et al., “In-Context Sharpness as Alerts: An Inner Representation Perspective for Hallucination Mitigation,” Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.01548
155. K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg, “Inference-Time Intervention: Eliciting Truthful Answers from a Language Model,” Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.03341
156. C. Chen et al., “INSIDE: LLMs’ Internal States Retain the Power of Hallucination Detection,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.03744
157. K. Yin and G. Neubig, “Interpreting Language Models with Contrastive Explanations,” Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.10419
158. Y. Yehuda, I. Malkiel, O. Barkan, J. Weill, R. Ronen, and N. Koenigstein, “InterrogateLLM: Zero-Resource Hallucination Detection in LLM-Generated Answers,” Aug. 2024. [Online]. Available: http://arxiv.org/abs/2403.02889
159. N. Varshney et al., “Investigating and Addressing Hallucinations of LLMs in Tasks Involving Negation,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.05494
160. G. Chrysostomou, Z. Zhao, M. Williams, and N. Aletras, “Investigating Hallucinations in Pruned Large Language Models for Abstractive Summarization”, Nov. 2023, [Online] https://arxiv.org/pdf/2311.09335
161. H. Wu, X. Li, X. Xu, J. Wu, D. Zhang, and Z. Liu, “Iter-AHMCL: Alleviate Hallucination for Large Language Model via Iterative Model-level Contrastive Learning,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.12130
162. Y. Liu et al., “Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.13860
163. E. Lavrinovics, R. Biswas, J. Bjerva, and K. Hose, “Knowledge Graphs, Large Language Models, and Hallucinations: An NLP Perspective,” Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.14258
164. D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, “Knowledge Neurons in Pretrained Transformers,” Mar. 2022, [Online]. Available: http://arxiv.org/abs/2104.08696
165. Y. Zhang et al., “Knowledge Overshadowing Causes Amalgamated Hallucination in Large Language Models,” Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.08039
166. F. Wan, X. Huang, L. Cui, X. Quan, W. Bi, and S. Shi, “Knowledge Verification to Nip Hallucination in the Bud,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.10768
167. T. B. Brown et al., “Language Models are Few-Shot Learners,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.14165
168. R. Thoppilan et al., “LaMDA: Language Models for Dialog Applications,” Jan. 2022, [Online]. Available: http://arxiv.org/abs/2201.08239
169. M. Turpin, J. Michael, E. Perez, and S. R. Bowman, “Language Models Don’t Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.04388
170. S. Kadavath et al., “Language Models (Mostly) Know What They Know,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2207.05221
171. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large Language Models are Zero-Shot Reasoners,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2205.11916
172. L. Guo, Y. Fang, F. Chen, P. Liu, and S. Xu, “Large Language Models with Adaptive Token Fusion: A Novel Approach to Reducing Hallucinations and Improving Inference Efficiency.” Oct. 24, 2024. doi: 10.22541/au.172979407.71044427/v1.
173. M. Dahl, V. Magesh, M. Suzgun, and D. E. Ho, “Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models,” Journal of Legal Analysis, vol. 16, no. 1, pp. 64–93, 2024, doi: 10.1093/jla/laae003.
