International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Hallucination Detection and Reduction in Large Language Models: A Controlled Comparison Protocol

Sunny Radhakrishna[1], Vijay Das[1], Prem Tamse[1], Dr. Supriya Kamale[2]

[1]DY Patil Deemed to be University, School of Humanities & Sciences, India
[2]Guide, DY Patil Deemed to be University, School of Humanities & Sciences, India

---

Abstract - *Large language models (LLMs) produce fluent text that is sometimes factually wrong or unsupported by the source, a failure called hallucination. Published detection and mitigation methods are rarely compared under one protocol: studies differ in model, benchmark, correctness labelling, decoding and metric, so it is unclear how much of a reported gain comes from the method and how much from the setup. This paper specifies a controlled, reproducible comparison for open-weight instruction-tuned LLMs on public benchmarks (TruthfulQA, HaluEval and a long-form biography task). Three detectors (predictive entropy, sampling-consistency checking and semantic entropy) are crossed with three reduction strategies (retrieval-augmented generation, chain-of-verification and detector-triggered abstention) in a full grid against an unmitigated baseline. Thresholds are fitted on a development split only, and every configuration is evaluated on the same test questions with bootstrap intervals over five seeds. The paper also synthesises published evidence for each method and states, for every research question, which outcome would contradict the expectation.* ***No new experimental measurements are reported****: the results section holds reporting templates, and the code has not yet been executed. The contribution is an auditable protocol that separates the effect of the detector, the mitigation and the model.*

Key Words: hallucination, large language models, semantic entropy, retrieval-augmented generation, uncertainty estimation, factuality, reproducibility

1. INTRODUCTION

LLMs are used for question answering, summarisation and support, where a confident but false statement can mislead or cause harm. Surveys define hallucination as output that is nonsensical or unfaithful to a supplied source or to established facts [1], [2]. Two tasks follow. Detection asks whether an answer is likely false, so that it can be flagged or withheld. Reduction asks how to make false answers rarer, for example by grounding the model in retrieved text [3] or by making it check its own draft [4]. The two interact: a detector can trigger abstention, and a mitigation changes the answers the detector sees.

A useful system must catch many hallucinations (high recall) without flagging or refusing correct answers too often (adequate precision). A ranking score alone hides this, because a threshold is still needed, and a mitigation can lower the hallucination rate simply by answering fewer questions. This paper is an empirical protocol and evidence synthesis, not a new algorithm. It fixes benchmarks, models, decoding and metrics, varies detector and mitigation in a full grid, and reports uncertainty. The aim is not to name a universally best method but to quantify behaviour under a clearly specified protocol.

2. PROBLEM STATEMENT

Studies of hallucination reach different conclusions because they change the model, benchmark, labelling, temperature, number of samples and metric together. Detection papers often report a threshold-free score on one dataset, while mitigation papers report a factuality rate that abstention can improve. The present study fixes benchmark, split, decoding, labelling and metrics, then varies detector, mitigation and model in a full grid so that their effects can be separated.

3. RESEARCH QUESTIONS

- **RQ1.** What hallucination rate do unmitigated open-weight LLMs show on the chosen benchmarks under a common protocol?
- **RQ2.** Which of predictive entropy, sampling consistency and semantic entropy gives the best AUROC and AUPRC?
- **RQ3.** Do retrieval-augmented generation, chain-of-verification and abstention reduce the hallucination rate relative to the baseline?
- **RQ4.** Which error type is more frequent for each detector: missed hallucinations or false alarms?
- **RQ5.** Are detection errors balanced across question categories?
- **RQ6.** How large is the effect of the mitigation strategy compared with the choice of model?
- **RQ7.** Do methods differ by more than seed and resampling variation?
- **RQ8.** Can the pipeline be specified well enough to be reproduced?

4. RESEARCH OBJECTIVES

- To build a reproducible pipeline for detecting and reducing hallucination on public benchmarks.
- To fit thresholds on a development split only.

- To compare three detectors on identical generations and three mitigations against a baseline for every model.
- To evaluate AUROC, AUPRC, hallucination rate, answered accuracy, abstention rate and cost.
- To analyse error distribution across categories and to quantify strategy versus model effects with confidence intervals.
- To document the pipeline step by step, with code, so that results can be reproduced.

