

A Comparison of LLM Observability Platforms

Zihe Liu^{1,2}

¹AI Verify, Infocomm Media Development Authority

² University of Cambridge

Abstract

This article compares various providers of LLM observability platforms, focusing on issues of reliability and robustness. We examine their evaluation metrics on detecting hallucinations, and measuring robustness against jailbreaks and prompt injection attacks. We also give an overview of state-of-the-art hallucination detection research.

1 Introduction

Reliability and robustness [Yang Liu, Yao, et al., 2024] are core safety issues surrounding deploying LLMs in industry. Current LLMs have seen transformative applications across many industries, with notable use cases being customer service chatbots and internal chatbots for enterprise knowledge management. Especially in high-stakes domains like finance, governance, or healthcare, it becomes crucial to ensure that LLM generate correct, consistent outputs, and are robust to malicious jailbreak attacks potentially exposing sensitive data. IMDA’s partners have surfaced these are their key concerns surrounding LLM applications.

By reliability, we mean producing factually correct and self-consistent results across time and tasks. A core problem LLMs exhibit is *hallucinations*, which refers to generating factually incorrect or content lacking proper context understanding. *Intrinsic hallucinations* occur when the generated output contradicts the input content, and *extrinsic hallucinations* occur when the generated output cannot be verified from the inputs, making false claims about the external world. Intrinsic hallucinations are particularly problematic in industry, where the Retrieval Augmentation Generation (RAG) [Lewis et al., 2021] framework is commonly used to answer queries using an existing internal database. The consequences of giving customers or employees wrong information and acting on them will be severe.

Core robustness issues of practical concern include jailbreaking [J. Wang et al., 2023], where malicious actors could specifically engineer prompts to elicit dangerous behaviour from models or in the worst case extract Personal Identifiable Information (PII) in the training data or RAG database. Prompts injections [Willison, 2024], occurs when untrusted user input is concatenated with a trusted prompt written by the application’s developer. Jailbreaking is a

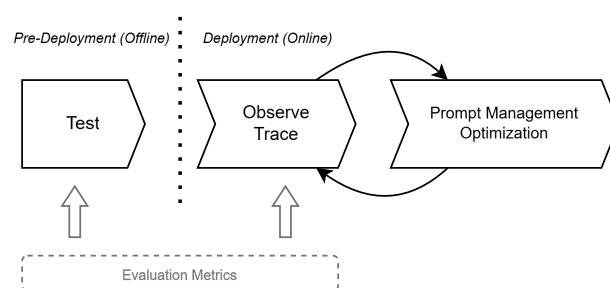


Figure 1: Observability Setup

larger, more complex attack subverting safety filters inbuilt into LLMs, eliciting unwanted behaviour.

To address these issues in deployment, various providers of *observability platforms* have created a suite of tools to monitor and investigate issues of LLMs in production. We give an overview of the various functions different providers offer in Section 2, and current research methods for hallucination detection in Section 3.

2 Overview of Providers

These observability platforms often come as an end-to-end package to test, deploy and monitor applications throughout the LLM development lifecycle. Common functionalities include:

- **Tracing:** Track and visualize LLM interaction in a multi-step chain, log input and output prompts, and metadata
- **Observe/Monitor:** Use evaluation metrics to analyze quality of LLM outputs, and provide insights to debug issues
- **Prompt Management:** Version, track and optimize prompts

Figure 1 illustrates how these functions work together to enhance LLM performance. We concentrate

on the observability aspect, which uses various evaluation metrics to measure qualities of input and output prompts, sometimes token probabilities, to troubleshoot LLM issues and understand why or where certain problems occur. We give an overview of the observability tools of various providers in Table 1, and summarize their functionalities and evaluation metrics for robustness and reliability:

Arize AI helps engineers monitor and evaluate their LLM applications. The platform provides tools for tracing and debugging data flows, evaluating LLM performance, managing prompts, and setting up guardrails for LLM outputs. It includes features like experiment tracking, automated monitoring for issues like hallucinations or PII leaks. The platform is built on OpenTelemetry.

