

Theoretical Foundations and Mitigation of Hallucination in Large Language Models

Esmail Gumaan*

Department of Computer Science, University of Sana'a

esmailG231601@ue.ye.eud

Abstract—Hallucination in Large Language Models (LLMs) refers to the generation of content that is not faithful to the input or the real-world facts. This paper provides a rigorous treatment of hallucination in LLMs, including formal definitions and theoretical analyses. We distinguish between intrinsic and extrinsic hallucinations, and define a *hallucination risk* for models. We derive bounds on this risk using learning-theoretic frameworks (PAC-Bayes and Rademacher complexity). We then survey detection strategies for hallucinations, such as token-level uncertainty estimation, confidence calibration, and attention alignment checks. On the mitigation side, we discuss approaches including retrieval-augmented generation, hallucination-aware fine-tuning, logit calibration, and the incorporation of fact-verification modules. We propose a unified detection and mitigation workflow, illustrated with a diagram, to integrate these strategies. Finally, we outline evaluation protocols for hallucination, recommending datasets, metrics, and experimental setups to quantify and reduce hallucinations. Our work lays a theoretical foundation and practical guidelines for addressing the crucial challenge of hallucination in LLMs.

I. INTRODUCTION

Large Language Models (LLMs) such as GPT-3 have demonstrated remarkable capabilities in natural language generation, achieving fluent and contextually relevant outputs in tasks from summarization to dialogue [3], [2], [1]. However, a critical challenge that has emerged with these models is the tendency to *hallucinate*—produce plausible-sounding content that is factually incorrect or not supported by the input or reality [4], [5]. Hallucinations can manifest in various forms, from minor factual inaccuracies to entire fabricated statements, undermining the reliability of LLMs in high-stakes applications (e.g., medical or legal domains) [6]. The prevalence of hallucinations has been observed across multiple domains: for instance, early neural dialogue systems sometimes generated inconsistent or untrue responses [7], neural machine translation systems occasionally produced unrelated outputs especially for out-of-distribution inputs [8], [9], and abstractive summarization models often include details not present in the source text [5]. Researchers have broadly categorized hallucinations in text generation into two types: *intrinsic* and *extrinsic* [5], [4]. Intrinsic hallucinations occur when the generated output contradicts or distorts the given source input, while extrinsic hallucinations introduce new information that cannot be verified against the source (often introducing facts that are entirely fabricated or irrelevant) [5]. Both types are problematic, with intrinsic hallucinations violating input

faithfulness and extrinsic hallucinations potentially spreading misinformation if taken as factual [4]. Recent surveys underscore that hallucination is a pervasive issue in current LLMs [4], [10], and significant research efforts are focused on understanding and mitigating this phenomenon.

In this paper, we undertake a comprehensive exploration of the theoretical foundations of hallucination in LLMs and the strategies to detect and mitigate it. We begin by providing formal definitions of hallucinations, distinguishing between intrinsic and extrinsic cases in mathematical terms. Building on these definitions, we introduce the notion of *hallucination risk* for a language model and derive theoretical bounds on this risk. In particular, we leverage tools from statistical learning theory, including PAC-Bayesian analysis and Rademacher complexity, to bound the probability of hallucination under certain assumptions. This theoretical perspective clarifies the limits of learning and highlights why completely eliminating hallucinations may be inherently difficult in general settings (echoing recent results that suggest hallucinations are fundamentally inevitable in sufficiently complex models [11]).

We then shift to practical aspects: first, we survey methods for detecting hallucinations in LLM outputs. These include token-level uncertainty estimation techniques (e.g., using probability distributions or entropy to flag low-confidence predictions [12], [13]), confidence calibration methods to adjust a model’s reported confidence to better reflect factual accuracy [14], [15], and attention-based checks that verify whether generated content is properly grounded in source input via the model’s attention patterns (particularly for tasks like summarization or translation) [16]. Next, we discuss mitigation strategies to reduce hallucinations during generation. Among these, retrieval-augmented generation (RAG) has emerged as a powerful approach, wherein the model consults an external knowledge source to ground its responses in factual data [17], [18], [30]. We also cover hallucination-aware fine-tuning and reinforcement learning strategies that train models to avoid unsupported content [20], [21], as well as techniques like logit calibration (adjusting the model’s output probabilities or decoding strategy to prevent overconfident leaps) [22], [23]. Another important direction is augmenting LLMs with fact-verification heads or modules that cross-check generated statements against reference knowledge bases or learned factual representations [24], [25]. By integrating a verification step, the model can potentially catch and correct hallucinations

before presenting the final output.

We integrate these insights by proposing a unified workflow for hallucination detection and mitigation. The workflow involves an LLM generating a preliminary answer, a detection module assessing the answer’s fidelity (through uncertainty signals and content verification), and a mitigation step (such as retrieving relevant facts or adjusting the answer) if a potential hallucination is detected. We present a diagram illustrating this pipeline and discuss how each component interacts.

Finally, we address the evaluation of hallucinations and their mitigation. We outline recommended datasets and benchmarks for assessing factuality and faithfulness, such as TruthfulQA for open-domain truthfulness [28], factual consistency benchmarks in summarization (e.g., datasets with human-annotated hallucinations [5], [29]), and domain-specific tests (like medical question answering where factual accuracy is critical). We also review common metrics for quantifying hallucinations, ranging from simple precision/recall of factual statements to more sophisticated automatic metrics like FactCC [24], QAGS [26], and TRUE [27]. We emphasize the importance of human evaluation as the gold standard for detecting subtle hallucinations and recommend experimental protocols (such as A/B testing of models with and without mitigation modules, and measuring improvements in factual accuracy and calibration).

The remainder of this paper is organized as follows. Section II formalizes the definition of hallucination in language generation, including intrinsic and extrinsic forms, and introduces the hallucination risk framework. Section III develops theoretical bounds on hallucination risk using PAC-Bayes and complexity measures. Section IV discusses approaches for detecting hallucinations in LLM outputs. Section V presents various mitigation techniques to reduce hallucinations. In Section VI, we propose a combined detection-mitigation workflow and provide a diagrammatic representation. Section VII gives recommendations for evaluation datasets, metrics, and experimental procedures to study hallucinations. Finally, Section VIII concludes with reflections on future research directions for making LLMs more truthful and reliable.

II. HALLUCINATION DEFINITIONS AND FORMALIZATION

In this section, we provide a formal definition of hallucination in the context of language models. We distinguish between *intrinsic* hallucinations (inconsistencies with respect to a given source input) and *extrinsic* hallucinations (content unsupported by any provided input or known reference), following the taxonomy introduced in the summarization and translation literature [5], [4].

A. Intrinsic vs. Extrinsic Hallucinations

Consider a conditional language generation setting with input x (e.g., a source document or prompt) and output text y produced by the model. Let $I(x)$ denote the set of factual assertions present in the input x . Likewise, let $O(y)$ denote the set of assertions made in the output y . We say that the output y is **intrinsically hallucinated** (with respect to x) if there exists at least one proposition in $O(y)$ that *directly contradicts*

information in $I(x)$. In other words, y asserts something that is negated or refuted by the source content x (thus violating consistency) [5]. Formally:

$$\exists p \in O(y) \text{ such that } p \text{ is logically incompatible with } I(x).$$

Intrinsic hallucinations are often easier to detect because the contradiction with the source can sometimes be identified by entailment-checking or overlap with source facts. For example, if x is a document stating "The Eiffel Tower is located in Paris" and the summary y states "The Eiffel Tower is located in Rome," y contains an intrinsic hallucination (a directly contradictory claim).

