Consistency Is the Key: Detecting Hallucinations in LLM Generated Text By Checking Inconsistencies About Key Facts

Anonymous ACL submission

Abstract

Large language models (LLMs), despite their remarkable text generation capabilities, often hallucinate and generate text that is factually incorrect and not grounded in real-world knowledge. This poses serious risks in domains like healthcare, finance, and customer support.A typical way to use LLMs is via the APIs provided by LLM vendors where there is no access to model weights or options to fine-tune the model.Existing methods to detect hallucinations in such settings where the model access is restricted or constrained by resources typically require making multiple LLM API calls, increasing latency and API cost. We introduce ConFactCheck, an efficient hallucination detection approach that does not leverage any external knowledge base and works on the simple intuition that responses to factual probes within the generated text should be consistent within a single LLM and across different LLMs. Rigorous empirical evaluation on multiple datasets that cover both the generation of factual texts and the open generation shows that ConFactCheck can detect hallucinated facts efficiently using fewer resources and achieves significantly higher accuracy scores compared to existing baselines that operate under similar conditions. Our code is available here.

1 Introduction

Large Language Models (LLMs) are the go-to tools for NLP applications given their excellent text generation capabilities (Zhao et al., 2023). However, despite recent developments in model architecture and training, even state-of-the-art models such as GPT-4 (Achiam et al., 2023) and PALM-540B (Chowdhery et al., 2023) often generate text that appears plausible, but is factually incorrect or non-sensical – a phenomenon termed *hallucination* (Huang et al., 2023). A formal analysis by Xu et al. (2024) shows that LLMs cannot learn all possible computational functions, and hence, by design, will always hallucinate, albeit to different degrees. Consequently, detecting when the LLM hallucinates is imperative to take corrective action and minimize misinformation from reaching users.

Such model hallucinations can be either *intrinsic* or *extrinsic* (Ji et al., 2023). Intrinsic hallucinations arise when model outputs contradict the input or in-context instructions and can often be detected by checking input-output consistency(Huang et al., 2023). Extrinsic hallucinations, on the other hand, occur when the model output is factually incorrect and is not grounded on the pre-training data (Huang et al., 2023). Given the volume of pre-training data and that it is typically inaccessible by the users, extrinsic hallucinations pose a greater challenge due to their unverifiable nature (Ji et al., 2023).

Hallucinations in LLMs are typically addressed by either *(i)* improving factual accuracy via training or fine-tuning (Tian et al., 2023; Azaria and Mitchell, 2023a; Chuang et al., 2023), or *(ii)* verifying model outputs using external knowledge sources (Cheng et al., 2024). However, in many practical cases, end-users or developers lack access to model weights or external verification sources. Recent approaches circumvent this by repeatedly querying the LLM (Manakul et al., 2023; Zhang et al., 2023a; Liu et al., 2022)to thoroughly verify responses or sample large number of outputs to estimate output probability distributions, leading to significantly increased cost and latency. To address these limitations, we propose ConFactCheck, a lightweight method for hallucination detection that relies solely on the LLM's internal knowledge. ConFactCheck is based on a simple idea: an LLM's understanding of a topic can be evaluated by asking related questions and measuring consistency. This recursive probing strategy has also been used in testing question-answering systems (Chen et al., 2021). As illustrated in Figure 1, ConFactCheck identifies key entities/tags (using NER/POS tagging) in the generated output

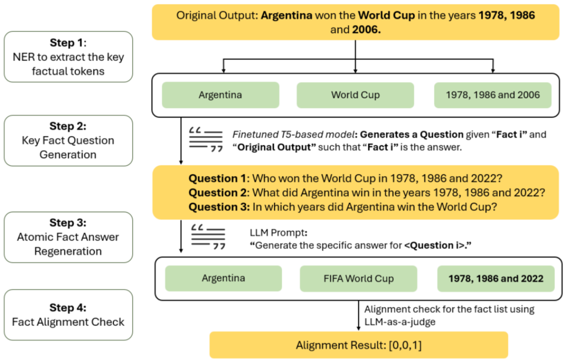

Figure 1: Key fact-based hallucination detection through the Fact Alignment check of our ConFactCheck pipeline. Each fact is used to generate a question, and the fact is regenerated by prompting the question to the LLM. The regenerated facts are compared with the original extracted key facts to check for their consistency.

and then formulates contextually relevant questions around these entities. We term these entities/tags as 'key facts', as these contain essential factual information in sentences. The LLM's answers to these questions are checked for consistency with the original response, with high consistency indicating that the output is grounded in the model's pre-training data (reflective of the world knowledge).

We evaluate ConFactCheck on four different datasets spanning question-answering (NQ_Open (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), WebQA (Berant et al., 2013)) and open-ended generation tasks where inputs to the LLM lack any additional context (WikiBio (Manakul et al., 2023)). ConFactCheck outperforms recent state-of-the-art self-check or self-consistency-based baselines (Manakul et al., 2023; Zhang et al., 2023a; Liu et al., 2022) along with baselines relying on the internal states of models (Chen et al., 2024) for LLMs of different model families. ConFactCheck achieves this outperformance while being significantly faster and requiring a lower number of LLM calls (*c.f.*, Table 2). We also report the results of various ablation studies guiding our design choices and conclude by discussing the strengths and limitations of ConFactCheck.

2 Related Work

LLMs are inherently prone to hallucinations (Xu et al., 2024; Ji et al., 2023), a phenomenon also observed in visual and multi-modal models (Bai et al., 2024; Liu et al., 2024). This has led to extensive research on hallucination detection and mitigation (Huang et al., 2023; Zhang et al., 2023b; Tonmoy et al., 2024). Existing methods fall broadly into two categories: *self-checking, prompt-based* approaches and those that require access to model weights or external knowledge sources.