Arize Phoenix offers modules for evaluating LLM outputs and quality of RAG retrieved contexts. These concatenate the query, context and output into a prompt template and asks an LLM to rate them on a custom metric. Usually categorical metrics, Phoenix provides templates for hallucinations, relevance of retrieved context, quality of output, presence of toxic content etc. Phoenix also provides guardrails, which check inputs and outputs for PII, toxic content etc. If a message fails such guardrails, the system will take a corrective action like sending a default response to the user or prompting the LLM to regenerate a response. Arize also offers evaluations for RAG specific applications, notifying developers when there is no documents in the database with a close enough embedding, or when the document retrieved was not helpful.

Langfuse is an open-source observability and tracing tool for LLMs. Users commonly track LLM inference, embedding retrieval, API usage. Langfuse is designed to be production-ready with minimal performance overhead and offers native SDKs for Python and JavaScript.

Langfuse provides options to evaluate traces. Users can write templates for LLM-Judge approaches, which test for hallucinations, relevancy, toxicity etc. Manual annotations allow users to annotate traces and observations by hand. Capturing user feedback allows explicit (e.g., thumbs up/down, 1-5 star rating) or implicit (e.g., time spent on a page, click-through rate, accepting/rejecting a model-generated output, human-in-the-loop) user feedback to evaluate LLM traces. Langfuse also gives users full flexibility to ingest custom evaluation scores. The scoring workflow allows you to run custom quality checks on the output of your workflows at runtime, notably

integrating well with DeepEval.

LangSmith is a platform tightly integrated with popular open-source LLM frameworks LangChain and LangGraph. Core functionalities include tracing, evaluating performance and prompt management to quickly refine prompts.

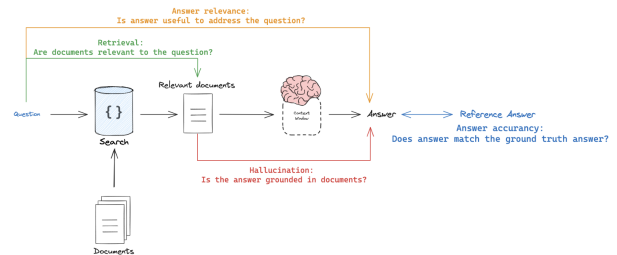


Figure 2: Common metrics for RAG applications. Taken from [LangSmith RAG Evaluators n.d.](#)

Similar to Langfuse, users can write their own prompts for LLM-Judge evaluations, manually evaluate traces, or define their own custom evaluators. For RAG applications, LangSmith has a set of prompts for LLM-Judge evaluators that are common to almost all providers evaluating RAG applications (Figure 2) For an input question, retrieved document, and output answer, we have:

- *Document Relevance* Are documents retrieved relevant to the question?
- *Answer Faithfulness* Is the answer grounded in the retrieved documents?
- *Answer Helpfulness* Does the answer help address the question?
- *Answer Correctness* Is the answer consistent with a reference answer?

Lunary provides observability and prompt management tools for LLM developers. LLM interactions analytics including costs, usage, latency, and errors are tracked, whereas logs and tracing help to track input/output messages and the agent chain.

LangKit is an open-source evaluation metrics toolkit for monitoring language models. Text quality metrics use metrics from `textstat` including readability scores (Flesch Reading Ease, Automated Readability Index), word complexity metrics, and basic text statistics. Text relevance metrics measure the semantic similarity (cosine similarity between text embeddings) between different texts, used for assessing the quality and safety of LLM outputs. One can measure the relevance of output to input prompt via their semantic similarity, and also the safety of outputs via similarity to a known set of jailbreaks or refusals.

LangKit provides a large suite of tools for monitoring security and privacy. Prompt injection detection can be detected via computing embedding similarity as described earlier, and using proactive injection detection [Yupei Liu et al., 2024]. Hallucination detection is done through the Self-Check GPT [Manakul et al., 2023] framework. Microsoft Presidio toolkit is used for regex-based methods for PII detection. Toxicity classifiers (fine-tuned DistilBERT) are used to detect toxicity.

Traceloop is built on top of OpenTelemetry to monitor, debug and test the quality of LLM outputs. Semantic evaluation metrics tests for answer faithfulness and relevancy, while safety metrics check for PII and toxic content. We are unable to verify the framework behind these evaluations from the documentation or source code.