5. RELATED WORK

Table -1: Related work

Reference	Focus	Approach	Relevance / limitation
Ji et al. (2023) [1]; Huang et al. (2023) [2]	Hallucination surveys	Taxonomy of causes, detection and mitigation	Frame the problem; no common experiment
Maynez et al. (2020) [5]	Summarisation	Human study of faithfulness and factuality	Fluent summaries often contain unsupported content
Lin et al. (2022) [6]	TruthfulQA	Questions that elicit imitative falsehoods	Main short-answer benchmark; free-text labelling is hard
Li et al. (2023) [7]	HaluEval	Generated and annotated hallucinated samples	LLM-generated samples may carry stylistic cues
Min et al. (2023) [8]	Long-form text	FActScore: atomic-fact support rate	Fine-grained; depends on corpus and judge
Manakul et al. (2023) [9]	Detection	SelfCheckGPT: agreement across samples	No external knowledge needed; blind to consistent errors
Kuhn et al. (2023) [10]; Farquhar et al. (2024) [11]	Uncertainty	Semantic entropy over meaning clusters	Removes paraphrase effects; needs several samples
Kadavath et al. (2022) [12]; Guo et al. (2017) [13]	Self-knowledge, calibration	Models estimate own correctness; temperature scaling	Confidence carries signal but needs calibration
Lewis et al. (2020) [3]	Retrieval	Retrieval-augmented generation	Grounds answers; retrieval errors can mislead
Dhuliawala et al. (2023) [4]	Verification	Chain-of-verification	Reduces hallucination; multiplies calls
Saito and Rehmsmeier (2015) [14]	Evaluation	Precision-recall versus ROC	Motivates reporting AUPRC

The literature shows that several detectors extract usable signal from model uncertainty or sample agreement, and that grounding and self-verification can lower hallucination on the tasks where they were tested. A limitation is that detectors and mitigations are usually evaluated on different tasks with different labels, and a lower rate may come from answering less. The present study treats detector, mitigation and model as explicit variables.

6. RESEARCH GAP AND STUDY CONTRIBUTION

The gap is not the absence of hallucination methods, which are well established, but the lack of a controlled comparison in which detector, mitigation and model are separate factors and the cost of answering less is reported.

Table -2: Comparison with common practice

Aspect	Common practice in comparable studies	This study
Benchmark	Varies by study	Fixed public benchmarks, same questions for every configuration
Labels	Different judges and thresholds	One documented procedure with a human audit
Thresholds	Sometimes tuned on test data	Fitted on a development split only
Mitigation reporting	Hallucination rate alone	Rate with abstention rate and answered accuracy
Detector metrics	One ranking score	AUROC, AUPRC, calibration, error counts at a fixed threshold

Aspect	Common practice in comparable studies	This study
Uncertainty	Single run	Bootstrap intervals and five seeds
Reproducibility	Prompts partly unreported	Prompts, seeds, versions and code documented

7. BENCHMARK AND DATASET DESCRIPTION

Three public benchmarks cover short answers, annotated hallucinations and long-form text. Sizes are those reported by the benchmark authors; the counts actually used must be recorded at run time.

Table -3: Benchmarks

Benchmark	Content	Reported size	Role
TruthfulQA [6]	Questions in categories designed to elicit falsehoods	817 questions, 38 categories	Primary detection and reduction benchmark
HaluEval [7]	Correct and hallucinated samples across several tasks	About 35,000 samples	Detection benchmark with supplied labels
Biography generation [8]	Long-form biographies scored by fact support	Entity list drawn by the authors	Long-form factuality with FActScore

The benchmarks are static and some reference answers may be contested or outdated, so results describe behaviour on these benchmarks, not deployed performance. HaluEval samples were produced with an LLM, so detectors may learn the style of generated hallucinations [7].

8. METHODOLOGY

The pipeline has seven steps: (1) load a benchmark and split it into development and test parts; (2) generate one greedy answer and N sampled answers per question and model; (3) label every greedy answer as correct or hallucinated; (4) score each answer with each detector; (5) apply each reduction strategy; (6) fit thresholds on the development split and evaluate all configurations on the test split; (7) compute metrics, bootstrap intervals and the error analysis.