174. M. Elhoushi et al., “LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding,” 2024. Accessed: Aug. 03, 2025. [Online]. Available: https://arxiv.org/abs/2404.16710
175. Y. Liang, Z. Song, H. Wang, and J. Zhang, “Learning to Trust Your Feelings: Leveraging Self-awareness in LLMs for Hallucination Mitigation,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.15449
176. D. Zhou et al., “Least-to-Most Prompting Enables Complex Reasoning in Large Language Models,” May 2022, [Online]. Available: http://arxiv.org/abs/2205.10625
177. N. Guha et al., “LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models,” Aug. 2023, [Online]. Available: http://arxiv.org/abs/2308.11462
178. H. Lightman et al., “Let’s Verify Step by Step,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.20050

179. N. Nonkes, S. Agaronian, E. Kanoulas, and R. Petcu, “Leveraging Graph Structures to Detect Hallucinations in Large Language Models,” Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.04485
180. M. Son, J. Jang, and M. Kim, “Lightweight Query Checkpoint: Classifying Faulty User Queries to Mitigate Hallucinations in Large Language Model Question Answering,” Jul. 2025. Accessed: Aug. 04, 2025. [Online]. Available: https://openreview.net/pdf?id=n9C8u6tpT4
181. J. He, Y. Gong, K. Chen, Z. Lin, C. Wei, and Y. Zhao, “LLM Factoscope: Uncovering LLMs’ Factual Discernment through Inner States Analysis,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.16374
182. Z. Zhang, Y. Wang, C. Wang, J. Chen, and Z. Zheng, “LLM Hallucinations in Practical Code Generation: Phenomena, Mechanism, and Mitigation,” Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.20550
183. Z. Ji et al., “LLM Internal States Reveal Hallucination Risk Faced With a Query,” Jul. 2024, [Online]. Available: http://arxiv.org/abs/2407.03282
184. J.-Y. Yao, K.-P. Ning, Z.-H. Liu, M.-N. Ning, Y.-Y. Liu, and L. Yuan, “LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.01469
185. P. Laban et al., “LLMs as Factual Reasoners: Insights from Existing Benchmarks and Beyond,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.14540
186. S. Banerjee, A. Agarwal, and S. Singla, “LLMs Will Always Hallucinate, and We Need to Live With This,” 2024, doi: 10.48550/arXiv.2409.05746.
187. R. Cohen, M. Hamri, M. Geva, and A. Globerson, “LM vs LM: Detecting Factual Errors via Cross Examination,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.13281
188. Y.-S. Chuang, L. Qiu, C.-Y. Hsieh, R. Krishna, Y. Kim, and J. Glass, “Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using Only Attention Maps,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2407.07071
189. N. M. Guerreiro, E. Voita, and A. F. T. Martins, “Looking for a Needle in a Haystack: A Comprehensive Study of Hallucinations in Neural Machine Translation,” Aug. 2022, [Online]. Available: http://arxiv.org/abs/2208.05309
190. T. Gao, A. Fisch, and D. Chen, “Making Pre-trained Language Models Better Few-shot Learners,” Jun. 2021, [Online]. Available: http://arxiv.org/abs/2012.15723
191. O. Yoran, T. Wolfson, O. Ram, and J. Berant, “Making Retrieval-Augmented Language Models Robust to Irrelevant Context,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.01558
192. A. Gundogmusler, F. Bayindiroglu, and M. Karakucukoglu, “Mathematical Foundations of Hallucination in Transformer-Based Large Language Models for Improvisation.” Jun. 24, 2024. doi: 10.36227/techrxiv.171925719.97800486/v1.
193. K. Li et al., “Measuring and Controlling Instruction (In)Stability in Language Model Dialogs,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.10962
194. J. Wei, Y. Yao, J.-F. Ton, H. Guo, A. Estornell, and Y. Liu, “Measuring and Reducing LLM Hallucination without Gold-Standard Answers,” Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.10412
195. A. Shrivastava, J. Hullman, and M. Lamparth, “Measuring Free-Form Decision-Making Inconsistency of Language Models in Military Crisis Simulations,” Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.13204
196. L. Yu, M. Cao, J. C. K. Cheung, and Y. Dong, “Mechanistic Understanding and Mitigation of Language Model Non-Factual Hallucinations,” Jun. 2024, [Online]. Available: http://arxiv.org/abs/2403.18167
197. Y. Kim et al., “Medical Hallucinations in Foundation Models and Their Impact on Healthcare,” Feb. 2025, [Online]. Available: http://arxiv.org/abs/2503.05777
198. M. Suzgun and A. T. Kalai, “Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.12954
199. A. Bilal, M. A. Mohsin, M. Umer, M. A. K. Bangash, and M. A. Jamshed, “Meta-Thinking in LLMs via Multi-Agent Reinforcement Learning: A Survey,” Apr. 2025, [Online]. Available: http://arxiv.org/abs/2504.14520
200. S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, MI, USA, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909/
201. W. Liu et al., “Mind’s Mirror: Distilling Self-Evaluation Capability and Comprehensive Thinking from Large Language Models,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.09214
202. S. Fairburn and J. Ainsworth, “Mitigate Large Language Model Hallucinations with Probabilistic Inference in Graph Neural Networks.” Jul. 01, 2024. doi: 10.22541/au.171987466.61187752/v1.