Ragas is a comprehensive suite of tools [Es et al., 2023] containing evaluation metrics and intelligent test generation. Ragas contains many RAG specific evaluations. These include context precision (measuring proportion of relevant chunks in the retrieved context), context recall (how much of the answer can be attributed to retrieved context), and noise sensitivity (proportion of response claims that are incorrect). Ragas uses both LLM-based and NLP methods to extract statements and compare attribution. The answer relevancy is defined as the mean cosine similarity of the original user input to a number of artificial questions, which were generated based on the response. The faithfulness metric uses Vectara’s HHEM-2.1-Open classifier model to detect hallucinations, and measures how many claims in the response were hallucinated given the retrieved context. Traditional NLP metrics for comparing semantic similarity, such as BLEU and ROUGE, are also provided.

Confident AI helps teams evaluate, monitor, and quality-assure their LLM applications throughout the entire development lifecycle, from testing to production deployment, by providing cloud infrastructure, proprietary evaluation models, and tools for tracking. It builds upon the open-source DeepEval framework, consisting of many research-backed metrics for hallucination detection, and detecting vulnerabilities according to the OWASP top 10 risks for LLMs. G-Eval [Yang Liu, Iter, et al., 2023] is a general framework for LLM-Judge using chain-of-thought reasoning and probability-weighting to give a final score. This metric can be adapted for any criteria. Uniquely, DeepEval has many conversational metrics such as conversational G-Eval, conversation relevancy, and

knowledge retention.

Guardrails AI Guardrails AI helps developers build reliable AI applications by validating and safeguarding LLM inputs and outputs through customizable validators and guards. It offers over 60 metrics to detect and prevent issues like hallucinations, toxic language, and data leaks. Hallucination metrics include provenance (answer faithfulness) and embedding similarity. Safety evaluations include toxicity detection, PII and data leak prevention, and jailbreaking detection mostly using fine-tuned versions of BERT.

Datadog is a cloud monitoring and analytics platform that helps companies track the performance of their entire technology stack, from servers and applications to user experience and security. LLM Observability is a new function allowing developers to monitor and troubleshoot their LLM applications. Evaluation metrics for quality evaluations cover answer relevancy and failure to answer. Safety evaluations covering toxicity, prompt injection detection, and PII (using the rule-based Sensitive Data Scanner). The full Ragas library is also integrated.

Galileo is a platform that helps teams build and maintain trustworthy AI applications, with a focus on evaluation, monitoring, and protection of LLMs. Their platform includes tools for testing and experimenting with AI models before deployment (Evaluate), real-time monitoring of AI applications in production (Observe), and implementing safeguards (Protect). Galileo’s Guardrail Metrics power all these tools, which are a suite of industry-standard and in-house research metrics. Output quality metrics using LLM-Judge include correctness (open domain extrinsic hallucinations), instruction adherence (answer helpfulness), ground truth adherence (answer correctness), and completeness (answer faithfulness). RAG quality metrics include context adherence (closed domain intrinsic hallucinations), chunk attribution and chunk utilization. Note that many metrics use Galileo’s in-house research Chainpoll [Friel et al., 2023] and LUNA [Song et al., 2024] to compute relevance of retrieved context to the output, on a token level. Uncertainty measures using token probabilities are also provided via input prompt perplexity and output uncertainty. Classic NLP metrics based on n-gram similarity such as BLEAU and ROUGE are also used. For PII/toxicity/prompt injection detection, Galileo utilizes a small language model trained on open source datasets to classify these.

Braintrust provides tools for iterative experimentation, performance evaluation, real-time monitoring, and data management of LLM-software. Braintrust performs evaluations by combining test datasets, task functions that generate outputs, and scoring functions that assess those outputs. The main types of evaluations in the `autoevals` library include LLM-Judge evaluations, heuristic evaluations like Levenshtein distance or Jaccard distance, statistical evaluations using BLEU and ROUGE, embedding-based evaluations that measure semantic similarity, and RAG-specific evaluations (LLM-Judge) that assess context usage and faithfulness.