In contrast, y is said to be **extrinsically hallucinated** if y includes information that is not present in $I(x)$ and cannot be verified by any accessible knowledge source, despite not necessarily contradicting x [5], [4]. Extrinsic hallucinations introduce *new* assertions that go beyond the input. Formally:

$$\exists q \in O(y) \text{ such that } q \not\vdash I(x) \text{ (not entailed by } I(x)),$$

and typically q corresponds to some factual claim for which x provides no evidence. For example, if x is an article about Paris and y (a summary) adds a sentence "Paris is home to the largest rainforest in Europe," this is extrinsic hallucination: the added detail is not in the source and is in fact a fabricated or unrelated claim (and cannot be verified as true from the given input). Extrinsic hallucinations often require external knowledge or fact-checking to detect, since the model might introduce a plausible-sounding but incorrect fact that the input never mentioned.

It is worth noting that whether a piece of generated content counts as a hallucination can depend on the task context and the expected scope of the output. In strictly input-bound tasks (like translation or faithful summarization), *any* content not grounded in the input is undesired (and thus extrinsically hallucinated). In open-ended creative generation or dialogue, extrinsic additions might be acceptable or even required (for engagement), as long as they remain consistent with general world knowledge and do not introduce false facts. In this work, we focus on hallucinations in contexts where factuality and faithfulness are expected, such as summarization, question-answering, and knowledge-grounded dialogue.

B. Hallucination Risk

We introduce the notion of *hallucination risk* to quantify how prone a language model is to hallucinate. Intuitively, hallucination risk refers to the probability that the model’s output will contain a hallucination (of either type) under the distribution of inputs of interest.

Let \mathcal{X} be a distribution over inputs (and possibly paired with ground-truth outputs or an underlying truth source). We assume there is a (possibly unknown) ground-truth function or oracle $f^*(x)$ that provides a fully truthful and contextually appropriate output for input x (for instance, the correct answer in a QA task or a perfectly faithful summary of a document).

We can then define a **hallucination indicator** for a model M on input x :

$$H(M, x) = \begin{cases} 1, & \text{if the output } y = M(x) \text{ hallucination,} \\ 0, & \text{if } M(x) \text{ is completely faithful to } f^*(x) \text{ (no hallucination).} \end{cases}$$

Using this indicator, the **hallucination risk** $R_{\text{hall}}(M)$ is the expected value:

$$R_{\text{hall}}(M) = \mathbb{E}_{x \sim \mathcal{X}} [H(M, x)],$$

which is simply the probability that M produces a hallucination on a random input from \mathcal{X} . In practice, since hallucination is a binary condition per output, this risk can also be interpreted as the hallucination rate.

We can similarly define an *intrinsic hallucination risk* and *extrinsic hallucination risk* if we wish to separate the two types:

$$R_{\text{int}}(M) = \Pr_x[M(x) \text{ has an intrinsic hallucination}],$$

$$R_{\text{ext}}(M) = \Pr_x[M(x) \text{ has an extrinsic hallucination}].$$

These are useful if the application cares differently about each type (for example, in summarization, intrinsic hallucinations might indicate serious errors against source fidelity, whereas extrinsic hallucinations might be less damaging if they are minor details, though both are generally undesirable).

Hallucination risk depends on both the model and the input distribution. A model might hallucinate rarely on inputs drawn from in-domain data (where it was trained or where it has strong knowledge) but hallucinate more on out-of-distribution or open-domain inputs where its knowledge is uncertain. This aligns with the intuition that hallucinations often occur when the model is faced with queries that exceed its knowledge or stray from the support of its training data.

III. THEORETICAL ANALYSIS OF HALLUCINATION RISK

Given the formalization above, we now consider theoretical bounds on hallucination risk. We can treat hallucination detection as a binary classification problem on model outputs (hallucinated vs. faithful). If we have a training dataset or some feedback that labels when outputs are hallucinations, one can in principle train a model or adjust the original model to minimize this risk. The challenge is that the space of all possible outputs is enormous, and directly supervising every case of hallucination is infeasible. Nonetheless, we can leverage generalization bounds from learning theory to reason about the hallucination behavior of models.

A. Generalization Perspective

From a learning perspective, a language model M can be seen as attempting to approximate the ground truth function f^* (which produces fully factual outputs) based on finite training data. Hallucinations are then instances of generalization error: cases where $M(x)$ deviates from $f^*(x)$ in a way that introduces false or unsupported content. In fact, one might say that the ultimate goal of an ideal training procedure (for tasks requiring factuality) is to minimize $R_{\text{hall}}(M)$.

If we had a way to automatically determine whether a given output is hallucinatory (for example, via human labeling or a fact-checking oracle), we could measure an empirical hallucination rate on a sample of n inputs:

$$\hat{R}_{\text{hall}}(M) = \frac{1}{n} \sum_{i=1}^n H(M, x_i),$$

where x_1, \dots, x_n are sample inputs (and we assume we can identify if $M(x_i)$ hallucinated). This is analogous to an empirical risk (error rate) in binary classification where "error" corresponds to hallucinating.

Standard generalization bounds would then relate $R_{\text{hall}}(M)$ to $\hat{R}_{\text{hall}}(M)$. For example, if M comes from a hypothesis class of bounded complexity (e.g., limited capacity or effectively controlled by regularization), we can invoke a uniform convergence bound. Using a VC-dimension or Rademacher complexity argument [36], [37], one would state that with high probability (over the choice of the n inputs):

$$R_{\text{hall}}(M) \leq \hat{R}_{\text{hall}}(M) + \mathcal{O}\left(\sqrt{\frac{\mathcal{C}}{n}}\right),$$

where \mathcal{C} is a measure of model complexity (e.g., VC-dimension or a bound on Rademacher complexity of the associated hypothesis class), and the big-O hides constant factors and $\ln(1/\delta)$ terms for a given confidence $1 - \delta$. In simpler terms, if a model exhibits a low hallucination frequency on a representative training set and the model class is not too complex, we expect it to have a low hallucination probability on new inputs as well.

However, modern LLMs are extremely high-capacity (often overparameterized) models, which makes classical complexity measures very large. In practice, they can have near-zero training error but still hallucinate on new inputs, meaning the challenge is often one of distribution shift or incomplete knowledge rather than traditional generalization in the statistical learning sense.