203. W. Su, Y. Tang, Q. Ai, C. Wang, Z. Wu, and Y. Liu, “Mitigating Entity-Level Hallucination in Large Language Models,” in Proceedings of the 2024 Annual International ACM SIGIR Conference on Research

and Development in Information Retrieval in the Asia Pacific Region, New York, NY, USA: ACM, Dec. 2024, pp. 23–31. doi: 10.1145/3673791.3698403.

204. A. Braverman, W. Zhang, and Q. Gu, "Mitigating Hallucination in Large Language Models with Explanatory Prompting," 2024, [Online]. Available: https://neurips.cc/virtual/2024/105546
205. M. Grayson, C. Patterson, B. Goldstein, S. Ivanov, and M. Davidson, "Mitigating Hallucinations in Large Language Models using a Channel-Aware Domain-Adaptive Generative Adversarial Network (CADAGAN)." Sep. 30, 2024. doi: 10.21203/rs.3.rs-5164079/v1.
206. Z. Tang, R. Chatterjee, and S. Garg, "Mitigating Hallucinated Translations in Large Language Models with Hallucination-focused Preference Optimization," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.17295
207. F. Harrington, E. Rosenthal, and M. Swinburne, "Mitigating Hallucinations in Large Language Models with Sliding Generation and Self-Checks." Aug. 06, 2024. doi: 10.36227/techrxiv.172297712.23792600/v1.
208. X. Guan et al., "Mitigating Large Language Model Hallucinations via Autonomous Knowledge Graph-Based Retrofitting," 2024. [Online]. Available: http://arxiv.org/abs/2311.13314
209. M. Hu, B. He, Y. Wang, L. Li, C. Ma, and I. King, "Mitigating Large Language Model Hallucination with Faithful Finetuning," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.11267
210. H. Li, G. Appleby, K. Alperin, S. R. Gomez, and A. Suh, "Mitigating LLM Hallucinations with Knowledge Graphs: A Case Study," Apr. 2025, [Online]. Available: http://arxiv.org/abs/2504.12422
211. J. Pfeiffer, F. Piccinno, M. Nicosia, X. Wang, M. Reid, and S. Ruder, "mmT5: Modular Multilingual Pre-Training Solves Source Language Hallucinations," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14224
212. K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.05722
213. D. Huh and P. Mohapatra, "Multi-agent Reinforcement Learning: A Comprehensive Survey," Jul. 2024, [Online]. Available: http://arxiv.org/abs/2312.10256
214. L. van der Poel, R. Cotterell, and C. Meister, "Mutual Information Alleviates Hallucinations in Abstractive Summarization," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.13210