Methods Requiring Access to Model Weights and External Sources: Tian et al. (2023) demonstrate that fine-tuning with factuality preferences improves output correctness. Azaria and Mitchell (2023b) use internal LLM activations passed through a classifier to estimate truthfulness. INSIDE (Chen et al., 2024) uses internal sentence embeddings and analyzes their covariance eigenvalues to detect hallucinations. Various decoding strategies (Chuang et al., 2023; Shi et al., 2024) have also been developed that utilize token probabilites at various layers to detect and mitigate hallucinations. Some approaches such as HaluAgent (Cheng et al., 2024) use additional tools such web search engines, code interpreters etc for text, code-based detection of hallucinations.

Self-Checking and Prompt-Based Methods: Zhang et al. (2023a) propose Semantic-Aware Cross-Check Consistency (SAC^3), a sampling-based method that checks for self-consistency across multiple generations. Similarly, SelfCheckGPT (Manakul et al., 2023) samples diverse outputs and scores their similarity to the original to estimate confidence. InterrogateLLM (Yehuda et al., 2024), focuses on regenerating the original query for a generated answer by reversing few-shot QA pairs to few-shot AQ pairs to self-check for model confidence during regeneration. These self-refining approaches often rely on the target LMs themselves, which is also demonstrated in Self-Refine (Madaan et al., 2023), an iterative mitigation-based approach for hallucinations. Mündler et al. (2023) explore self-contradictions using two LLMs – one for generation and one for contradiction analysis. TRUE (Honovich et al., 2022),evaluates factual consistency using a range of metrics (n-gram, NLI, model-based) on the FEVER dataset (Thorne et al., 2018). Liu et al. (2022) propose a reference-free, token-level method for detecting hallucinations and also present the Hallucination Detection dataset (HaDes), with raw web text being perturbed and then annotated by humans to design it for hallucination detection as a classification task. FactScore (Min et al., 2023) breaks outputs into atomic facts, and verifies them using reliable *external* knowledge sources. We also utilize the notion of atomic facts in ConFactCheck , however, instead of leveraging external sources, we check for consistency in LLM outputs about the atomic facts.

3 The ConFactCheck Approach

Figure 2 summarizes our proposed hallucination detection approach comprising of two main steps – *(i)* a *fact alignment* check where key facts in the output are compared with facts obtained by targeted probing of the LLM; and *(ii)* a *uniform distribution check* that filters out the low confidence predictions. We now describe the overall pipeline in detail.

3.1 Fact Alignment Check

Extracting Key Facts: To check whether a piece of text, $\mathcal{A}$, generated by an LLM $\mathcal{M}$ is hallucinated, we start with the assumption that the generated text is correct. We then generate questions targeting each key fact in $\mathcal{A}$, such that they can be answered solely using the content of $\mathcal{A}$. Subsequently, we employ the LLM to answer the questions and see if the answers match the information in $\mathcal{A}$, a mismatch indicating hallucinations. The initial step is to identify the factual components within a sentence. According to Kai et al. (2024), factual information in a sentence is typically conveyed through specific parts of speech, *viz.*, nouns, pronouns, cardinal numbers, and adjectives. We highlight tags with such information as key facts that are to be extracted. Min et al. (2023) use a similar concept, where they classify short sentences in text (obtained by InstructGPT generation and human annotation) as atomic facts. However, the key facts we discuss are extracted NER/POS tags containing factual information, and hence are different. Key facts can be extracted by performing part-of-speech (POS) tagging or Named Entity Recognition (NER) on the sentence. Given an LLM output $\mathcal{A}$, we perform coreferencing and decompose $\mathcal{A}$ into sentences $S_1, S_2, \ldots, S_N$, where N is the total number of sentences, such that $\mathcal{A} = \{S_1, S_2 \ldots, S_N\}$. Each sentence is tagged to extract key facts a_{ij}, where $i \in \{1, \ldots, N\}$, and j depends on the number of tagged entities in a sentence. The tagging can be either POS-based or NER-based, as discussed in Section 5.4.3. For example, given the original sentence "*Argentina won the World Cup in the years, 1978, 1986 and 2006.*", in Figure 1, the key facts consist of $a = [a_{11} = \textit{Argentina}, a_{12} = \textit{World Cup}, a_{21} = \textit{1978, 1986 and 2006}]$.

Targeted Question Generation: After identifying key facts, the next step involves verifying whether each fact is hallucinated within the context of the sentence. Unlike previous methodologies that assign a hallucination score to each sentence, ConFactCheck focuses on key facts, thereby enhancing explainability by pinpointing the exact parts of a sentence that are hallucinated and providing reasons for this determination, as detailed in Section 5.5. Specifically, for each key fact a_{ij} given sentence S_i, a corresponding question q_{ij} is generated (using a T5-based model that is specifically finetuned for this task of question regeneration), with a_{ij} as the target answer and S_i as the context, expressed as $q_{ij} = \mathcal{Q}(a_{ij}|S_i)$, where $\mathcal{Q}$ represents the question generation module. In Figure 1, each key fact provides one question $q = [q_{11} = \text{Question 1}, q_{12} = \text{Question 2}, q_{13} = \text{Question 3}]$. LLM $\mathcal{M}'$ is then used to evaluate these questions at a low temperature to ensure response consistency, as it enables the LLM to generate high-quality and deterministic outputs. Each individual key fact-based question is answered by the LLM with greater precision and therefore helps to better identify whether the fact is correct or incorrect (Dhuliawala et al., 2024). Note that $\mathcal{M}'$ may or may not be the same as $\mathcal{M}$, as another LLM can be used to evaluate the responses of LLM $\mathcal{M}$.