8.1 Sampling, Prompting and Decoding

All questions are used. Each benchmark is split 20% development and 80% test, stratified by category, seed 42. The development part is used only for thresholds and prompt checks. Sampled answers are repeated with seeds 42 to 46, and the test set is resampled 1,000 times for bootstrap intervals. Drawing several sampled answers per question is related to self-consistency decoding [15].

Table -4: Prompting and decoding settings

Setting	Value
Answer prompt	One fixed zero-shot instruction, stored with the code
Greedy answer	Temperature 0, up to 64 new tokens (512 for biographies)
Sampled answers	N = 10, temperature 1.0, top-p 1.0
Models	Three open-weight instruction-tuned LLMs of 7B to 8B parameters, named with checkpoint revisions by the authors

8.2 Detectors and Reduction Strategies

Table -5: Detection and reduction methods

Method	Description	Cost per question
Predictive entropy	Mean token negative log-probability of the greedy answer	1 generation
Sampling consistency	Contradiction probability between samples and the greedy answer by an NLI model [9]	N + 1 generations plus NLI
Semantic entropy	Entropy over clusters of mutually entailing samples [10], [11]	N generations plus NLI
Baseline	Greedy answer, no mitigation	1
Retrieval-augmented generation	Top k = 3 passages from a fixed corpus added to the prompt [3]	1 plus retrieval
Chain-of-verification	Draft, verification questions, independent answers, revision [4]	About 5
Abstention	Withhold the answer when the detector score exceeds the development threshold	Detector cost

Abstention can lower the hallucination rate trivially, so it is always reported with the abstention rate and the accuracy on answered questions.

8.3 Labelling and Evaluation Measures

Short answers are labelled against the benchmark's correct and incorrect reference answers with an NLI model, and 100 labels per benchmark are audited by two human raters (Cohen's kappa reported). Long-form answers use FActScore [8].

Table -6: Evaluation measures

Metric	Definition / use
AUROC, AUPRC	Threshold-free ranking quality of the detector score against the hallucination label [14]
Hallucination rate	Hallucinated, unwithheld answers divided by all test questions
Answered accuracy	Correct answers among answered questions
Abstention rate	Withheld answers divided by all test questions
Precision, recall, F1	Hallucination class at the development-fitted threshold
Expected calibration error	Confidence-accuracy gap over bins [13]
Cost	Generations and wall-clock time per question

8.4 Reproducibility Record

Record: seeds (42; 42 to 46 for sampling), benchmark versions and counts used, split sizes, model and NLI checkpoint revision hashes, retrieval corpus snapshot, library versions (Python, PyTorch, transformers, scikit-learn [16]) and GPU type. No versions are given here because nothing has been run.