215. T. Kwiatkowski et al., "Natural Questions: A Benchmark for Question Answering Research", doi: 10.1162/tacl.
216. D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.0473
217. N. Dziri, A. Madotto, O. Zaiane, and A. J. Bose, "Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.08455
218. J. Hoscilowicz et al., "Non-Linear Inference Time Intervention: Improving LLM Truthfulness," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.18680
219. J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On Faithfulness and Factuality in Abstractive Summarization," May 2020, [Online]. Available: http://arxiv.org/abs/2005.00661
220. Y. Xiao and W. Y. Wang, "On Hallucination and Predictive Uncertainty in Conditional Language Generation," Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.15025
221. C. Jiang et al., "On Large Language Models' Hallucination with Regard to Known Facts," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.20009
222. L. Parcalabescu and A. Frank, "On Measuring Faithfulness or Self-consistency of Natural Language Explanations," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.07466
223. N. Dziri, S. Milton, M. Yu, O. Zaiane, and S. Reddy, "On the Origin of Hallucinations in Conversational Models: Is it the Datasets or the Models?" Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.07931
224. N. Shazeer et al., "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," Jan. 2017, [Online]. Available: http://arxiv.org/abs/1701.06538
225. R. Chen, A. Arditi, H. Sleight, O. Evans, and J. Lindsey, "Persona Vectors: Monitoring and Controlling Character Traits in Language Models," Jul. 2025, [Online]. Available: http://arxiv.org/abs/2507.21509
226. N. Joshi, J. Rando, A. Saparov, N. Kim, and H. He, "Personas as a Way to Model Truthfulness in Language Models," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.18168
227. D. Zhu et al., "PoLLMgraph: Unraveling Hallucinations in Large Language Models via State Transition Dynamics," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.04722
228. J. N. Yan et al., "Predicting Text Preference Via Structured Comparative Reasoning," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.08390
229. X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation," Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.00190
230. Z. Sun et al., "Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision," May 2023, [Online]. Available: http://arxiv.org/abs/2305.03047

231. S. Cao et al., "Probabilistic Tree-of-thought Reasoning for Answering Knowledge-intensive Complex Questions," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.13982
232. C. Si et al., "Prompting GPT-3 To Be Reliable," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.09150
233. E. Y. Chang, "Prompting Large Language Models With the Socratic Method," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2303.08769
234. Q. Jin, B. Dhingra, Z. Liu, W. W. Cohen, and X. Lu, "PubMedQA: A Dataset for Biomedical Research Question Answering," Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.06146
235. A. Chen, P. Pasupat, S. Singh, H. Lee, and K. Guu, "PURR: Efficiently Editing Language Model Hallucinations by Denoising Language Model Corruptions," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14908
236. O. Honovich, L. Choshen, R. Aharoni, E. Neeman, I. Szpektor, and O. Abend, Q2: Evaluating Factual Consistency in Knowledge-Grounded Dialogues via Question Generation and Question Answering," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.08202
237. L. Du et al., "Quantifying and Attributing the Hallucination of Large Language Models via Association Analysis," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.05217
238. N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying Memorization Across Neural Language Models," Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.07646
239. H. Zhang et al., "R-Tuning: Instructing Large Language Models to Say `I Don't Know'," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.09677
240. H. Q. Yu and F. McQuade, "RAG-KG-IL: A Multi-Agent Hybrid Framework for Reducing Hallucinations and Enhancing LLM Reasoning through RAG and Incremental Knowledge Graph Learning Integration," Mar. 2025, [Online]. Available: http://arxiv.org/abs/2503.13514
241. C. Niu et al., "RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2401.00396
242. J. J. Ross, E. Khramtsova, A. van der Vegt, B. Koopman, and G. Zuccon, "RARR Unraveled: Component-Level Insights into Hallucination Detection and Mitigation," in SIGIR 2025 - Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, Inc, Jul. 2025, pp. 3286–3295. doi: 10.1145/3726302.3730337.
243. L. Gao et al., "RARR: Researching and Revising What Language Models Say, Using Language Models," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.08726
244. D. Su et al., "Read before Generate! Faithful Long Form Question Answering with Machine Reading," 2022. Accessed: May 12, 2025. [Online]. Available: https://arxiv.org/abs/2203.00343
245. S. Hao et al., "Reasoning with Language Model is Planning with World Model," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14992
246. E. Berberette, J. Hutchins, and A. Sadovnik, "Redefining 'Hallucination' in LLMs: Towards a psychology-informed framework for mitigating misinformation," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.01769
247. S. Suzuoki and K. Hatano, "Reducing Hallucinations in Large Language Models: A Consensus Voting Approach Using Mixture of Experts." Jun. 24, 2024. doi: 10.36227/techrxiv.171925057.75949684/v1.
248. Y. Liu et al., "Reducing hallucinations of large language models via hierarchical semantic piece," Complex and Intelligent Systems, vol. 11, no. 5, May 2025, doi: 10.1007/s40747-025-01833-9.