B. PAC-Bayesian Bound on Hallucination Risk

Another way to derive a bound is through the PAC-Bayes framework, which is well-suited for reasoning about the generalization of complex models by introducing a prior and considering a distribution over models [38]. We can derive a PAC-Bayes bound on the hallucination risk as follows. Assume a prior distribution P over models (e.g., before observing any data, some distribution reflecting our initial belief about model parameters) and let Q be a posterior distribution (concentrated around the trained model). For any $\delta > 0$, with probability at least $1 - \delta$ over the random draw of the training data, the following bound holds for all distributions Q (over models in our hypothesis space):

$$\mathbb{E}_{M \sim Q}[R_{\text{hall}}(M)] \leq \mathbb{E}_{M \sim Q}[\hat{R}_{\text{hall}}(M)] + \sqrt{\frac{\text{KL}(Q||P) + \ln \frac{1}{\delta}}{2n}}, \quad (1)$$

where $\text{KL}(Q||P)$ is the Kullback-Leibler divergence between Q and P . This is a direct application of the PAC-Bayesian

generalization bound for 0-1 loss (which hallucination indicator essentially is) [38]. If we take Q to be a point mass at our learned model M (i.e., we are considering the deterministic model we have after training), the bound simplifies to an upper bound on $R_{\text{hall}}(M)$ in terms of the empirical hallucination rate of M plus a complexity penalty that scales with the description length of M relative to the prior and with $1/\sqrt{n}$.

The utility of the bound in (1) is mostly conceptual for our purposes: it tells us that if a model has low hallucination rate on the training data (perhaps through fine-tuning on high-quality, factual responses) and if the model is not overly complex relative to our prior beliefs, then we can guarantee a low hallucination risk on new data, with high probability. Of course, in practice, defining a sensible prior P and computing the KL term can be challenging for large neural networks. Nonetheless, PAC-Bayesian analysis has been used to explain generalization even in overparameterized models by choosing informative priors (e.g., centered at an earlier state of the model before fine-tuning).

C. On the Impossibility of Eliminating Hallucination Completely

A recent theoretical result suggests that for sufficiently powerful models, some degree of hallucination may be fundamentally unavoidable [11]. In a formal setting, one can prove that no computable model can perfectly reproduce another arbitrary computable ground-truth function f^* in all cases (this is related to results in computational learning theory and the limitations of generalization). Informally, if an LLM is used as a general problem solver across an open-ended space of queries, there will always be some inputs for which the model fails to produce the correct output (unless the model is as powerful as the oracle f^* itself, which in realistic terms it is not). These failures manifest as hallucinations when the model still produces an answer, but that answer is not the correct or truthful one. In other words, hallucination in extremely general settings can be seen as a consequence of the fact that LLMs cannot know everything or perfectly generalize to every possible query [11]. This aligns with intuition: an LLM that has not been exposed to a particular rare fact during training might “guess” and thereby hallucinate when asked about it.

The theoretical takeaway is that while we can reduce the probability of hallucination (and aim to make it very low, especially in critical applications), completely eliminating hallucinations for all possible inputs is likely infeasible. Thus, detection and mitigation strategies (discussed next) are crucial complements to training better models.

IV. DETECTION OF HALLUCINATIONS

Before we can mitigate or prevent hallucinations, we must detect when they occur or are likely to occur. Detection can happen *post hoc* (after a model generates an output, identify if it contains a hallucination) or *online* (during generation, identify tokens or sequences that are potentially hallucinated in real-time). We explore several approaches to hallucination detection in LLM outputs, focusing on:

- **Uncertainty and token-level cues** – methods that use the model’s own predicted probabilities or variations to gauge confidence.
- **Confidence calibration and self-evaluation** – methods where the model or an auxiliary model assesses the likelihood of its output being correct.
- **Attention and attribution-based checks** – methods that inspect whether the model’s output content is properly grounded in the input via alignment techniques.

A. Token-Level Uncertainty Estimation

One indicative signal of a potential hallucination is the model’s *uncertainty* in generating certain tokens or facts. Intuitively, if the model is not confident (according to its own probability distribution) about a particular piece of generated information, that piece may be a hallucination. However, raw probabilities from a language model are not always well-calibrated (a model might assign high probability to a guess due to learned patterns, not because it is certain of factual correctness) [13].

Approaches to quantify uncertainty at the token or sequence level include:

- **Entropy or variance of predictions:** The entropy of the model’s next-token distribution is a measure of uncertainty. High entropy means the model is unsure which token to produce next. If a model produces a token (or sequence) while the entropy is high, it might be “guessing,” which can correlate with hallucination. Similarly, one can sample multiple continuations from the model (via Monte Carlo dropout or an ensemble of model snapshots [12]) and measure variance among outcomes. A high variance in answers to the same prompt often signals low confidence in any single answer [6].
- **Consistency under perturbations:** A method known as *self-consistency* involves posing the same question or prompt to the model multiple times (or with slight rephrasings or different sampling seeds) and seeing if the model’s answer remains consistent. If the model’s answers fluctuate significantly (especially on factual questions) this can indicate it does not actually “know” the answer and might be confabulating [6]. For example, if asked “What is the target of Sotorasib” and the model sometimes answers one protein and other times another despite identical instructions, it suggests hallucination risk.
- **Surprise relative to training data:** Another angle is to measure how likely a generated statement is under a reference distribution of truthful statements (for example, using a smaller fact-grounded model or n-gram statistics). If a sentence in y contains a very rare or unprecedented combination of tokens that was not seen in truthful contexts, it could be flagged.

In practice, Farquhar *et al.* (2024) propose computing uncertainty at the level of semantic content by clustering paraphrases of the output and measuring entropy in the semantic space,

which they found effective in detecting what they call "confabulations" (arbitrary, incorrect answers) [6]. This method aims to overcome the limitation that one idea can be expressed in many lexical ways – instead of surface-level entropy, they consider if the model is uncertain about the meaning it wants to convey.

Token-level uncertainty methods are appealing because they do not require external data; they use the model’s own behavior as a signal. However, not all hallucinations come with obvious uncertainty — sometimes a model will assert a wrong fact with high confidence (low entropy), which is a worst-case scenario. This is where calibration and external checks become vital.

B. Confidence Calibration and Self-Evaluation

Confidence calibration refers to the process of adjusting or interpreting the model’s output probabilities so that they reflect the true likelihood of correctness [14]. An uncalibrated model might be overconfident in false outputs or underconfident in true outputs. Calibrating a language model’s confidence could involve:

- **Temperature scaling:** Applying a softmax temperature or isotonic regression on predicted probabilities to better align them with empirical correctness likelihoods [14]. For example, one might fine-tune a model on a set of QA examples with known true/false answers to calibrate the relationship between the model’s probability distribution and whether the answer was correct.
- **Ask the model to rate its confidence:** Some works have shown that LLMs can produce a qualitative self-assessment when prompted (e.g., "How sure are you about the above answer"). While not entirely reliable, in certain settings large models can indicate uncertainty (like by saying "I’m not entirely sure") which correlates with actual error [13]. There are also methods where the model is trained or prompted to output a probability or confidence score along with the answer [15].
- **Calibrating through few-shot examples:** Providing a few examples of answers with confidence levels (or probabilities of being correct) in a prompt can sometimes calibrate the model in a zero-shot or few-shot manner. This instructs the model on how it should distribute probability mass when unsure.
- **External calibration models:** We can train a separate model (or use a smaller verifier network) that takes as input the LLM’s output (and possibly the question or context) and predicts a probability that the output is correct. This is akin to a regression or classification (true vs false) on the content of the answer. If well-trained, such a model can effectively flag likely hallucinations by outputting a low score for unfaithful answers [24], [27].