Weave is a lightweight toolkit developed by Weights & Biases that helps track and evaluate LLM applications. It provides a simple way to track application changes and performance over time by decorating Python functions, while automatically logging metadata, token usage, and costs for various LLM providers. Evaluation metrics include output quality metrics (hallucination detection and summarization quality, using LLM-Judge), embedding similarity metrics, and methods from Ragas.

Athina helps teams build, test, and monitor production-grade AI applications. Hallucination detection uses LLM-judge, while summary-specific hallucinations use a LLM-Judge incorporating 10 key questions to gauge summary quality. Safety metrics for prompt injection and PII detection use open-source fine-tuned DeBERTa models. Harmful/toxic content detection uses OpenAI's `omni-moderation-latest` which is a classifier based off GPT-4o. The Guardrails AI and Ragas library are also tightly integrated.

Humanloop enables product teams to build robust LLM applications, through evaluation driven development and collaborative development between domain experts and engineers. Humanloop evaluators take LLM-generated logs and returns a judgement. Evaluators can be run online to monitor model performance during deployment, or run offline during pre-deployment testing. Code evaluators include rule based approaches like cost, latency, and regex rules on the output. AI evaluators use the LLM-Judge framework for factuality and helpfulness, while an intuitive UI is provided for human evaluation.

HoneyHive AI is an LLM observability platform providing comprehensive tools for logging execution data, evaluating application quality both offline and online, collecting user feedback, and managing artifacts like prompts and datasets. Limited evaluation

metrics are provided, mostly custom Python (rule-based) and LLM-Judge metrics.

PromptLayer is a platform for managing, versioning, and evaluating LLM prompts that allows teams to create and iterate on prompts through a visual dashboard while programmatically retrieving them in their applications. It provides tools for prompt engineering, including A/B testing, evaluation pipelines, analytics, and collaboration features. Rule based evaluations (pattern matching), LLM-Judge evaluations and embedding similarity measures are provided.

3 Hallucination Detection

As hallucinations is a core reliability issue, we give an overview of various hallucination detection methods gathered directly from [Chakraborty et al., 2024], [Luo et al., 2024] and [Yang Liu, Yao, et al., 2024]. We focus on the detection task (excluding mitigation), and on the natural language set up, for this overview. As in [Chakraborty et al., 2024], we classify methods into 3 types: white-box, grey-box and black-box methods. White-box methods assume access to the output text sequence, output token probabilities, weights, and activations of the model. Grey-box methods only access the output sequence and token probabilities. As most LLM providers only give input and output text, black-box methods assume access to this only. Crucially, all these methods require LLMs to have some self-awareness of their uncertainty levels, which we do have evidence of [Kadavath et al., 2022].

3.1 White-box Methods

Hidden States These methods utilize the intermediate hidden states (or activations). [Azaria et al., 2023] use a simple classifier trained on the intermediate activations to output the probability of the generated claim being truthful. [Song et al., 2024] use a discrete time Markov chain (DTMC) to encode the activations into a lower dimensional space, cluster them into abstract discrete states, and learn transition functions between states. They then identify untrustworthy states and transitions, and use these to determine trustworthiness during inference time. [C. Chen et al., 2024] samples multiple outputs and measure the semantic diversity among activations.

Attention Weights Attention scores contain information on the relative importance paid to previous tokens when predicting the next token. [Huang et al.,

Table 1: Functionalities of various providers. Evaluation metrics are for both hallucination detection and jailbreaking and PII detection, using terminology from Section 3. Refer to Section 2 for breakdown of metrics for each provider.