249. S. Verma, K. Tran, Y. Ali, and G. Min, "Reducing LLM Hallucinations using Epistemic Neural Networks," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.15576
250. Z. Zhao, S. B. Cohen, and B. Webber, "Reducing Quantity Hallucinations in Abstractive Summarization," Sep. 2020, [Online]. Available: http://arxiv.org/abs/2009.13312
251. X. Hu et al., "RefChecker: Reference-based Fine-grained Hallucination Checker and Benchmark for Large Language Models," May 2024, [Online]. Available: http://arxiv.org/abs/2405.14486
252. T. Yan and T. Xu, "Refining the Responses of LLMs by Themselves," May 2023, [Online]. Available: http://arxiv.org/abs/2305.04039
253. N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.11366
254. R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction." 2015, [Online]. Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
255. W. Shi et al., "REPLUG: Retrieval-Augmented Black-Box Language Models," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.12652
256. A. Zou et al., "Representation Engineering: A Top-Down Approach to AI Transparency," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.01405

257. K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval Augmentation Reduces Hallucination in Conversation," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.07567
258. J. Feng, Q. Wang, H. Qiu, and L. Liu, "Retrieval In Decoder benefits generative models for explainable complex question answering," Neural Networks, vol. 181, Jan. 2025, doi: 10.1016/j.neunet.2024.106833.
259. P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," May 2020, [Online]. Available: http://arxiv.org/abs/2005.11401
260. Z. Ji et al., "RHO (ϱ): Reducing Hallucination in Open-domain Dialogues with Knowledge Grounding." May 2023, [Online]. Available: https://arxiv.org/abs/2212.01588
261. H. Lee et al., "RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.00267
262. K. Yang, D. Klein, A. Celikyilmaz, N. Peng, and Y. Tian, "RLCD: Reinforcement Learning from Contrastive Distillation for Language Model Alignment," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.12950
263. C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in Proceedings of the ACL Workshop on Text Summarization Branches Out, Barcelona, Spain, Jul. 2004, pp. 74–81.
264. H. W. Chung et al., "Scaling Instruction-Finetuned Language Models," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.11416
265. X. Wang et al., "SciBench: Evaluating College-Level Scientific Problem-Solving Abilities of Large Language Models," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.10635
266. P. Wang, Z. Wang, Z. Li, Y. Gao, B. Yin, and X. Ren, "SCOTT: Self-Consistent Chain-of-Thought Distillation," May 2023, [Online]. Available: http://arxiv.org/abs/2305.01879
267. M. Li, B. Peng, M. Galley, J. Gao, and Z. Zhang, "Self-Checker: Plug-and-Play Modules for Fact-Checking with Large Language Models," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14623
268. X. Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models." May. 2023, [Online]. Available: https://arxiv.org/abs/2203.11171
269. N. Mündler, J. He, S. Jenko, and M. Vechev, "Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation," May 2023, [Online]. Available: http://arxiv.org/abs/2305.15852
270. T. Schick, S. Udupa, and H. Schütze, "Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2103.00453
271. Y. Wang et al., "Self-Instruct: Aligning Language Models with Self-Generated Instructions," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.10560
272. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.11511
273. A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.17651
274. P. Manakul, A. Liusie, and M. J. F. Gales, "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.08896
275. X. Qiu and R. Miikkulainen, "Semantic Density: Uncertainty Quantification for Large Language Models through Confidence Measurement in Semantic Space," May 2024, [Online]. Available: http://arxiv.org/abs/2405.13845
276. J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal, "Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs," 2024. Accessed: May 03, 2025. [Online]. Available: https://arxiv.org/pdf/2406.15927
277. J. Kai, T. Zhang, H. Hu, and Z. Lin, "SH2: Self-Highlighted Hesitation Helps You Decode More Truthfully," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.05930
278. Z. He, B. Zhang, and L. Cheng, "Shakespearean Sparks: The Dance of Hallucination and Creativity in LLMs' Decoding Layers," Mar. 2025, [Online]. Available: http://arxiv.org/abs/2503.02851
279. Y. Zhang et al., "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.01219
280. H. Nguyen, Z. He, S. A. Gandre, U. Pasupulety, S. K. Shivakumar, and K. Lerman, "Smoothing Out Hallucinations: Mitigating LLM Hallucination with Smoothed Knowledge Distillation," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.11306
281. N. McKenna, T. Li, L. Cheng, M. J. Hosseini, M. Johnson, and M. Steedman, "Sources of Hallucination by Large Language Models on Inference Tasks," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14552
282. P. Elchafei and M. Abu-Elkheir, "Span-Level Hallucination Detection for LLM-Generated Answers," Apr. 2025, [Online]. Available: http://arxiv.org/abs/2504.18639
283. Y. Qiu, Z. Zhao, Y. Ziser, A. Korhonen, E. M. Ponti, and S. B. Cohen, "Spectral Editing of Activations for Large Language Model Alignment," May 2024, [Online]. Available: http://arxiv.org/abs/2405.09719

284. R. Tian, S. Narayan, T. Sellam, and A. P. Parikh, “Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.08684
285. B. A. Levinstein and D. A. Herrmann, “Still No Lie Detector for Language Models: Probing Empirical and Conceptual Roadblocks,” Jun. 2023, doi: 10.1007/s11098-023-02094-3.