Confidence-based detectors often yield a scalar "factuality score" or likelihood of correctness for a given output. For example, a verification model might be built using a dataset of known factual vs hallucinated outputs and then applied to new outputs (similar to how Fact was a BERT-based classifier trained to detect factual consistency of summaries [24]). A

well-calibrated system would ideally either abstain (choose not to answer) or indicate uncertainty when it is likely to hallucinate, thereby preventing misinformation. Techniques like selective prediction [33] implement this: the system only outputs an answer when it is confident it is correct, otherwise it says "I don’t know" or defers.

C. Attention Alignment and Source Attribution

For tasks where the model is provided with a source (e.g., document, knowledge base, or context), we can exploit the model’s internal attention or alignment mechanisms to detect hallucinations. The intuition is that for a factual statement in the output, there should be some part of the input or retrieved context that the model attended to or based that statement on. If the model outputs a sentence that has no corresponding source span and the model’s attention distribution during generation of that sentence did not focus on any relevant input tokens, this could be a sign of hallucination (specifically extrinsic hallucination).

Several methods in this vein:

- **Attention-based provenance:** For each token or phrase the model generates, one can look at the encoder-decoder attention weights (in a seq2seq model like a Transformer) to see which input tokens influenced it. If a noun phrase or a factual assertion in the output has uniformly low attention weights across all input tokens (or attends to irrelevant parts of the input), it suggests the model is "hallucinating" that content without grounding [16]. For example, in translation, if the model outputs a phrase that was never in the source and attention doesn’t align it to any source phrase, it’s likely a hallucination.
- **Gradient or attribution methods:** Beyond raw attention, one can use input attribution techniques (like integrated gradients or attention flow) to identify which parts of the input most contributed to the generation of a specific output segment. Hallucinated content would show weak attribution to input features, whereas faithful content should trace back to some input evidence.
- **Verification with retrieval:** This approach is related to mitigation but can be used purely for detection: given a model output sentence, issue it (or its claim) as a query to a search engine or knowledge base. If no supporting document or evidence can be found, flag it as potential hallucination. This is essentially how a human fact-checker might operate. While not an internal model method, it is a practical detection strategy. Research prototypes have used web search to detect likely factual errors in model outputs by seeing if the facts appear in credible sources.

For summarization tasks, metrics like entity overlap or content coverage have been used: e.g., count how many named entities in the summary appear in the source. A summary that introduces unseen entities or numbers is likely hallucinating extrinsic details [5]. Similarly, in knowledge-grounded dialogue, if the conversation is supposed to stay grounded in

provided knowledge snippets and the model response introduces a new factoid not in the snippets, a detection system can catch that by string matching or semantic matching against the knowledge source [30].

Attention alignment checks have their limitations: high attention weight doesn't guarantee correctness (the model could attend to the right source but still generate an incorrect interpretation of it), and low attention weight might sometimes be deceiving (since attention can be diffused). Nonetheless, combined with other signals, attention patterns are a useful indicator.

V. MITIGATION STRATEGIES FOR HALLUCINATION

We now discuss strategies to mitigate (reduce or prevent) hallucinations in LLMs. These methods can be applied during the model training phase, at decoding time, or as post-processing steps, and many are complementary (they can be combined for better results). We focus on four broad categories that have shown promise:

A. Retrieval-Augmented Generation (RAG)

One of the most effective ways to ground a language model in factual content is to provide it with relevant reference information at generation time [17]. Retrieval-Augmented Generation (RAG) frameworks augment the model with a retrieval mechanism: given an input (e.g., a question), the system first retrieves documents or knowledge from a large external database or the web, and then the language model conditions its generation on both the input and the retrieved evidence. Because the model has access to explicit knowledge, it is less likely to fill gaps with hallucinated facts from its parameters; instead, it can quote or fuse information from the retrieved text.

Key aspects of RAG include:

- **Training with retrieval:** Some models like REALM [18] and RETRO incorporate retrieval directly into training. They learn to use a differentiable retrieval component such that at inference time, they continue to retrieve relevant text for each query. These models have been shown to produce more accurate, factual statements since they can look up facts rather than rely purely on memorized knowledge.
- **Open-domain QA and knowledge-grounded dialogue:** RAG has been successfully applied in open-domain question answering, where models like DrQA and RAG [17] retrieve Wikipedia articles to answer questions, dramatically reducing the chance of unsupported answers. Similarly, dialog systems (like BlenderBot 2.0) retrieve knowledge to ground their responses, thereby mitigating hallucinations in conversation [30].
- **Plug-and-play retrieval modules:** Even if the base LLM is not trained with retrieval, one can implement a pipeline: first retrieve top- k relevant documents (using an IR system or a dense retriever), then prepend or concatenate those documents to the model's input context, and finally generate the output. This provides the model with the

opportunity to copy or use actual facts from the evidence, rather than guessing. Many recent LLM applications (e.g., Bing's chat or other QA systems) use this approach to improve factual accuracy.

- **Challenges:** RAG is not a panacea; if the retrieval fails (e.g., no relevant document is found) or if the model misinterprets the retrieved text, hallucinations can still occur. There is also the risk of the model citing a retrieved fact incorrectly. However, the overall empirical finding is that grounding generation in retrieved data significantly reduces hallucination rates in tasks like QA, as long as a correct reference can be found [17].

By anchoring the generation to external knowledge, RAG effectively shifts the problem from "the model must know everything" to "the model must know how to find and use information," which is easier to achieve reliably. The model becomes an assembler of facts from its sources rather than a sole source of facts.

B. Hallucination-Aware Fine-Tuning and Instruction Tuning

Another approach is to train the model explicitly to avoid hallucinations. This can be done through fine-tuning on datasets that emphasize factual correctness, or via reinforcement learning with feedback on factuality.

Some methods include:

- **Supervised fine-tuning on truthful data:** If we have high-quality datasets of question-answer pairs, summaries, or dialogues where the output is guaranteed to be factual (and we have negatives that are hallucinated), we can fine-tune the LLM on this data to encourage factual generation. For example, models can be fine-tuned to produce "faithful summaries" by using a dataset of summaries that was cleaned of hallucinations or by adding a loss term that penalizes including content not present in the source.
- **Reinforcement Learning from Human Feedback (RLHF):** RLHF has been used to align LLMs with human preferences, which include truthfulness [20]. In RLHF, the model generates outputs, and a reward model (often trained on human preference data) gives higher scores to outputs that are correct and lower to those that are hallucinated or incorrect. By optimizing for this reward via policy gradients or proximal policy optimization (PPO), the model learns to avoid outputs that humans would label as hallucinations or unhelpful. InstructGPT and related models have used this to reduce the incidence of blatant false statements [20].
- **Penalizing unsupported content:** One can introduce a training signal that explicitly checks whether each statement in the output is supported by the input (for tasks where input is present). If not, a penalty is applied. This can be done by integrating a differentiable verification mechanism or by data augmentation (provide negative examples of unsupported statements and train the model to output a special token or refrain in those cases).

- **Encouraging refusals for unknowns:** Instruction tuning can teach the model to respond with uncertainty when it doesn't know an answer. For example, including prompts in the fine-tuning data like "Q: [difficult question] A: I'm sorry, I don't have enough information to answer that." helps the model learn that saying "I don't know" is acceptable and preferable to hallucinating [28]. This strategy directly combats extrinsic hallucinations by essentially opting out of answering when likely to hallucinate.