Consistency Checking The responses from $\mathcal{M}'$ yield regenerated facts f_{ij}, which are subsequently checked for consistency with a_{ij}. To check for the similarity between f_{ij} and a_{ij}, we follow the LLM-as-a-judge paradigm (Zheng et al., 2023), by querying GPT4.1-mini using few-shot prompting to assess whether each pair is aligned or not. For instance, the set f for Figure 1 being $f = [f_{11} = \textit{Argentina}, f_{12} = \textit{FIFA World Cup}, f_{21} = \textit{1978, 1986 and 2022.}]$, and original key facts being $a = [a_{11} = \textit{Argentina}, a_{12} = \textit{World Cup}, a_{21} = \textit{1978, 1986 and 2006}]$. In this case, facts f_{21} and a_{21} are non-aligned; whereas, the pairs $< f_{11}, a_{11} >$ and $< f_{12}, a_{12} >$ are aligned as per the judge's output. For each aligned and non-aligned pairs, we assign the score of 0 and 1 respectively. Note that since the number of extracted *facts* varies based on the sentence, the number of questions generated per sentence also varies. The consistency checking step, thus, enables the decomposition of sentence-level information into discrete factual elements and leverages and operates under the assumption that the LLM's responses will remain consistent for factual information when sampled at a low temperature.

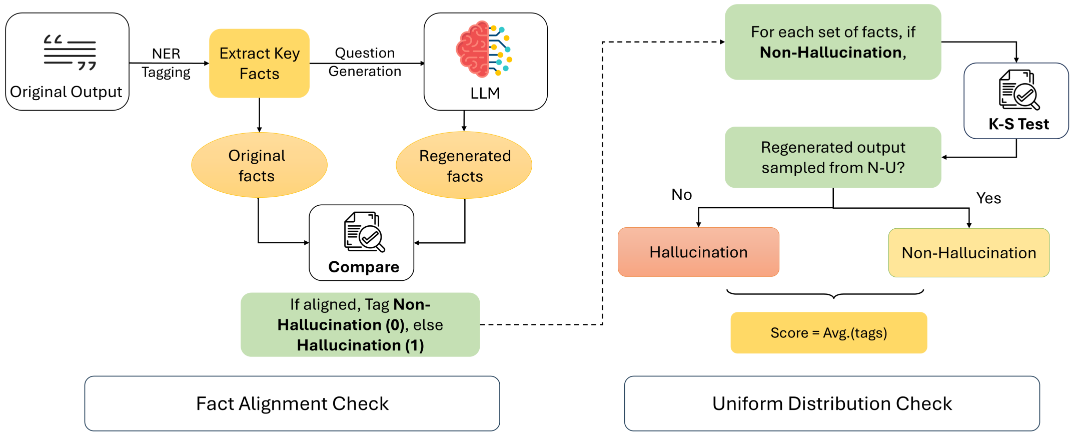

Figure 2: Pipeline of the CONFACTCHECK approach, with NER tagging of outputs followed by the first comparison-based check (Fact Alignment Check) and the secondary KS test-based probability check (Uniform Distribution Check) for rechecking the classfied non-hallucinations, result in the final tagging of hallucinations.

3.2 Uniform Distribution Check

After the fact-alignment step, we perform a subsequent step to check if the facts were regenerated with high confidence. The underlying intuition behind this step is that if the LLM is confident in regenerating a fact correctly, the probability distribution of the generated tokens will be skewed, with the selected tokens having significantly higher probabilities than the other possible tokens. This results in a non-uniform distribution of token probabilities. Conversely, if the LLM is uncertain, even though the generated tokens may have the highest relative probability, their values will be closer to those of alternative tokens (closer to a uniform distribution) and indicating less confidence in LLM prediction. To quantify this effect, we apply the Kolmogorov–Smirnov (K–S) test to the top five tokens associated with each regenerated fact f_{ij}. The test is conducted using a standard significance level of 0.05. A p-value below this threshold leads to the rejection of the null hypothesis (i.e., the top tokens are drawn from a uniform distribution) implying that the LLM exhibits confidence in its generation.If the test indicates a non-uniform distribution, the LLM is deemed confident in regeneration, and original fact a_{ij} is classified as non-hallucinated. However, if the token probabilities follow a uniform distribution, it is concluded that the particular fact is hallucinated, reflecting the LLM's lack of confidence. The final hallucination score for a sentence S_i is calculated by averaging the individual scores of a_{ij} present in it to give a probability of how likely a sentence has been hallucinated.

4 Experimental Protocol

4.1 Task and Datasets

We consider two common task settings – question answering (QA) and text summarization. In the QA setting, LLMs are particularly susceptible to factual hallucinations, especially when no external context or information is provided with the input questions. The summarization task is a representative of the long-form text generation tasks where the output is not limited to be a short answer (a phrase or a sentence), and hence enables us to evaluate the ability of various methods to detect hallucinations in longer pieces of text. Further, this setting also tests the ability of the LLM to generate text that is *faithful* to the input context (text to be summarized).