Provider	Type	Tracing	Prompt Management	Evaluation Metric Types
Arize	Open-Source	•	•	LLM-Judge
Langfuse	Open-Source	•	•	LLM-Judge Human Annotation User Feedback Custom Metrics
LangSmith	Open-Source	•	•	LLM-Judge Human Annotation Custom Metrics
Lunary	Open-Source	•	•	-
LangKit	Open-Source	-	-	LLM-Judge Analyzing Samples Embedding Similarity Proxy Model PII Detection (Regex)
TraceLoop	Open-Source	•	•	LLM-Judge Embedding Similarity Proxy Model
Ragas	Open-Source	-	-	LLM-Judge Embedding Similarity Proxy Model
Confident AI	Open-Source	-	•	LLM-Judge Proxy Model PII Detection (Regex) Custom Metrics
Guardrails AI	Open-Source	-	-	LLM-Judge Proxy Model PII Detection (Regex)
Datadog	Commercial	•	-	LLM-Judge PII Detection (Regex)
Galileo	Commercial	•	-	LLM-Judge Analyzing Samples Proxy Model Concept Probabilities PII Detection (Regex) Custom Metrics
BrainTrust	Commercial	•	•	LLM-Judge Embedding Similarity Custom Metrics
Weave	Commercial	•	•	LLM-Judge Embedding Similarity
Athina	Commercial	•	•	LLM-Judge Proxy Model
HumanLoop	Commercial	•	•	LLM-Judge Embedding Similarity Human Annotation Custom Metrics
HoneyHive	Commercial	•	•	LLM-Judge Custom Metrics
PromptLayer	Commercial	-	•	LLM-Judge Embedding Similarity

Table 2: Summary of hallucination detection methods. Modified from [Chakraborty et al., 2024]. Deployment scenarios are split into question-answering (QA), information retrieval (IR), image captioning (IC) tasks. Custom datasets is abbreviated as CD.

Modality	Method Type	Deployment	Paper & Datasets
White-box	Hidden States	QA	[Azaria et al., 2023] CD
		QA	[Song et al., 2024] TruthfulQA
		QA	[C. Chen et al., 2024] TruthfulQA, CoQA etc
	Attention Weights	IC	[Huang et al., 2024] MSCOCO, Visual Genome
		QA	[Chuang et al., 2024] Natural Questions, XSum
	Honesty Alignment	QA	[Lin et al., 2022] CD
Grey-box	Concept Probabilities	IR/QA	[Varshney et al., 2023] HotpotQA
		IR/QA	[Sriramanan et al., 2024] FAVA, RAGTruth
		IR/QA	[Valentin et al., 2024] TriviaQA, HaluEval
	Conformal Prediction	IR/QA	[Quach et al., 2024] MIMIC-CXR, CNN/DM (Hermann et al., 2015), TriviaQA
		QA	[Kumar et al., 2023] MMLU
Black-box	LLM-Judge	IR/QA	[Friel et al., 2023] TriviaQA, COVID-QA
		IR/QA	[Yang Liu, Iter, et al., 2023] SummEval, Topical-Chat
		IR/QA	[Belyi et al., 2024] CD
		IR/QA	[Ravi et al., 2024] RAGTruth, COVID-QA
		IR/QA	[B. Wang et al., 2024] CD
		IR/QA	[Es et al., 2023] CD
		IR/QA	[Saad-Falcon et al., 2024] CD
		QA	[Dhuliawala et al., 2023] MultiSpanQA, CD
	Analyzing Samples	IR/QA	[Manakul et al., 2023] CD
		IR/QA	[Du et al., 2023] GSM8K, MMLU
		IR/QA	[Mündler et al., 2024] Quest, MultiSpanQA, FActScore
		IR/QA	[Zha et al., 2023] SNLI, SQuAD V2
	Adversarial Prompting	IR/QA	[Yu et al., 2024] Natural Questions, CD
	Proxy Model	IR/QA	[Y. Chen et al., 2024] SQuAD, HotpotQA, TriviaQA
		IR/QA	[Pacchiardi et al., 2023] CD
	Grounding Knowledge	IR/QA	[S. Zhang et al., 2024] FuzzyQA

2024] find *aggregation patterns*, specific patterns in the attention score matrix that suggests over-importance on uninformative tokens, and hallucinations often follow. [Chuang et al., 2024] measure the lookback ratio, relative attention scores paid to input tokens versus new tokens generated, and trains a classifier on these lookback ratios to identify hallucinations.

Honesty Alignment [Lin et al., 2022] fine-tune GPT3 to output its certainty that its prediction is correct.