Hallucination-aware training leverages the training process to instill caution and fact-awareness in the model. However, it requires relevant training data or feedback. Human annotation of hallucinations can be expensive, so some works use semi-supervised approaches: e.g., generate candidate outputs and automatically label them as hallucinated or not using a heuristic or another model, then fine-tune on that.

It is also noteworthy that focusing too much on factual correctness can sometimes degrade the model's creativity or ability to generalize (a phenomenon sometimes called the "alignment tax" where making models safer or more factual might reduce some capability [4]). Thus, fine-tuning must strike a balance, and often it's combined with other methods rather than being the sole solution.

C. Logit Calibration and Decoding Strategies

Even without additional training or external knowledge, we can often reduce hallucinations by carefully controlling the generation process. Hallucinations are sometimes linked to the model "overshooting" with a highly likely token that leads down a wrong path (especially in greedy or beam search decoding), or conversely, sampling too freely such that random errors slip in. Techniques to calibrate or constrain the model's logits (the raw probabilities for next tokens) and to adjust decoding can help:

- **Lowering the temperature / Nucleus sampling:** By using a lower temperature in softmax sampling, we make the model's output distribution more peaky, effectively making it more deterministic and less likely to produce low-probability (potentially nonsensical) tokens. Nucleus (top- p) sampling [23] limits the sampling to a subset of tokens that cover a cumulative probability p (often $p = 0.9$). This avoids tail tokens that the model assigned small probability, which could be wild off-track continuations. These methods generally improve coherence and relevance, which indirectly reduces hallucinations. However, too low a temperature can also cause repetition or sticking to safe phrases.
- **Ban or penalize unsupported tokens:** If we have some idea of what tokens or phrases are likely to be hallucinated (for example, the model might consistently make up references or URLs in a certain format), we can apply a logit penalty or mask to prevent those from being generated unless certain conditions are met. Some production AI systems maintain blacklists or use detection during generation to stop output if a hallucination

pattern is detected (though this is usually more for toxic content, it can be adapted to factual errors).

- **Constrained decoding with knowledge:** Another advanced idea is to integrate a verification step into decoding. For instance, after each sentence generated, an auxiliary checker (like a fact-check model or a retrieval query) could validate the statement. If the checker indicates a likely error, the model can be guided to revise or drop that part. This requires careful orchestration but has been explored in methods allowing an LLM to call external tools mid-generation.

- **Logit adjustment for calibration:** If the model is known to be overconfident (its probability distribution has too low entropy compared to actual uncertainty), one can flatten the distribution (increase entropy) in a targeted way. Conversely, if a model tends to guess when it shouldn't, one might detect such situations (e.g., the query is obscure) and dynamically lower the probability of tokens that would assert facts (like names, dates) to encourage the model to say it doesn't know or to hedge.

Holtzman *et al.* [23] showed that typical max-likelihood decoding (greedy or beam search) often yields degenerate repetitive text because it over-commits to high-probability words, whereas sampling can produce more natural text. But pure sampling can also produce off-topic or incorrect info. Nucleus sampling was a compromise ensuring both coherence and diversity. In the context of factuality, one might similarly tune decoding parameters to favor safer, on-distribution completions. A very aggressive strategy is to use beam search but with a re-ranking at the end by a factuality model, selecting the highest plausible answer that is factually consistent.

D. Fact-Verification Modules and Auxiliary Heads

Finally, one can extend the model architecture by adding components dedicated to factual verification. This can take several forms:

- **Classifier head on the decoder:** For instance, during generation, after each token or sentence, an auxiliary classifier (attached to the model's hidden state) could predict whether the sequence so far is factual and consistent with the source. If it predicts a high probability of hallucination, the model could adjust or a constraint could be applied to steer it back. Training such an internal classifier might involve multi-task learning: the model not only learns to generate the next token but also to judge the truth of what has been generated so far.
- **Two-pass generation (draft and verify):** In this approach, the model first generates a draft answer. Then, a second pass (either by the same model or a specialized verification model) checks each statement in the draft. The model can then be prompted to correct any statements that the verifier flagged as likely incorrect. This chain-of-verification idea is explored by Dhuliawala *et al.* (2023) [34], where the model generates and then formulates verification questions about its own output, answering them with an external tool or its own internal

knowledge to validate the content. This effectively adds a self-check mechanism.

- **Tool use and fact-checkers:** LLMs can be augmented to use external tools like search engines, calculators, or knowledge bases mid-generation (e.g., Toolformer techniques). For factual questions, the model can decide to issue a query and then incorporate the result, rather than directly answering from its parametric memory. If integrated properly, this ensures that specific facts are fetched from a reliable source. The model's architecture might include a decision head that triggers a tool usage when confidence is low.
- **Knowledge incorporation in training:** Another angle is to embed a knowledge graph or database into the model's representations, or to pre-train/fine-tune the model with objectives that tie it to factual knowledge (such as link prediction tasks or masked language modeling on factual texts). A model that has an internal knowledge graph can effectively have a built-in verifier that checks consistency against that graph. Some recent studies incorporate knowledge graph embeddings into the transformer hidden states to bias generation toward known facts [35].

The idea of a fact-verification module is analogous to having an editor or fact-checker watch over the writer (the LLM) in real-time. This can catch errors that the writer might make inadvertently. For example, if the model says "In 1975, the population of X was Y," a verification head might internally cross-check that with its training data or an external table and realize it's hallucinated, then either adjust that part or not output it.

An important consideration is that any verification system is only as good as the knowledge it has. If either the model or the verifier lack certain knowledge, they might not catch a hallucination. Thus, combining retrieval (to supply knowledge) with verification is often most effective.

VI. PROPOSED DETECTION AND MITIGATION WORKFLOW

Synthesizing the above detection and mitigation strategies, we propose a workflow for deploying an LLM in a setting where factual accuracy is paramount. The workflow ensures that for each user query or input, the system checks for possible hallucinations and applies mitigation steps before finalizing a response. Figure 1 illustrates this pipeline.

The steps in the workflow are:

- 1) **Initial Generation:** Given a user query or task input, the LLM produces an initial answer. This is done with a potentially cautious decoding strategy (e.g., moderately low temperature to avoid too much randomness).
- 2) **Hallucination Detection:** The draft answer is passed to a detection module. This could involve:
 - Checking the model's self-reported uncertainty or the entropy of the generation (from logs recorded during generation).
 - Using a classifier or heuristic to identify unsupported factual claims (for instance, scanning the

answer for sentences containing facts and verifying each against the input or a knowledge source).

- If the input provides grounding (like documents), using an overlap or attention-based metric to see if all facts in the answer come from the input.

If the detection module finds no clear hallucination (the answer appears factual and supported), the pipeline proceeds to output the answer. If a potential hallucination is flagged, we move to the next step.

- 3) **Mitigation Actions:** Upon detecting a likely hallucination, the system can take one or more mitigation actions:

- *Retrieve and Refine:* Perform a retrieval query for the contentious parts of the answer or the whole question. For example, if the answer stated a specific fact that is unverified, query a search engine or database with that fact or question. Incorporate the retrieved evidence into the context and prompt the LLM to regenerate or adjust the answer using the new information (essentially a RAG second-pass).
- *Verify and Edit:* Use a fact-checker module to pinpoint which part of the answer is false. Then either programmatically edit the answer (e.g., remove or replace the false statement) or prompt the LLM with feedback. For instance, "In your answer, the statement X seems incorrect. Please correct it." This utilizes the model's ability to do targeted correction when guided.
- *Abstain or Qualify:* If the hallucination is due to a question that the model genuinely cannot answer correctly (no knowledge available), the mitigation might be to replace the answer with a refusal or a statement of uncertainty (like "I'm sorry, I don't have that information."). It's better to have no answer than a wrong one in many applications.