We use the following datasets for evaluation, encompassing both QA and summarization settings:
1. Natural Questions (NQ)-open (Kwiatkowski et al., 2019) is an open-domain QA benchmark derived from the Natural Questions dataset (Lee et al., 2019).We use these questions as input for the LLM to generate answers, which are then checked for hallucination by various methods.
2. HotpotQA (Yang et al., 2018) is a QA dataset that features complex questions requiring multi-hop reasoning.
3. WebQA (Berant et al., 2013) dataset is a factoid QA dataset where the questions are derived from the Freebase knowledge base.
4. WikiBio (Manakul et al., 2023) is a hallucination detection dataset derived from Wikipedia biographies. It consists of 238 randomly selected articles from among the longest 20% Wikipedia

articles. It also provides synthetic text generated by GPT-3 for each of the original articles, along with labels for factual correctness of the sentences.

4.2 Baselines

We use following four representative self-check and self-consistency based hallucination detection methods as baselines.

HaDes (Liu et al., 2022) is an external reference-free method that leverages various token-level features such as POS tags, average word probability, mutual information, and TF-IDF scores to identify if a token is hallucinated or not.

SelfCheckGPT (Manakul et al., 2023) is a sampling based approach built upon the intuition that for hallucinated responses, stochastically sampled responses for the same input are likely to diverge.

SAC^3 (Zhang et al., 2023a), another sampling-based approach that generates responses to multiple semantically similar inputs to the original input and checks for consistency in the generated outputs.

INSIDE (Chen et al., 2024) detects hallucinations using the EigenScore metric, calculated using the eigenvalues of the covariance matrix of the responses to measure the semantic consistency/diversity in the dense embedding space of the generated outputs.

4.3 Implementation details

Models Used. We use `LLaMA3.1-8B-Instruct` and `Qwen2.5-7B-Instruct` as the base LLMs for comparing ConFactCheck and various baselines. Further, we use different models of Phi-3 family to study how well ConFactCheck performs with LLMs of varying scale (Section 5.3). We present ablations that guided our design choices in Sections 5.4.2 and 5.4.3. We use the official implementation of HaDes[1] for our experiments. For SAC^3 (Zhang et al., 2023a), we compute the question-level consistency SAC^3-Q score and employ predetermined thresholds to discern the presence of hallucinated outputs.

Metrics for Analysis: We consider hallucination detection as a binary classification task where the text generated by the LLM is either hallucinated or not. For QA datasets, we assign labels of 1 for hallucination and 0 for non-hallucination to the original outputs by comparing them with the golden answers in the QA datasets using GPT4.1-mini as a judge LLM. For WikiBio, each sentence-level golden label is provided in the dataset itself. We compare the baselines with our approach (see Table 1) and report the AUC-PR scores on the 3 open-domain QA datasets, as well as the WikiBio summarization dataset. Note that the SelfCheck-GPT baseline is applicable on the WikiBio dataset, as the others deal with only the QA task and require questions as part of their input.

[1]`https://github.com/microsoft/HaDes`

5 Empirical Results

5.1 ConFactCheck for Hallucination Detection

Table 1 summarizes the results of different methods for the four datasets and across two LLM backbones (`LLaMA3.1-8b` and `Qwen2.5-7B`). We observe that ConFactCheck outperforms all four baselines for all the datasets and the two LM backbones. The second-best performing method in each column (LLM backbone and dataset combination) is underlined. We note that no baseline model achieves consistently high performance across all the settings. While INSIDE achieves the second-best performance on NQ-Open (with `LLaMA3.1`) and WebQA (with `Qwen2.5`), SelfCheckGPT achieves the second-best performance on three other QA settings. Further, only SelfCheckGPT can be used for detecting hallucinations in free-form text (WikiBio dataset), as the other baselines are designed for detecting hallucinations in QA tasks and need questions as part of their input. ConFactCheck, on the other hand, can detect hallucinations in QA as well as free-form text settings and achieves strong outperformance across all settings, with decent relative percentage gains in four of the eight settings (from 7% to 20%). Such strong performance of ConFactCheck can be attributed to the fact that it identifies the key factual tokens in the generated text and probes the LLM regarding its knowledge around these tokens.

5.2 Computational Efficiency of Different Methods

Recall from discussions in Section 1 that self-check or self-refinement style methods suffer from high latencies due to the need to query the LLM repeatedly to estimate the output probability distributions or for a thorough verification of the generated output. ConFactCheck, on the other hand, identifies key facts in the generated output and generates targeted questions around these facts, thereby greatly reducing the number of LLM calls. Fur-

Model	NQ Open		HotpotQA		WebQA		WikiBio	
	LLaMA3.1	Qwen2.5	LLaMA3.1	Qwen2.5	LLaMA3.1	Qwen2.5	LLaMA3.1	Qwen2.5
HaDes (Liu et al., 2022)	0.54	0.67	0.68	0.69	0.46	0.48	N/A	N/A
SAC^3 (Zhang et al., 2023a)	0.59	0.71	0.68	0.59	<u>0.63</u>	0.55	N/A	N/A
SelfCheckGPT (Manakul et al., 2023)	0.56	<u>0.75</u>	<u>0.76</u>	<u>0.77</u>	0.51	0.63	0.82	0.83
INSIDE (Chen et al., 2024)	<u>0.61</u>	0.54	0.56	0.60	0.58	<u>0.68</u>	N/A	N/A
ConFactCheck	**0.73**	**0.80**	**0.83**	**0.84**	**0.66**	**0.71**	**0.86**	**0.85**
% gain over best baseline	+20%	+7%	+9%	+9%	+5%	+4%	+5%	+2%

Table 1: AUC-PR scores for NQ Open, HotpotQA, WebQA, and WikiBio datasets. We compare ConFactCheck in the same settings as the baselines, using LLaMA3.1-8B-Inst and Qwen2.5-7B-Inst as the base models. Settings for ConFactCheck results use beam decoding on the whole pipeline (this yields best possible scores). The best performing method in a given column is in **bold** and the second best performing model is <u>underlined</u>.