3.2 Grey-box Methods

Concept Probabilities This approach aggregates individual token-level probabilities (often using log probabilities) to estimate the uncertainty for the entire sentence. [Varshney et al., 2023] show lower token probabilities lead to a higher tendency to hallucinate. Various measures are possible:

For example, $\max$ ¹ and average aggregation² can be used to estimate the uncertainty of a sentence:

$$\text{Max}(-\log p) = \max_j (-\log p_j),$$

$$\text{Avg}(-\log p) = -\frac{1}{J} \sum_j \log p_j,$$

where p_j is the probability of predicting the token at position j , of a sentence of length J . We can also leverage entropy, $H(X_j) = -\sum p(x_j) \log p(x_j)$, using the entire probability distribution to estimate sentence uncertainty:

$$\text{Max}(H) = \max_j [\mathcal{H}_j],$$

$$\text{Avg}(H) = \frac{1}{J} \sum_j \mathcal{H}_j,$$

Recently, [Sriramanan et al., 2024] use a combination of perplexity, logit entropy, and activations to estimate uncertainty. [Valentin et al., 2024] also introduce multi-scoring to incorporate various uncertainty measures, giving higher accuracy than individual measures.

Conformal Prediction This works by using token probabilities to give statistical guarantees that a given set of outputs contains the correct output. [Quach et al., 2024] samples outputs and adds them to a prediction set until user-defined confidence guarantees

¹ Taking the maximum negative log probability is equivalent to finding the minimum probability

² Taking average log probabilities is equivalent to measuring sentence perplexity

are met. [Kumar et al., 2023] use a similar method for answering multiple choice questions.

3.3 Black-box Methods

LLM-Judge The simplest method passes input and outputs to an LLM, asking LLMs to detect hallucinations, possibly on a binary metric or multi-class labels. This is by far the most common industry method. [Friel et al., 2023] ChainPoll from Galileo AI uses chain-of-thought prompting on LLMs to detect hallucinations on a binary yes/no metric, and averages the result of multiple samples. Similarly, G-Eval [Yang Liu, Iter, et al., 2023] uses chain-of-thought prompting and a probability weighted average to give a score from the LLM-Judge. Later on, Galileo AI [Belyi et al., 2024] fine-tune DeBERTA to output the probability of entailment as in the NLI setup, through LUNA. [Ravi et al., 2024] use supervised fine-tuning on LLAMA-3 to output the faithfulness of the output to the input context, along with a reasoning chain. Similarly, [B. Wang et al., 2024] use DPO to fine-tune Mistral-7B to output a reasoning chain and whether the output claim is supported by the input context. RAGAS [Es et al., 2023] breaks down the output into statements, and measures the faithfulness of these to detect hallucinations. [Saad-Falcon et al., 2024] train 3 LLM judges on context relevance, answer faithfulness and answer relevance to detect hallucinations, in ARES. [Dhuliawala et al., 2023] uses chain-of-thought prompting to generate possible verification questions, thereafter cross-checking with answers to verifications to produce a more robust final answer.

Analyzing Samples These methods sample multiple outputs, using similarity between outputs as a gauge for uncertainty. SelfCheckGPT [Manakul et al., 2023] samples multiple outputs, and use different consistency measures to compare the consistency of these outputs. These include BERTScore [T. Zhang et al., 2020], which uses cosine similarity between sentence embeddings, and NLI [Maynez et al., 2020] which uses a classifier to predict consistency among sentences. [Du et al., 2023] uses an ensemble of models to generate independent outputs, critique one another’s outputs, and refine them. [Mündler et al., 2024] generates multiple sentences about key concepts in an output, and uses an analyzer LLM to gauge consistency among sentences on the same concept. [Zha et al., 2023] uses an alignment function to check whether sentences in the output are supported by the input context.

Adversarial Prompting [Yu et al., 2024] perturbs the input evidence with incorrect statements, and checks if the model follows the new incorrect information to gauge hallucination.

Proxy Model These are non-LLM-based external classifiers. [Y. Chen et al., 2024] trains an external classifier to determine hallucination purely via input/output. [Pacchiardi et al., 2023] prompts the LLM with a set of binary questions, and concatenates all answers into a single embedding that is fed to an external classifier.

Grounding Knowledge Similar to RAG, these methods use evidence from an external knowledge base to identify hallucinations. [S. Zhang et al., 2024] attempts to map a user’s questions to relevant attributes in the knowledge base. If the model is uncertain about the mapping, it generates clarifying questions, and user clarifications are used along with the knowledge-base information to generate outputs.