After mitigation, an updated answer is produced.

- 4) **Final Answer:** The refined answer (post-mitigation) is delivered as the final output. Ideally, this answer has any hallucinated content removed or corrected. In cases where retrieval was used, the answer might now explicitly include references or evidence ("According to [source], ...") to increase user trust.

This workflow can be iterative. If the final answer is still uncertain, the detection step could run again. In practice, one iteration is usually aimed for, since too many loops could cause delays. But a system might allow a second loop if the first mitigation still produced an answer that the detector is not fully satisfied with.

An example scenario: User asks a complex question, LLM gives an answer that includes a date and name that detector flags as potentially wrong. The system retrieves relevant Wikipedia info, finds the correct date and name, prompts LLM to correct those. The final answer with correct facts is returned. If the retrieval found no evidence, the system might respond with uncertainty rather than risk a guess.

The above workflow is in line with what some deployed

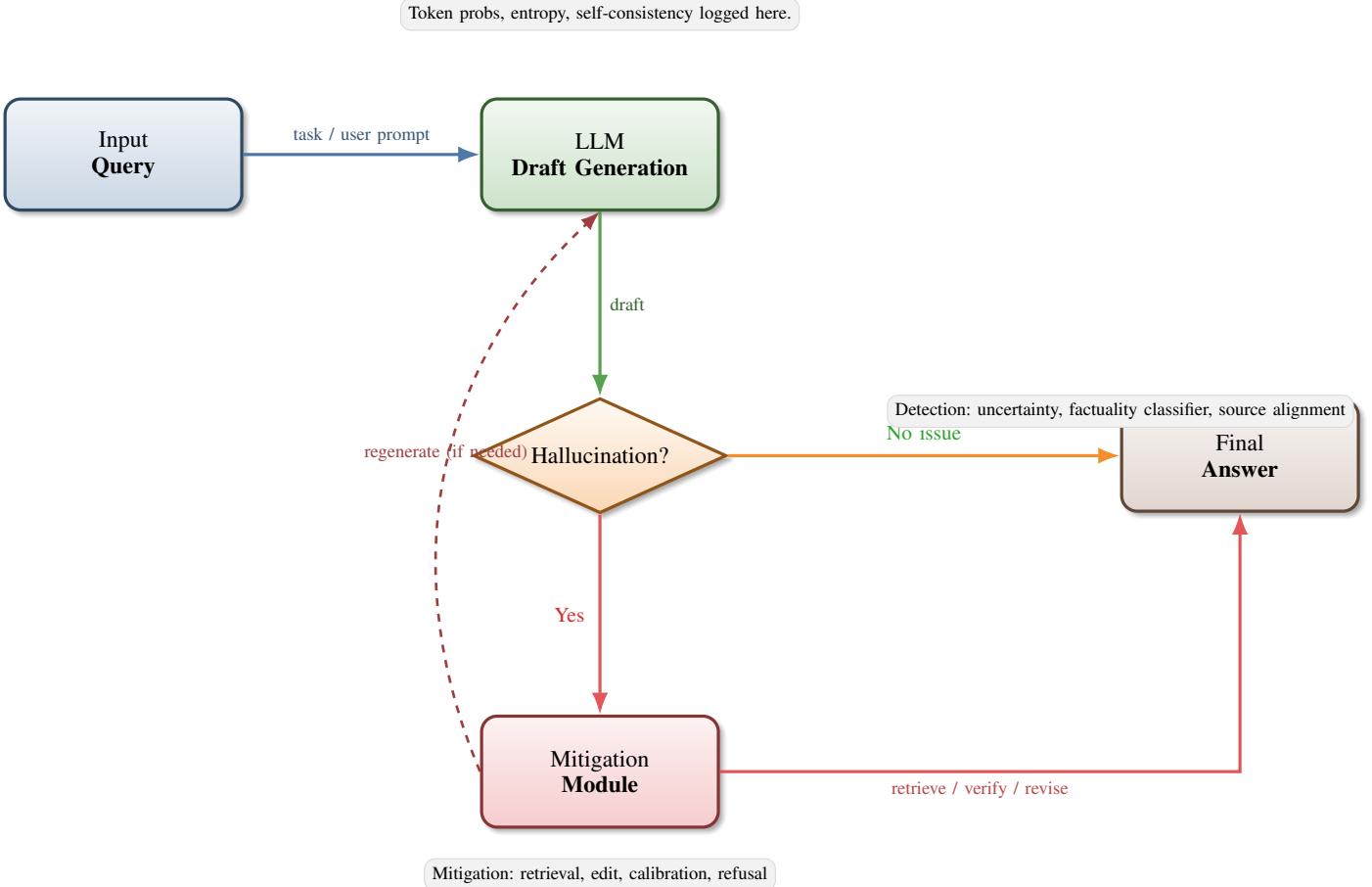


Fig. 1. Proposed workflow for hallucination detection and mitigation. The LLM generates a draft response given an input query. A detection module then evaluates the response for potential hallucinations (intrinsic or extrinsic). If no hallucination is detected (No), the response is finalized and returned. If a hallucination is detected (Yes), a mitigation module is invoked, which could involve retrieving additional information, applying corrections, or re-generating portions of the answer. The mitigated answer is then produced as the final answer.

AI assistants do, combining large LLMs with search engines and checkers to reduce incorrect outputs. It leverages both the generative strength of LLMs and the precision of knowledge-based systems.

VII. EVALUATION PROTOCOLS FOR HALLUCINATION

Evaluating hallucination in LLMs requires careful consideration, as it involves assessing factual correctness and faithfulness, which can be subtle. We outline recommended practices for empirically evaluating hallucination and the effectiveness of mitigation techniques.

A. Datasets and Benchmarks

A variety of benchmarks have been proposed to stress-test models' tendency to hallucinate:

- **Factual Question Answering Benchmarks:** Datasets like TruthfulQA [28] specifically target whether models produce truthful answers to questions that might prompt common misconceptions or require knowledge. TruthfulQA provides questions and categorizes answers as truthful or false, enabling a direct measure of hallucination (falsehood) rates. Another is the "Factoid QA" where

every question has a verifiable answer in Wikipedia (e.g., NaturalQuestions, WebQuestions) - we can check if the model's answer matches the known truth.

- **Summarization Benchmarks with Faithfulness Annotations:** For instance, XSum and CNN/DM summaries have known issues with hallucination. Maynez et al. (2020) annotated XSum model outputs for intrinsic and extrinsic hallucinations [5]. Recent datasets (e.g., from Pagnoni et al. 2021 [29]) include human judgments on factual consistency for many summaries. Using these, one can measure how often a model's summary has unfaithful content and see if mitigation (like a grounded summarizer) lowers that frequency.
- **Knowledge-Grounded Dialogue Benchmarks:** Datasets like Wizard-of-Wikipedia and Holl-E provide dialogues where a model must stick to given knowledge. They often come with metrics like knowledge F1 [30] that measure overlap between the model's response and the gold knowledge. A low precision in this overlap means the model introduced content that was not in the knowledge (hallucination).