ther, ConFactCheck relies on lightweight comparisons and statistical operations (Section 3) to check if the answers to targeted questions align with the original output. Table 2 presents the average number of LLM calls made and the average inference time for different methods. We note from the table that ConFactCheck achieves fast inference times for both the `LLaMA3.1` and `Qwen2.5` backbones. INSIDE is slightly faster than ConFactCheck, however our pipeline offers up to≈3.5x speedup compared to SelfCheckGPT (Manakul et al., 2023) (9.51s vs. 33.69s for LLaMA3.1) and ≈3x when compared to SAC^3 (on Qwen2.5 model). Note also that in the case of ConFactCheck the number of calls being made to the LLM is equivalent to the average number of key facts extracted per input in the dataset plus one additional call to the judge-LLM for Fact Alignment. On the other hand, SelfCheckGPT and SAC^3 need to repeatedly query the LLM to compute their respective scores and the accuracies increase with increasing number of queries to the LLM. In Table 2), we report the latency numbers for SelfCheckGPT and SAC^3 with 5 LLM calls per question, and INSIDE with 10 LLM calls per question as recommended by the respective papers. Also note that the performance numbers for SelfCheckGPT and SAC^3 in Table 1 are with these higher number of LLM calls (5 each, while they can be lower) to exhibit their best performance with efficiency. All experiments on ConFactCheck and the baselines as reported were run using NVIDIA A6000 GPUs, using the mentioned open-source LLMs for querying and execution.

5.3 ConFactCheckwith LLMs of Varying Scale

We now study how the performance of ConFactCheck varies with the scale of the underlying LLM. We use the `Phi-3-Instruct` family (Abdin et al., 2024) of models for this purpose and chose models of 3 sizes – 3.8B, 7B, and 13B. Table 3 summarizes the results for the three Phi-3 models on the three QA datasets. In addition to the AUC-PR of hallucination detection, we also report the percentage of hallucinated outputs in each setting to understand the severity of hallucinations at different model scales. We note from the table that for these datasets, there is a decent amount of hallucinated outputs, which wavers from the 3.8B to 13B models. This shows that just increasing the model size may not eliminate hallucinations. We also note that the ability of ConFactCheck to detect hallucinations is similar and consistent across different model sizes. While the Phi3-7B slightly outperforms on NQ-open, the increasing model sizes show moderate gains for the HotpotQA and WebQA datasets.

Method	# LLM calls	LLaMA3.1	Qwen2.5
SelfCheckGPT	5	33.69 s	24.81 s
SAC^3	5	15.46 s	29.37 s
INSIDE	10	4.89 s	5.68 s
ConFactCheck	3.8	9.51 s	9.03 s

Table 2: Average inference time (in seconds) for ConFactCheck and the baselines (which have configurable amount of LLM calls) over the samples of the NQ_Open dataset while using `LLaMA3.1` and `Qwen2.5` models. ConFactCheck offers significant speedups over the self-check baselines.

5.4 Ablation Studies

We now describe different ablation studies that guided different design choices for ConFactCheck. We report the impact of *fact-alignment* and *uniform distribution check*

Model	NQ Open		HotpotQA		WebQA	
	AUC	%Hall.	AUC	%Hall.	AUC	%Hall.
Phi-3-4b	0.69	0.65	0.74	0.69	0.63	0.49
Phi-3-7b	**0.73**	0.58	0.74	0.60	0.62	0.46
Phi-3-13b	0.71	0.54	**0.76**	0.64	**0.65**	0.50

Table 3: Performance of CONFACTCHECK for different size models of the Phi-3 family. We report AUC-PR of hallucination detection and percentage of hallucinated outputs (Hall.) for the 3.8B, 7b, and 13B models for the three QA datasets.

steps in the pipeline (Section 3). We also describe the effects of different decoding strategies and methods for detecting key facts in the input.

5.4.1 Role of Different Components in CONFACTCHECK

Recall that there are two main steps in CONFACTCHECK – *fact alignment* and *uniform distribution check*. The fact alignment step attempts to regenerate the key facts in the generated output by querying the LLM with targeted questions. The regenerated facts are then compared with the original output for consistency. The subsequent uniform distribution check acts as another verification layer by relying on the model's confidence in the generation of regenerated key facts. Table 4 summarizes the hallucination detection scores achieved by just the fact-alignment step along with the improvements achieved by performing the subsequent uniform distribution check (the complete pipeline). We note from the table that the uniform distribution step plays a crucial role in the overall performance of CONFACTCHECK with maximum gains of up to 18%.

Component	LLM	NQ Open	HotpotQA	WebQA
Fact Alignment	LLaMA3.1	0.66	0.79	0.56
+ Distribution Check	LLaMA3.1	**0.73**	**0.83**	**0.66**
% gain		11%	5%	18%
Fact Alignment	Qwen2.5	0.79	0.82	0.68
+ Distribution Check	Qwen2.5	**0.8**	**0.84**	**0.71**
% gain		1%	2%	5%

Table 4: AUC-PR scores achieved by the two major components of CONFACTCHECK. A uniform distribution check after the fact alignment step leads to significant performance gains.