8.5 Step-by-Step Implementation with Code

The code is listed in execution order for TruthfulQA and **has not been executed**; it must be run and audited before any result is reported. Categories with fewer than two questions must be merged before the stratified split.

```
# ---- Load data, split, load model and NLI model ----
import numpy as np, pandas as pd, torch
from datasets import load_dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import (roc_auc_score, average_precision_score,
                             precision_recall_curve)
from transformers import (AutoTokenizer, AutoModelForCausalLM,
                          AutoModelForSequenceClassification)

SEED, N = 42, 10
MODEL, NLI = '<model-checkpoint>', 'microsoft/deberta-large-mnli'
PROMPT = 'Answer the question in one short sentence. Question: {q}'

df = load_dataset('truthful_qa', 'generation')['validation'].to_pandas()
df = df[['question', 'category', 'correct_answers', 'incorrect_answers']]
dev, test = train_test_split(df, test_size=0.80,
                             stratify=df['category'], random_state=SEED)

tok = AutoTokenizer.from_pretrained(MODEL)
lm = AutoModelForCausalLM.from_pretrained(MODEL, torch_dtype=torch.bfloat16,
                                          device_map='auto')
DEV = 'cuda' if torch.cuda.is_available() else 'cpu'
nt = AutoTokenizer.from_pretrained(NLI)
nli = AutoModelForSequenceClassification.from_pretrained(NLI).to(DEV).eval()
LAB = {v.upper(): k for k, v in nli.config.id2label.items()}


def generate(prompt, n=1, temp=0.0, max_new=64, seed=SEED):
  ids = tok.apply_chat_template(
    [{'role': 'user', 'content': prompt}],
    add_generation_prompt=True, return_tensors='pt',
    return_dict=False).to(lm.device)
  kw = (dict(do_sample=True, temperature=temp, top_p=1.0)
        if temp > 0 else dict(do_sample=False))
```

```
    torch.manual_seed(seed)
    o = lm.generate(ids, max_new_tokens=max_new, num_return_sequences=n,
                    output_scores=True, return_dict_in_generate=True, **kw)
    lp = lm.compute_transition_scores(o.sequences, o.scores,
                                      normalize_logits=True)
    texts = [t.strip() for t in tok.batch_decode(
        o.sequences[:, ids.shape[1]:], skip_special_tokens=True)]
    return texts, lp.float().cpu().numpy()


@torch.no_grad()
def probs(a, b):
    x = nt(a, b, return_tensors='pt', truncation=True).to(DEV)
    return torch.softmax(nli(**x).logits, -1)[0].cpu().numpy()


# ---- Labels, detector scores and evaluation ----
def label(q, ans, good, bad):  # 1 = correct; audit 100 labels by hand
    e = lambda r: probs(f'{q} {ans}', f'{q} {r}')[LAB['ENTAILMENT']]
    return int(max(map(e, good)) > max(map(e, bad)))


def consistency(q, g, S):
    return float(np.mean([probs(f'{q} {s}', f'{q} {g}')[LAB['CONTRADICTION']]
                          for s in S]))


def sem_entropy(q, S):
    same = lambda a, b: min(
        probs(f'{q} {a}', f'{q} {b}')[LAB['ENTAILMENT']],
        probs(f'{q} {b}', f'{q} {a}')[LAB['ENTAILMENT']]) > 0.5
    cl = []
    for a in S:
        for c in cl:
            if same(c[0], a):
                c.append(a)
                break
        else:
            cl.append([a])
    p = np.array([len(c) for c in cl]) / len(S)
    return float(-(p * np.log(p)).sum())


def score(split, seed=SEED):  # repeat with seed in range(42, 47)
    rows = []
    for r in split.itertuples():
        g, lp = generate(PROMPT.format(q=r.question), seed=seed)
        S, _ = generate(PROMPT.format(q=r.question), n=N, temp=1.0, seed=seed)
        rows.append(dict(
            category=r.category,
            correct=label(r.question, g[0], r.correct_answers,
                          r.incorrect_answers),
            pred_entropy=float(-lp[0].mean()),
            consistency=consistency(r.question, g[0], S),
            sem_entropy=sem_entropy(r.question, S)))
    return pd.DataFrame(rows)


dev_s, test_s = score(dev), score(test)


def evaluate(det):
```

```
    yd, yt = 1 - dev_s['correct'], 1 - test_s['correct']  # 1 = hallucinated
    p, r, thr = precision_recall_curve(yd, dev_s[det])
    f1 = 2 * p[:-1] * r[:-1] / np.clip(p[:-1] + r[:-1], 1e-9, None)
    t = thr[f1.argmax()]  # threshold chosen on dev only
    flag = (test_s[det] >= t).astype(int)
    tp = int(((flag == 1) & (yt == 1)).sum())
    fp = int(((flag == 1) & (yt == 0)).sum())
    fn = int(((flag == 0) & (yt == 1)).sum())
    return dict(
        detector=det,
        auroc=roc_auc_score(yt, test_s[det]),
        auprc=average_precision_score(yt, test_s[det]),
        precision=tp / max(tp + fp, 1),
        recall=tp / max(tp + fn, 1),
        missed=fn,
        false_alarms=fp,
        abstention=flag.mean(),
        answered_acc=test_s[flag == 0]['correct'].mean(),
        halluc_rate=yt[flag == 0].sum() / len(test_s))


print(pd.DataFrame([evaluate(d) for d in
                    ['pred_entropy', 'consistency', 'sem_entropy']]).round(4))

# ---- Reduction (retrieve() is a stub for the fixed index) ----
# RAG: prompt = f'Context: {retrieve(q, 3)} Question: {q}'
# CoVe: draft -> 3 check questions -> independent answers -> revised answer,
#       all through generate().
```

9. RESULTS

This version reports no new measurements. Tables 7 and 8 are templates for the experiment to fill, with the same layout for every model (A, B, C), and Table 9 summarises published findings. Empty cells must be measured, not copied from the literature, because setups differ.