- **Machine Translation Hallucination Sets:** There are known cases of hallucination in NMT, often with low-resource language pairs. Some research has test sets where source sentences were perturbed or out-of-domain and they check if the translation outputs irrelevant text [9]. These can be used to evaluate how often a model produces content not present in the source (an extrinsic hallucination in MT context).
- **Domain-specific factuality tests:** e.g., for medical LLMs, one can use questions from medical exams or factual checks where the answers are known. For coding assistants, hallucination might mean producing code that doesn't compile; there, test suites can catch functional hallucinations.

When evaluating a mitigation like RAG or fine-tuning, it's important to test on queries that are challenging and likely to induce hallucination. This can include deliberately out-of-scope questions, ambiguous prompts, or those requiring up-to-date knowledge (which base models might not have learned).

B. Metrics for Hallucination

We have touched on some metrics earlier, but summarizing:

- **Hallucination Rate / Factuality Score:** The simplest metric is the percentage of outputs that contain a hallucination. This typically requires human evaluation or a highly trusted automatic method. For a given test set, you could count how many answers are fully correct vs have any incorrect info.
- **Intrinsic/Extrinsic Breakdown:** If possible, it is insightful to report the breakdown: e.g., "20% of summaries had hallucinations: 5% intrinsic, 15% extrinsic." This requires labeling each hallucinated case as one or the other, which is usually manual.
- **Knowledge F1 / Content Precision and Recall:** These metrics compare the set of facts in the output to the set in the source or reference. For example, Knowledge F1 [30] measures the overlap of factual content (often entities) between a dialogue response and the provided knowledge. A low precision in this overlap means the model introduced content that was not in the knowledge (hallucination).
- **Entailment-based Metrics:** Use a natural language inference (NLI) model to judge if the model's output is entailed by the source (for tasks with source). If the NLI model says the output is not entailed (or contradicted), that's a signal of unfaithfulness. Metrics like FactCC [24] and other BERT-based classifiers effectively do this; some works fine-tune NLI models specifically for factual consistency.
- **Question-Answering based Metrics (QAGS):** Generate questions from the model's output, then see if a QA system can answer them correctly using the source text [26]. If the answers from the source don't match the output, the output likely had unsupported info. This is an indirect but often effective automatic metric for summarization factuality.

- **Human Evaluation Scales:** When possible, use human evaluators to rate outputs on a scale (e.g., 1 to 5) for factual correctness. Define criteria clearly: 5 = no hallucination, fully faithful; 4 = maybe a trivial extrinsic detail added; 3 = some minor incorrect info; 2 = major incorrect info; 1 = almost entirely hallucinated or unusable. This helps gauge severity, not just binary presence.

- **Calibration metrics:** If one aim is to have the model know when it's guessing, one can measure calibration. E.g., the Brier score or Expected Calibration Error (ECE) for the model's predicted probabilities vs actual correctness [14]. For generative models, a variant might be needed (like measuring if when the model says "I am 90% sure," it's correct 90% of the time). Good calibration means fewer unwarranted confident hallucinations.

It is often beneficial to use multiple metrics to get a full picture. Automatic metrics can be noisy or one-dimensional, so confirm with some human assessment on a subset.

C. Experimental Settings and Reporting

We recommend the following when designing experiments to evaluate hallucination:

- **Compare Base vs Mitigated Models:** Always evaluate the original model (without the mitigation strategy) versus the model with the strategy. For example, compare GPT-3 vs GPT-3 + retrieval on the same questions. This directly shows the reduction in hallucination (if any) and helps quantify the benefit.
- **Diverse Test Cases:** Use a mix of easy and hard queries. Some queries that are straightforward factual (which the model likely knows) to establish a baseline of performance, and some deliberately tricky ones. For hallucination study, bias toward those that are challenging.
- **Ablation Studies:** If you introduce a pipeline with multiple components (say retrieval + verification + fine-tuning), perform ablations to see which contribute most. For instance, test retrieval alone, fine-tuning alone, and combined, to evaluate their individual and combined effect on hallucination rate.
- **Measure Impact on Fluency/Other Metrics:** Ensure that efforts to reduce hallucination don't overly degrade language quality or other desired traits. So, measure something like BLEU/ROUGE for summarization (if applicable) or user satisfaction if possible. In many cases, factuality improvements come with minimal quality loss, but it's good to verify. If a mitigation harms the model's ability to answer at all (maybe it refuses too often), note that trade-off.
- **Statistical significance:** Given the variability in generation, use sufficiently large sample sizes and statistical tests if claiming one method is better. Also consider running multiple trials if using stochastic generation to account for randomness.
- **Error Analysis:** Present a brief analysis of common failure modes even after mitigation. For example, maybe with retrieval the model rarely makes up proper nouns

now, but still occasionally mis-states numerical values. Understanding what hallucinations remain can guide future improvements.

As an example, an evaluation for a QA model might look like: 500 questions from TruthfulQA, measure truthful answer percent; 100 questions beyond training knowledge (requiring current events info) to see how it handles unknowns; measure ECE of its self-reported confidence; etc., and then compare those metrics before and after applying retrieval + calibration.

In summary, evaluation should be comprehensive, covering both whether hallucinations are reduced and whether the model remains useful and fluent. By following these protocols, researchers can reliably track progress on making LLMs more factual and identify areas that need more work.

VIII. CONCLUSION AND FUTURE WORK

Hallucination in large language models remains a significant barrier to their deployment in many real-world scenarios that require reliability and factual accuracy. In this paper, we provided a thorough examination of the problem from theoretical foundations to practical solutions. We formalized what it means for an LLM to hallucinate, distinguishing intrinsic contradictions from extrinsic fabrications, and introduced the concept of hallucination risk as a measurable quantity. We discussed how classical learning theory can bound this risk, yet also noted theoretical results implying that some level of hallucination may be innate for general-purpose models, reinforcing the need for ongoing mitigation efforts [11].

On the practical side, we surveyed a spectrum of detection methods (uncertainty-based, calibration-based, and attention-based) and mitigation strategies (RAG, fine-tuning, calibrated decoding, and verification modules). Each approach contributes a piece to the puzzle: retrieval brings grounded knowledge, fine-tuning aligns model behavior with truthfulness, calibration and uncertainty quantification help the model judge when it might be wrong, and verification acts as a safety net to catch mistakes. By integrating these, as illustrated in our proposed workflow, one can build systems that significantly reduce hallucination rates compared to naive LLM usage.

Our recommendations for evaluation serve as a guide to measure progress. It's crucial that the community converges on robust benchmarks and shares best practices for testing factuality. Only through rigorous evaluation can we confidently deploy LLMs in sensitive domains like healthcare, law, or education, where a hallucinated statement could have serious repercussions.

Looking forward, there are several exciting directions for future research. One is **knowledge boundary estimation**: enabling models to explicitly know and indicate the limits of their knowledge (essentially learning a model of their own ignorance). This could involve the model internally predicting whether it has seen sufficient evidence for a query or if it should defer to an external source [4]. Another direction is **dynamic retrieval and reasoning**, where models not only fetch facts but also perform reasoning steps (e.g., using chain-of-thought prompting combined with tool use) to ensure

consistency and correctness of multi-hop answers. Advances in **multi-modal grounding** may also help; for instance, linking text to images or databases to cross-verify information could reduce hallucination (like verifying a generated caption against the actual image content).