5.4.2 Effect of Decoding Strategies

Regardless of how the original response, subject to hallucination assessment, was generated, we examine the variations in regenerated factual responses when decoding strategies are varied. The following decoding strategies were utilized:

- **Greedy Decoding**: Greedy decoding involves selecting the token from the vocabulary V with the highest conditional probability. This suggests prioritizing key facts for which the model has the highest immediate confidence.
- **Beam Decoding**: Beam decoding represents an enhancement over greedy decoding. In Beam decoding, a parameter known as beam_size determines the number of tokens with the highest conditional probabilities considered at each time step t. For our experiments, we considered the beam size to be 5.

Model	NQ Open	HotpotQA	WebQA	WikiBio
LLaMA3.1 (Greedy)	0.70	0.81	0.62	0.86
LLaMA3.1 (Beam)	**0.73**	**0.83**	**0.66**	0.86
Qwen2.5 (Greedy)	0.79	0.82	0.66	0.85
Qwen2.5 (Beam)	**0.80**	**0.84**	**0.71**	0.85

Table 5: The AUC-PR scores of CONFACTCHECK with `LLaMA3.1-8B-Inst` and `Qwen2.5-7b-Inst` models using different decoding strategies for fact regeneration on the QA datasets. Beam decoding (beam size = 5) outperforms Greedy Decoding in most of the settings.

Beam decoding improves the detection of hallucinations during fact regeneration compared to greedy search. This advantage likely arises because beam decoding explores multiple possible answer paths before selecting the most likely one. Beam decoding also implicitly mitigates hallucinations by preferring sequences with higher cumulative confidence, which are more likely to reflect consistent factual patterns across generations. As a result, when regenerating key facts, beam decoding ensures a more informed selection of entities, and the results in Table 5 show its improvements. Chen et al. (2018) further corroborate this by indicating that beam decoding generally outperforms greedy decoding. By maintaining multiple candidate generations, beam decoding reduces the likelihood of factual errors, ensuring the correct regeneration of facts. However, this decoding strategy does involve a trade-off with computational efficiency compared to greedy decoding.

5.4.3 Tagging of key-facts

Identifying of key facts in the generated text is a crucial step in CONFACTCHECK as they are used to probe the LLM in a targeted fashion. Hence, the choice of method used for identifying key facts in the generated text can have significant impact on the overall performance. Kai et al. (2024) suggests that factual information in a sentence can be identified using POS tagging, specifically 'NNP' or 'NNPS'. Building on this, we selected the tags 'NNP', 'NNPS', 'CD', and 'RB' to be considered key facts. As an alternative, we also evaluated using NER tagging and considering identified named entities as key facts. We used Stanford's Stanza (Qi et al., 2020) library for NER and POS tagging. Additionally, we also sampled random tokens from the sentence and used them as key facts, ensuring that the number of sampled tokens equaled the number of NER tags present. Table 6 summarizes the results for the three strategies and reveals that though the results are similar, NER outperforms both POS tagging and random token sampling in more settings to identify which tokens contribute to the factuality of a sentence or paragraph.

Tagging	NQ Open		HotpotQA		WebQA	
	LLaMA3.1	Qwen2.5	LLaMA3.1	Qwen2.5	LLaMA3.1	Qwen2.5
Random	0.72	0.78	0.82	0.83	**0.68**	0.69
POS	0.71	**0.81**	0.82	0.83	0.66	0.7
NER	**0.73**	0.8	**0.83**	**0.84**	0.66	**0.71**

Table 6: The AUC-PR scores while using different tagging strategies on LLaMA3.1-8B-Inst and Qwen2.5-7B-Inst for identifying key facts in the sentence. NER is observed to perform slightly better in more cases over these three QA datasets.

5.5 Key Strengths of CONFACTCHECK

We now discuss the major strengths of CONFACTCHECK which are summarized as follows.

Training-Free Operation: Our generic approach requires only the LLM-generated output for fact-alignment check stage of the pipeline and does not necessitate dataset- or task-specific training. The number of generated questions is determined by the factual content within the generated sentence, avoiding heuristic selection.In the uniform distribution check, when the original output has been generated using an API where the internal states of the model are not available for accessing the probability distribution, CONFACTCHECK can leverage an open-source LLM compatible with the user's hardware in the fact-alignment check to cross-verify facts and compute token probabilities.

Ease of Implementation: CONFACTCHECK does not require access to model weights or underlying training data. Requiring only the model's output and the LLM used for response generation, our method can be deployed on the same device as the response generation process, whether through a web interface, API, or a locally executed model. Even for the use of KS test, we require only the output token probabilities of the top-5 generations, which can be directly stored during LLM generation.

Consistent Sample Scoring: Unlike previous stochastic hallucination detection methods, such as SelfCheckGPT (Manakul et al., 2023), CONFACTCHECK does not rely on multiple LLM outputs as CONFACTCHECK probes factual tokens at 0 temperature. This ensures score consistency across repeated evaluations of the same sample. Furthermore, by avoiding multiple LLM calls for a single query, CONFACTCHECK reduces the computational overhead compared to methods requiring multiple LLM generations.

Interpretability: CONFACTCHECK provides key-fact-level scoring, enabling users to identify specific hallucinated facts. For instance, in the running example of Figure 1, in addition to classifying the output text as hallucinated, CONFACTCHECK explicitly identifies that the *fact* $a_{21} = \{1978, 1986 \text{ and } 2006\}$ is hallucinated (non-aligned). Operating on fine-grained facts rather than entire sentences, our pipeline offers a greater degree of explainability than previous approaches like SAC (Zhang et al., 2023a), clarifying the rationale behind a hallucination classification.