286. Z. Ji et al., “Survey of Hallucination in Natural Language Generation,” July. 2024, [Online]. Available: https://arxiv.org/pdf/2202.03629
287. E. Jones et al., “Teaching Language Models to Hallucinate Less with Synthetic Tasks,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06827
288. S. Lin, J. Hilton, and O. Evans, “Teaching Models to Express Their Uncertainty in Words,” May 2022, [Online]. Available: http://arxiv.org/abs/2205.14334
289. V. Raunak, A. Menezes, and M. Junczys-Dowmunt, “The Curious Case of Hallucinations in Neural Machine Translation,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.06683
290. A. Slobodkin, O. Goldman, A. Caciularu, I. Dagan, and S. Ravfogel, “The Curious Case of Hallucinatory (Un)answerability: Finding Truths in the Hidden States of Over-Confident Large Language Models,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.11877
291. A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The Curious Case of Neural Text Degeneration,” Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.09751
292. J. Li et al., “The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.03205
293. R. Xu et al., “The Earth is Flat because...: Investigating LLMs’ Belief towards Misinformation via Persuasive Conversation,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.09085
294. S. Marks and M. Tegmark, “The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06824
295. G. Hong et al., “The Hallucinations Leaderboard -- An Open Effort to Measure Hallucinations in Large Language Models,” Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.05904
296. A. Azaria and T. Mitchell, “The Internal State of an LLM Knows When It’s Lying,” Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.13734
297. S. Zhang, L. Pan, J. Zhao, and W. Y. Wang, “The Knowledge Alignment Problem: Bridging Human and External Knowledge for Large Language Models,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.13669
298. K. van Deemter, “The Pitfalls of Defining Hallucination,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.07897
299. B. Lester, R. Al-Rfou, N. Constant, and G. Research, “The Power of Scale for Parameter-Efficient Prompt Tuning.” [Online]. Available: https://arxiv.org/abs/2104.08691
300. V. Rawte et al., “The Troubling Emergence of Hallucination in Large Language Models -- An Extensive Definition, Quantification, and Prescriptive Remediations,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.04988
301. X. Cheng, J. Li, W. X. Zhao, and J.-R. Wen, “Think More, Hallucinate Less: Mitigating Hallucinations via Dual Process of Fast and Slow Thinking,” Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.01306
302. Y. Qiu, V. Embar, S. B. Cohen, and B. Han, “Think While You Write: Hypothesis Verification Promotes Faithful Knowledge-to-Text Generation,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.09467
303. Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung, “Towards Mitigating Hallucination in Large Language Models via Self-Reflection,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06271
304. Z. Lin, S. Guan, W. Zhang, H. Zhang, Y. Li, and H. Zhang, “Towards trustworthy LLMs: a review on debiasing and dehallucinating in large language models,” Artificial Intelligence Review, vol. 57, no. 9, Sep. 2024, doi: 10.1007/s10462-024-10896-y.