From a theoretical standpoint, developing more refined frameworks to analyze why and when hallucinations occur could inform training regimes. Could we characterize certain training distributions or model architectures that inherently minimize hallucinations? The intersection of causal inference and LLM training might offer insights into how models pick up spurious facts and how to mitigate that.

In conclusion, while hallucination in LLMs is a challenging problem, the combination of theoretical understanding and a multifaceted engineering approach provides a promising path to taming it. By continuing to ground models in reality, encourage them to know what they don't know, and rigorously checking their outputs, we move closer to LLMs that can be both creative and consistently truthful. Such models will greatly enhance trust and broaden the safe applicability of AI in society.

REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar *et al.*, “Attention is all you need,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
- [2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. 2019 Conference of the North American Chapter of the ACL (NAACL)*, 2019, pp. 4171–4186.
- [3] T. Brown, B. Mann, N. Ryder *et al.*, “Language models are few-shot learners,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1877–1901.
- [4] Z. Ji, N. Lee, R. Frieske *et al.*, “Survey of hallucination in natural language generation,” *ACM Computing Surveys*, vol. 55, no. 12, p. 248, 2023.
- [5] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, “On faithfulness and factuality in abstractive summarization,” in *Proc. 58th Annual Meeting of the ACL*, 2020, pp. 1906–1919.
- [6] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, “Detecting hallucinations in large language models using semantic entropy,” *Nature Machine Intelligence*, vol. 6, no. 9, pp. 1019–1027, 2024.
- [7] O. Vinyals and Q. Le, “A neural conversational model,” in *Proc. ICML Deep Learning Workshop*, 2015, arXiv:1506.05869.
- [8] P. Koehn and R. Knowles, “Six challenges for neural machine translation,” in *Proc. 1st Workshop on Neural Machine Translation*, 2017, pp. 28–39.
- [9] V. Raunak, A. Menezes, and M. Junczys-Dowmunt, “The curious case of hallucinations in neural machine translation,” in *Proc. 18th International Conference on Spoken Language Translation (IWSLT)*, 2021, pp. 1172–1183.
- [10] Y. Zhang, A. Elgohary, M. Zhao *et al.*, “A survey on hallucination in large language models,” arXiv:2305.11717, 2023.
- [11] Y. Xu, L. Fan, and M. Kankanhalli, “Hallucination is inevitable: An innate limitation of large language models,” arXiv:2310.03992, 2024.
- [12] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 1050–1059.
- [13] N. Kadavath, S. Bai, A. Coner *et al.*, “Language models (mostly) know what they know,” in *Proc. Neural Information Processing Systems (NeurIPS)*, 2022.
- [14] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Proc. 34th International Conference on Machine Learning (ICML)*, 2017, pp. 1321–1330.

- [15] H. Jiang, B. Kassim, H. Nepal, and M. Bansal, “How can we know when language models know? On the calibration of language models for question answering,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 962–977, 2021.
- [16] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” in *Proc. 55th Annual Meeting of the ACL*, 2017, pp. 1073–1083.
- [17] P. Lewis, E. Perez, A. Piktus *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 9459–9474.
- [18] K. Guu, K. Lee, Z. Tong, P. Pasupat, and M. Chang, “REALM: Retrieval-augmented language model pre-training,” in *Proc. 37th International Conference on Machine Learning (ICML)*, 2020, pp. 3929–3938.
- [19] K. Shuster, S. Bhuwan, M. Chen *et al.*, “Knowledge-informed dialogue generation,” in *Proc. 2021 Conference on Empirical Methods in NLP (EMNLP)*, 2021, pp. 1656–1679.
- [20] L. Ouyang, J. Wu, X. Jiang *et al.*, “Training language models to follow instructions with human feedback,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [21] R. Nakano, J. Hilton, S. Balaji *et al.*, “WebGPT: Browser-assisted question-answering with human feedback,” arXiv:2112.09332, 2022.
- [22] S. Desai and G. Durrett, “Calibration of pre-trained transformers,” in *Proc. 2020 Conference on Empirical Methods in NLP (EMNLP)*, 2020, pp. 295–302.
- [23] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” in *Proc. 8th International Conference on Learning Representations (ICLR)*, 2020.
- [24] W. Kryściński, B. McCann, C. Xiong, and R. Socher, “Evaluating the factual consistency of abstractive text summarization,” in *Proc. 2020 Conference on Empirical Methods in NLP (EMNLP)*, 2020, pp. 9332–9346.
- [25] P. Manakul, J. D’Haro, and S. Sakti, “SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models,” in *Proc. 61st Annual Meeting of the ACL*, 2023, pp. 2229–2244.
- [26] X. Wang, P. Gan, Y. Zhang, and J. Zhao, “Asking and answering questions to evaluate the factual consistency of summaries,” in *Proc. 58th Annual Meeting of the ACL*, 2020, pp. 5008–5020.
- [27] O. Honovich, E. Lahav, A. Katz, and J. Goldberg, “TRUE: Re-evaluating factual consistency evaluation,” in *Proc. 2nd Workshop on Document-grounded Dialogue and Conversational QA (DialDoc@ACL)*, 2022, pp. 161–175.
- [28] S. Lin, J. Hilton, and O. Evans, “TruthfulQA: Measuring how models mimic human falsehoods,” in *Proc. 39th International Conference on Machine Learning (ICML)*, 2022, pp. 13667–13690.
- [29] A. Pagnoni, S. Balachandran, and A. Tsvetkov, “Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics,” in *Proc. 2021 Conference on Empirical Methods in NLP (EMNLP)*, 2021, pp. 4818–4837.
- [30] K. Shuster, M. Chen, S. Ju *et al.*, “Knowledge-based metrics for dialogue: Unifying structured and unstructured knowledge,” in *Proc. 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM@ACL)*, 2021, pp. 9–23.
- [31] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “FEVER: a large-scale dataset for fact extraction and verification,” in *Proc. 2018 Conference of the North American Chapter of the ACL (NAACL)*, 2018, pp. 809–819.
- [32] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5574–5584.
- [33] A. Kamath, P. De Cao, and R. Cipolla, “Selective question answering under domain shift,” in *Proc. 58th Annual Meeting of the ACL*, 2020, pp. 5684–5696.
- [34] S. Dhuliawala, M. Komeili, J. Tang *et al.*, “Chain-of-verification reduces hallucination in large language models,” arXiv:2309.11495, 2023.
- [35] H. Zhao, J. Sun, G. Lai *et al.*, “Mitigating hallucination by integrating knowledge graphs into LLMs,” in *Proc. 37th AAAI Conference on Artificial Intelligence*, 2023, pp. 12088–12096.
- [36] P. L. Bartlett and S. Mendelson, “Rademacher and gaussian complexities: Risk bounds and structural results,” *Journal of Machine Learning Research*, vol. 3, pp. 463–482, 2002.
- [37] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.
- [38] D. McAllester, “PAC-Bayesian model averaging,” in *Proc. 12th Annual Conference on Computational Learning Theory (COLT)*, 1999, pp. 164–170.