6 Conclusions

In this work, we propose CONFACTCHECK, a novel fact-based hallucination detection pipeline, and compare it to existing approaches. We evaluate our method on four factuality measurement datasets, providing the first comparison between existing factual hallucination detection methods. Our findings reveal that despite being less computationally expensive and not requiring any training, our method performs on par with other approaches while being significantly faster.

7 Limitations

Despite the high performance, ease of use, and efficiency offered by CONFACTCHECK, it is not without limitations. We analyze and present representative examples of failure cases to highlight its shortcomings and possible future areas of improvement.

Effect of incorrect tags on correct outputs: Consider the following example from HotpotQA: *Which of the office buildings used to staff the White House used to be known as the State, War, and Navy Building?* For this question, the answer provided by an LLM is the following. *The office building used to staff the White House that was once known as the State, War, and Navy Building is now known as the **Eisenhower Executive Office Building**. This building was constructed in 1952 and was named after President Dwight D. Eisenhower.* Although Eisenhower Executive Office Building is factually correct, our pipeline categorizes the paragraph as hallucinated. This discrepancy arises because our model identifies the fact '1952' as hallucinated because of the building's actual construction period between 1871 and 1888. This contrasts with the golden output from HotpotQA, which does not flag the answer as hallucinated (when the judge LLM is used on the original output and golden answer to get the golden label). However, due to the presence of other hallucinated facts, our pipeline assigns a hallucinated tag to the paragraph. Summarily, while the model correctly identifies the building as the Eisenhower Executive Office Building, it erroneously states the construction year as 1952 (actual: 1871–1888). As a result, CONFACTCHECK tags this factual mismatch, leading to a hallucination score for the entire paragraph.

Inefficiency in question generation: The generated questions extracted key facts are done by the T5-based finetuned model. While it is efficient in generating pinpointing questions with the extracted fact as answer with original output as context, some ambigious questions such as "Who was the building named after?" can be generated. This ambiguity can result in inaccuracies when regenerating facts. For this, using a much larger LLM can be useful, however it would be computationally expensive and time-inefficient while not providing significant improvements.

Language-based limited usecases: In addition, we also note that the proposed CONFACTCHECK has only been tested for English language and LLMs trained mostly on English data. Although the framework is theoretically language-agnostic, its reliance on NER/POS tools constrains applicability in low-resource languages lacking robust NLP pipelines. Further, the performance of CONFACTCHECK depends crucially on intermediate steps requiring NER and POS tagging, which may not always be available for low-resource languages.

References

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Hassan Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahmoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Xihui (Eric) Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Olatunji Ruwase, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. Technical Report MSR-TR-2024-12, Microsoft.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

Amos Azaria and Tom Mitchell. 2023a. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Amos Azaria and Tom Mitchell. 2023b. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. 2024. Hallucination of multimodal large language models: A survey. *ArXiv preprint*, abs/2404.18930.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: llms' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. 2021. Testing your question answering software via asking recursively. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 104–116.

Yun Chen, Victor O.K. Li, Kyunghyun Cho, and Samuel Bowman. 2018. A stable and effective learning strategy for trainable greedy decoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 380–390, Brussels, Belgium. Association for Computational Linguistics.

Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, Hongzhi Zhang, Fuzheng Zhang, Di Zhang, Kun Gai, and Ji-Rong Wen. 2024. Small agent can also rock! empowering small language models as hallucination detector. *ArXiv preprint*, abs/2406.11277.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *ArXiv preprint*, abs/2309.03883.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3563–3578, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 161–175, Dublin, Ireland. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv preprint*, abs/2311.05232.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12).

Jushi Kai, Tianhang Zhang, Hai Hu, and Zhouhan Lin. 2024. Sh2: Self-highlighted hesitation helps you decode more truthfully. *ArXiv preprint*, abs/2401.05930.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. 2024. A survey on hallucination in large vision-language models. *ArXiv preprint*, abs/2402.00253.

Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. 2022. A token-level reference-free hallucination detection benchmark for free-form text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6723–6737, Dublin, Ireland. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore:

Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *ArXiv preprint*, abs/2305.15852.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2023. Fine-tuning language models for factuality. *ArXiv preprint*, abs/2311.08401.

S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *Preprint*, arXiv:2401.01313.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *ArXiv preprint*, abs/2401.11817.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yakir Yehuda, Itzik Malkiel, Oren Barkan, Jonathan Weill, Royi Ronen, and Noam Koenigstein. 2024. InterrogateLLM: Zero-resource hallucination detection in LLM-generated answers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9333–9347, Bangkok, Thailand. Association for Computational Linguistics.

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley Malin, and Sricharan Kumar. 2023a. SAC^3: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15445–15458, Singapore. Association for Computational Linguistics.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023b. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Models and Implementations

A.1 SelfCheckGPT (Manakul et al., 2023)

One of the first papers to counter zero-resource hallucination detection, we compare SelfCheckGPT MQAG scores present in Table 1. We set the number of questions per sentence to be 5. The scoring method selected was Bayes with Alpha. Both β_1 and β_2 were set to 0.95.

A.2 SAC3 (Zhang et al., 2023a)

As discussed above, for using SAC^3 as one of the baselines, we evaluate it using the instruction finetuned model version of `LLaMA3.1-8B` and `Qwen2.5-7B`. We calculate the question-level consistency score (SAC^3-Q) which is highlighted in the original study as a score describing the cross-check consistency between 2 types of QA pairs, i) the original question and generated answer as a pair and ii) a number of semantically similar generated questions along with their answers as pairs. For feasibility in accordance with our available computational resources, we experimented with

2 generated perturbated QA pairs. This number can be increased or varied to check for different comparisons, but Zhang et al. (2023a) suggest that using between 2 to 5 perturbed questions per data sample yields similar quantitative results.