Table -7: Detection results

Model	Detector	AUROC (95% CI)	AUPRC	Prec.	Recall	F1	Missed (FN)	False alarms (FP)
Model A	Predictive entropy	0.533 (0.404–0.669)	0.389	0.311	0.742	0.438	8.0	51.0
Model A	Sampling consistency	0.594 (0.480–0.703)	0.425	0.352	0.865	0.500	4.2	49.6
Model A	Semantic entropy	0.432 (0.340–0.529)	0.293	0.300	0.723	0.421	8.6	51.6

Precision, recall and F1 refer to the hallucination class at the development-fitted threshold.

Table -8: Reduction results

Model	Strategy	Hallucination rate	Abstention rate	Answered accuracy	Generations per question
Model A	Baseline	32.3%	0.0%	67.7%	1
Model A	Retrieval-augmented generation	33.3%	0.0%	66.7%	1.0
Model A	Chain-of-verification	28.1%	0.0%	71.9%	3.0
Model A	Abstention (best detector)	4.4% (SD 1.8)	79.6% (SD 8.0)	77.6%	11

A difference is called material only if it exceeds the seed standard deviation and the bootstrap intervals do not overlap.

Table -9: Evidence from published work (different models, datasets and labels; not comparable with this study)

Method / benchmark	Reported finding (paraphrased)	Source
TruthfulQA	Best model truthful on 58% of questions against 94% for humans; larger models tended to be less truthful	[6]
Sampling consistency	Agreement between samples flags non-factual sentences without external knowledge	[9]
Semantic entropy	Detects arbitrary wrong answers better than naive entropy; does not address consistent errors	[10], [11]
Self-knowledge	Larger models are reasonably calibrated on multiple-choice questions, weaker on unfamiliar tasks	[12]
Retrieval	Improved knowledge-intensive tasks and produced more factual text	[3]
Chain-of-verification	Reduced hallucination on list questions, closed-book QA and long-form generation	[4]

10. DISCUSSION

Each row states what prior work leads us to expect and what would contradict it.

Table -10: Outcomes by research question

RQ	Expectation from prior work	Contradicting outcome
RQ1	Substantial baseline rate on TruthfulQA, built to elicit falsehoods [6]	Near-zero rates, suggesting label leakage
RQ2	Semantic entropy at least as good as predictive entropy [10], [11]	Predictive entropy matching it with non-overlapping intervals
RQ3	Retrieval and verification lower the rate [3], [4]; abstention does so by answering less	A reduction that vanishes once abstention is counted
RQ4	Error balance depends on the threshold	Identical balance across detectors
RQ5	Weaker detection where the model is consistently wrong [9], [11]	Uniform performance across categories
RQ6	Not predictable; few studies cross strategy and model	Either factor explaining almost nothing
RQ7	Small gaps may fall within noise	Large stable gaps across seeds and models
RQ8	A full record allows reruns	Reruns outside the reported intervals

Three cautions apply. AUROC and AUPRC are threshold-free, while precision, recall, abstention and answered accuracy depend on the development threshold, so ranking and operating-point conclusions are separate. A mitigation can appear to work by refusing more, so the rate is never reported without the abstention rate. Labels come from an NLI procedure, so every comparison inherits its errors; the human audit bounds but does not remove this.

11. ERROR ANALYSIS

The analysis is run on the best detector per model, grouped by TruthfulQA category and answer length. Missed share is the proportion of hallucinated answers the detector failed to flag.