References

- Liu, Yang, Yuanshun Yao, et al. (2024). *Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models’ Alignment*. arXiv: 2308.05374 [cs.AI]. URL: <https://arxiv.org/abs/2308.05374>.
- Lewis, Patrick et al. (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. arXiv: 2005.11401 [cs.CL]. URL: <https://arxiv.org/abs/2005.11401>.
- Wang, Jindong et al. (2023). *On the Robustness of ChatGPT: An Adversarial and Out-of-distribution Perspective*. arXiv: 2302.12095 [cs.AI]. URL: <https://arxiv.org/abs/2302.12095>.
- Willison, Simon (2024). *Prompt injection and jailbreaking are not the same thing*. URL: <https://simonwillison.net/2024/Mar/5/prompt-injection-jailbreaking/>.
- LangSmith RAG Evaluators (n.d.). <https://docs.smith.langchain.com/assets/images/rag-types-4ff470c7c97ea68ec46956326513fb52.png>.
- Liu, Yupei et al. (2024). *Formalizing and Benchmarking Prompt Injection Attacks and Defenses*. arXiv: 2310.12815 [cs.CR]. URL: <https://arxiv.org/abs/2310.12815>.
- Manakul, Potsawee, Adian Liusie, and Mark J. F. Gales (2023). *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. arXiv: 2303.08896 [cs.CL]. URL: <https://arxiv.org/abs/2303.08896>.
- Es, Shahul et al. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. arXiv: 2309.15217 [cs.CL]. URL: <https://arxiv.org/abs/2309.15217>.
- Liu, Yang, Dan Iter, et al. (2023). *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. arXiv: 2303.16634 [cs.CL]. URL: <https://arxiv.org/abs/2303.16634>.
- Friel, Robert and Atindriyo Sanyal (2023). *Chainpoll: A high efficacy method for LLM hallucination detection*. arXiv: 2310.18344 [cs.CL]. URL: <https://arxiv.org/abs/2310.18344>.
- Song, Da et al. (2024). *LUNA: A Model-Based Universal Analysis Framework for Large Language Models*. arXiv: 2310.14211 [cs.LG]. URL: <https://arxiv.org/abs/2310.14211>.
- Chakraborty, Neeloy, Melkior Ornik, and Katherine Driggs-Campbell (2024). *Hallucination Detection in Foundation Models for Decision-Making: A Flexible Definition and Review of the State of the Art*. arXiv: 2403.16527 [cs.AI]. URL: <https://arxiv.org/abs/2403.16527>.
- Azaria, Amos and Tom Mitchell (2023). *The Internal State of an LLM Knows When It’s Lying*. arXiv: 2304.13734 [cs.CL]. URL: <https://arxiv.org/abs/2304.13734>.
- Chen, Chao et al. (2024). *INSIDE: LLMs’ Internal States Retain the Power of Hallucination Detection*. arXiv: 2402.03744 [cs.CL]. URL: <https://arxiv.org/abs/2402.03744>.
- Huang, Qidong et al. (2024). *OPERA: Alleviating Hallucination in Multi-Modal Large Language Models via Over-Trust Penalty and Retrospection-Allocation*. arXiv: 2311.17911 [cs.CV]. URL: <https://arxiv.org/abs/2311.17911>.
- Chuang, Yung-Sung et al. (2024). *Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using Only Attention Maps*. arXiv: 2407.07071 [cs.CL]. URL: <https://arxiv.org/abs/2407.07071>.
- Lin, Stephanie, Jacob Hilton, and Owain Evans (2022). *Teaching Models to Express Their Uncertainty in Words*. arXiv: 2205.14334 [cs.CL]. URL: <https://arxiv.org/abs/2205.14334>.
- Varshney, Neeraj et al. (2023). *A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation*. arXiv: 2307.03987 [cs.CL]. URL: <https://arxiv.org/abs/2307.03987>.
- Sriramanan, Gaurang et al. (2024). “LLM-Check: Investigating Detection of Hallucinations in Large Language Models”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=LYx4w3CAgy>.