305. M. Sharma et al., “Towards Understanding Sycophancy in Language Models,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.13548
306. J. Hoffmann et al., “Training Compute-Optimal Large Language Models,” Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.15556
307. L. Ouyang et al., “Training language models to follow instructions with human feedback,” Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.02155
308. P. Feldman, J. R. Foulds, and S. Pan, “Trapping LLM Hallucinations Using Tagged Context Prompts,” Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.06085
309. S. Yao et al., “Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” May 2023, [Online]. Available: http://arxiv.org/abs/2305.10601
310. M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension,” May 2017, [Online]. Available: http://arxiv.org/abs/1705.03551

311. Z. Gekhman, J. Herzig, R. Aharoni, C. Elkind, and I. Szpektor, "TrueTeacher: Learning Factual Consistency Evaluation with Large Language Models," May 2023, [Online]. Available: http://arxiv.org/abs/2305.11171
312. W. Shi, X. Han, M. Lewis, Y. Tsvetkov, L. Zettlemoyer, and S. W. Yih, "Trusting Your Evidence: Hallucinate Less with Context-aware Decoding," May 2023, [Online]. Available: http://arxiv.org/abs/2305.14739
313. Z. Chen et al., "Truth Forest: Toward Multi-Scale Truthfulness in Large Language Models through Intervention without Tuning," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.17484
314. A. Simhi, I. Itzhak, F. Barez, G. Stanovsky, and Y. Belinkov, "Trust Me, I'm Wrong: High-Certainty Hallucinations in LLMs," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.12964
315. S. Lin, J. Hilton and O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," Long Papers, May 2022. [Online]. Available: https://arxiv.org/abs/2109.07958
316. S. Zhang, T. Yu, and Y. Feng, "TruthX: Alleviating Hallucinations by Editing Large Language Models in Truthful Space," Feb. 2024, [Online]. Available: http://arxiv.org/abs/2402.17811
317. L. Chen, X. Wu, Z. Xiong, and X. Kang, "Two Stage Psychology-Guided Fine-Grained Editing and Sampling Approach for Mitigating Hallucination in Large Language Models Publication," 2025. Accessed: Aug. 04, 2025. [Online]. Available: https://escholarship.org/uc/item/0gn8m1qq
318. X. Liang et al., "UHGEval: Benchmarking the Hallucination of Chinese Large Language Models via Unconstrained Generation," Nov. 2023, doi: 10.18653/v1/2024.acl-long.288.
319. W. Xu, S. Agrawal, E. Briakou, M. J. Martindale, and M. Carpuat, "Understanding and Detecting Hallucinations in Neural Machine Translation via Model Introspection," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.07779
320. A. Pagnoni, V. Balachandran, and Y. Tsvetkov, "Understanding Factuality in Abstractive Summarization with FRANK: A Benchmark for Factuality Metrics," Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.13346
321. G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.01644
322. D. Cheng et al., "UPRISE: Universal Prompt Retrieval for Improving Zero-Shot Evaluation," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.08518
323. Abd Elrahman Amer and Magdi Amer, "Using multi-agent architecture to mitigate the risk of LLM hallucinations," Jul. 2025, Accessed: Jul. 05, 2025. [Online]. Available: https://arxiv.org/pdf/2507.01446
324. R. Zhao, X. Li, S. Joty, C. Qin, and L. Bing, "Verify-and-Edit: A Knowledge-Enhanced Chain-of-Thought Framework," May 2023, [Online]. Available: http://arxiv.org/abs/2305.03268
325. M. Rateike, C. Cintas, J. Wamburu, T. Akumu, and S. Speakman, "Weakly Supervised Detection of Hallucinations in LLM Activations," Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.02798
326. J. D. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts," in Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, Apr. 2023. doi: 10.1145/3544548.3581388.
327. A. Lewis, M. White, J. Liu, T. Koike-Akino, K. Parsons, and Y. Wang, "Winning Big with Small Models: Knowledge Distillation vs. Self-Training for Reducing Hallucination in QA Agents," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2502.19545
328. C. Xu et al., "WizardLM: Empowering Large Language Models to Follow Complex Instructions," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.12244
329. J. Luo, C. Xiao, and F. Ma, "Zero-Resource Hallucination Prevention for Large Language Models," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.02654

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.