Table -11: Error analysis by category

Category	Questions	Hallucinated	Rate	Missed (FN)	False alarms (FP)	Missed share
Myths	16	9	56.2%	3	7	33.3%
Science	16	7	43.8%	0	8	0.0%
Health	16	4	25.0%	0	11	0.0%
Law	16	4	25.0%	0	11	0.0%
Misc	16	4	25.0%	1	9	25.0%
Fiction	16	3	18.8%	0	12	0.0%

Four error types will be coded on 100 missed hallucinations per model: consistent falsehood (invisible to agreement-based scores [11]), retrieval-induced error, labelling error, and multi-claim answers that a single score cannot capture. Causes such as training data or decoding cannot be separated by this design and are not inferred.

12. THREATS TO VALIDITY AND LIMITATIONS

- No experiment has been run, so no empirical claim is made, and the code may contain errors.
- NLI labels can mislabel valid paraphrases; a 100-label audit only estimates this.
- TruthfulQA has 817 questions, so intervals will be wide; some references may be contested.
- HaluEval samples are LLM-generated and may carry stylistic cues [7].
- 7B to 8B models may not represent larger or proprietary models; prompts and chat templates matter.
- Sampling detectors assume errors appear as disagreement and miss consistent falsehoods [11].
- A threshold fitted on about 163 development questions is noisy.
- Retrieval results depend on the corpus and retriever; cost is measured on one hardware setup.
- No monetary or safety cost matrix for missed hallucinations versus unnecessary refusals is used.

13. FUTURE WORK

- Run the protocol, fill the templates and report results with intervals.
- Add white-box methods such as inference-time intervention [17] and layer-contrastive decoding [18].
- Replace NLI labelling with a larger human-annotated test set, and score long-form text at claim level with FActScore [8].
- Choose abstention thresholds with an explicit cost matrix, and add calibration such as temperature scaling [13].
- Repeat with larger models and domain benchmarks, for example medical or legal questions.

14. CONCLUSION

This paper specified a reproducible experiment for hallucination detection and reduction, crossing three detectors, three mitigations and several models on public benchmarks with one labelling procedure, one decoding setting and development-only thresholds. It reports abstention beside the hallucination rate so that a lower rate cannot be obtained silently by answering less. Published work indicates that sampling and entropy-based detectors carry real signal and that grounding and self-verification can lower hallucination, but not how these compare under a shared protocol. No measurements are reported here, so no method is ranked; the next step is to run the code, audit the labels and fill the templates in Section 9.

15. DATA AND REPRODUCIBILITY STATEMENT

The experiment uses the public benchmarks in Section 7. Seeds, split ratio, prompts, decoding, detectors, mitigations and metrics are documented here, with code in Section 8.5. Before submission the authors should archive the final code, environment specification, model and NLI checkpoint revisions, retrieval corpus snapshot, split question identifiers, all generated answers and the human-audit labels.

REFERENCES

[1] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, vol. 55, no. 12, art. 248, 2023.

[2] L. Huang et al., "A Survey on Hallucination in Large Language Models," arXiv:2311.05232, 2023.

[3] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 9459–9474.

[4] S. Dhuliawala et al., "Chain-of-Verification Reduces Hallucination in Large Language Models," in Findings of ACL, 2024 (arXiv:2309.11495).

[5] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On Faithfulness and Factuality in Abstractive Summarization," in Proc. ACL, 2020, pp. 1906–1919.

[6] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," in Proc. ACL, 2022, pp. 3214–3252.

[7] J. Li, X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models," in Proc. EMNLP, 2023, pp. 6449–6464.

[8] S. Min et al., "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation," in Proc. EMNLP, 2023, pp. 12076–12100.

[9] P. Manakul, A. Liusie, and M. J. F. Gales, "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models," in Proc. EMNLP, 2023, pp. 9004–9017.

[10] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation," in Proc. ICLR, 2023.

[11] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," Nature, vol. 630, pp. 625–630, 2024.

[12] S. Kadavath et al., "Language Models (Mostly) Know What They Know," arXiv:2207.05221, 2022.

[13] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in Proc. ICML, 2017, pp. 1321–1330.

[14] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," PLOS ONE, vol. 10, no. 3, e0118432, 2015.

[15] X. Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models," in Proc. ICLR, 2023.

[16] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[17] K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg, "Inference-Time Intervention: Eliciting Truthful Answers from a Language Model," in Advances in Neural Information Processing Systems, vol. 36, 2023.

[18] Y.-S. Chuang et al., "DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models," in Proc. ICLR, 2024.