- Valentin, Simon et al. (2024). *Cost-Effective Hallucination Detection for LLMs*. arXiv: 2407.21424 [cs.CL]. URL: <https://arxiv.org/abs/2407.21424>.
- Quach, Victor et al. (2024). *Conformal Language Modeling*. arXiv: 2306.10193 [cs.CL]. URL: <https://arxiv.org/abs/2306.10193>.
- Kumar, Bhawesh et al. (2023). *Conformal Prediction with Large Language Models for Multi-Choice Question Answering*. arXiv: 2305.18404 [cs.CL]. URL: <https://arxiv.org/abs/2305.18404>.
- Belyi, Masha et al. (2024). *Luna: An Evaluation Foundation Model to Catch Language Model Hallucinations with High Accuracy and Low Cost*. arXiv: 2406.00975 [cs.CL]. URL: <https://arxiv.org/abs/2406.00975>.
- Ravi, Selvan Sunitha et al. (2024). *Lynx: An Open Source Hallucination Evaluation Model*. arXiv: 2407.08488 [cs.AI]. URL: <https://arxiv.org/abs/2407.08488>.
- Wang, Binjie et al. (2024). *Halu-J: Critique-Based Hallucination Judge*. arXiv: 2407.12943 [cs.CL]. URL: <https://arxiv.org/abs/2407.12943>.
- Saad-Falcon, Jon et al. (2024). *ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems*. arXiv: 2311.09476 [cs.CL]. URL: <https://arxiv.org/abs/2311.09476>.
- Dhuliawala, Shehzaad et al. (2023). *Chain-of-Verification Reduces Hallucination in Large Language Models*. arXiv: 2309.11495 [cs.CL]. URL: <https://arxiv.org/abs/2309.11495>.
- Du, Yilun et al. (2023). *Improving Factuality and Reasoning in Language Models through Multiagent Debate*. arXiv: 2305.14325 [cs.CL]. URL: <https://arxiv.org/abs/2305.14325>.
- Mündler, Niels et al. (2024). *Self-contradictory Hallucinations of Large Language Models: Evaluation, Detection and Mitigation*. arXiv: 2305.15852 [cs.CL]. URL: <https://arxiv.org/abs/2305.15852>.
- Zha, Yuheng et al. (2023). *AlignScore: Evaluating Factual Consistency with a Unified Alignment Function*. arXiv: 2305.16739 [cs.CL]. URL: <https://arxiv.org/abs/2305.16739>.
- Yu, Xiaodong et al. (2024). *ReEval: Automatic Hallucination Evaluation for Retrieval-Augmented Large Language Models via Transferable Adversarial Attacks*. arXiv: 2310.12516 [cs.CL]. URL: <https://arxiv.org/abs/2310.12516>.
- Chen, Yuyan et al. (2024). *Hallucination Detection: Robustly Discerning Reliable Answers in Large Language Models*. arXiv: 2407.04121 [cs.CL]. URL: <https://arxiv.org/abs/2407.04121>.
- Pacchiardi, Lorenzo et al. (2023). *How to Catch an AI Liar: Lie Detection in Black-Box LLMs by Asking Unrelated Questions*. arXiv: 2309.15840 [cs.CL]. URL: <https://arxiv.org/abs/2309.15840>.
- Zhang, Shuo et al. (2024). *The Knowledge Alignment Problem: Bridging Human and External Knowledge for Large Language Models*. arXiv: 2305.13669 [cs.CL]. URL: <https://arxiv.org/abs/2305.13669>.
- Luo, Junliang et al. (2024). *Hallucination Detection and Hallucination Mitigation: An Investigation*. arXiv: 2401.08358 [cs.CL]. URL: <https://arxiv.org/abs/2401.08358>.
- Kadavath, Saurav et al. (2022). *Language Models (Mostly) Know What They Know*. arXiv: 2207.05221 [cs.CL]. URL: <https://arxiv.org/abs/2207.05221>.
- Zhang, Tianyi et al. (2020). *BERTScore: Evaluating Text Generation with BERT*. arXiv: 1904.09675 [cs.CL]. URL: <https://arxiv.org/abs/1904.09675>.
- Maynez, Joshua et al. (2020). *On Faithfulness and Factuality in Abstractive Summarization*. arXiv: 2005.00661 [cs.CL]. URL: <https://arxiv.org/abs/2005.00661>.