A.3 HaDes (Liu et al., 2022)

HaDeS is a novel token-free hallucination detection dataset for free-form text generation. For the dataset creation, raw text from web data is perturbed with out-of-box BERT model. Human annotators are then employed to assess whether the perturbed text spans are hallucinations given the original text. The final model is a binary classifier for detecting hallucinated/non-hallucinated text.

A.4 INSIDE

(Chen et al., 2024) INSIDE is a hallucination detection method which deals with the interal states of LLMs during generation to detect for hallucinations in outputs. Their approach utilizes the layer of sentence embedding outputs and exploits the eigenvalues of the covariance matrix of outputs to measure consistency in the dense embedding space. The define a particular score known as EigenScore, which is the logarithmic determinant of the covariance matrix between a certain K number of outputs' sentence embeddings (to check for the consistency in the relationship of those K outputs' embeddings). Using it as a baseline, we implement it with our settings with `LLaMA3.1-8B` and `Qwen2.5-7B` as the LLMs on the 3 QA datasets and calculate the AUC-PR scores.

B Usage of ConFactCheck on datasets

B.1 Open-Domain Question Answering

Three datasets are used for this particular task, as shown above. We use ConFactCheck on the originally generated outputs for each of the questions in the datasets, to check for whether the LLMs generating the original answers have hallucinated or not. ConFactCheck is applied on a sentence-level basis, where the outputs are split into sentences, following which key facts are extracted and ConFactCheck begins the checking mechanism.

B.2 Text-based Summarization

For this particular task, we use the WikiBio dataset which contains summaries of individuals collected from Wikipedia, along with synthetic GPT3 generated summaries of the same. ConFactCheck is applied as a sentence-level detector on the respective sentences of each of the provided synthetic summaries, which have be annotated with their hallucination labels at the said sentence-level as part of the dataset. We obtain sentence level hallucination scores and compare those with the golden annotate labels per sentence, and for passage-level hallucinations, we average over the sentence-level scores to get overall scores for passages.

C F1-Score based Matching

In our primary pipeline, factual alignment is determined using an LLM-as-a-judge approach. Specifically, we query OpenAI's `GPT-4.1-mini` via the API to compare extracted and regenerated facts and assign binary alignment labels. While this method yields strong performance, it requires reliable access to the OpenAI API and incurs associated computational and cost overheads.

To support use cases where API access is restricted or an external LLM judge is unavailable, we also explore an alternative matching strategy based on simple lexical overlap using F1-score. In this variant, alignment between fact pairs is determined by computing the F1-score of their token overlap, and pairs exceeding a predefined threshold are marked as aligned. The table below presents the AUC-PR scores across three datasets using this heuristic method at various F1-score thresholds, where the $\mathcal{M}'$ is `LLaMA3.1-8B-Instruct` (used for the fact regeneration). For this scoring, we split the extract and regenerated facts into lists of individual words, and compute the F1-scores on these lists. Different thresholds are used (as shown in Table 7 below) to assign 0/1 labels for similar/dissimilar facts.

Although this approach is less semantically robust than LLM-based judgment, it offers a lightweight, fully offline alternative that still provides reasonable scores that are close to the main scores in our pipeline, especially in resource-constrained settings.

F1-score	LLaMA3-NQopen	LLaMA3-Hotpot	LLaMA3-WebQA
0.4	0.640	0.791	0.550
0.5	0.648	0.795	0.556
0.6	0.659	0.796	0.556
0.7	0.662	0.798	0.562
0.8	0.664	0.800	0.570

Table 7: F1-score based matching with different thresholds in fact alignment (ranging from 0.4 to 0.8)

D Pseudocode for the algorithm proposed

The hallucination detection algorithm is designed as a two-step process applied at the sentence level for a generated answer. Given a generated answer $\mathcal{A}$ and a model $\mathcal{M}'$, the goal is to produce a score for each sentence indicating the likelihood of hallucination.

In the first step as highlighted in **Algorithm 1**, the generated answer is split into sentences, and each sentence is analyzed to extract atomic facts using Named Entity Recognition (NER). For each key fact a_{ij} in sentence S_i, a corresponding question q_{ij} is generated. The model $\mathcal{M}'$ then provides an answer f_{ij} to this question. A separate **Align** function (which uses a judge LLM for fact pair comparison) evaluates whether the fact a_{ij} is consistent with the answer f_{ij}. If aligned, the fact is marked as consistent (score 0), otherwise as hallucinated (score 1). This step yields an initial binary score list for all facts.

In **Algorithm 2**, for each fact marked as consistent (score 0) in Step 1, we compute the logit scores of the top k tokens in the model's answer f_{ij}. These scores are converted into a probability distribution. We then perform a Kolmogorov–Smirnov (KS) test to statistically compare this empirical distribution against a uniform distribution. If the KS test yields a p-value less than a significance threshold (typically 0.05), the null hypothesis — that the two distributions are the same — is rejected. This indicates that the distribution is significantly different from uniform, and the fact remains marked as consistent (score 0). However, if the p-value is greater than or equal to 0.05, the distribution is considered close to uniform, signaling high uncertainty in the model's response. In this case, the fact is reclassified as hallucinated (score 1). Sentence-level hallucination scores are then calculated by averaging the final scores of all facts in the sentence.

Algorithm 1 Fact Alignment Check

Input: Generated Answer $\mathcal{A}$, Model $\mathcal{M}'$
Output: Initial Score List $[s_{ij}]$ for all facts a_{ij}

```
▷ Step 1: Sentence splitting and fact extraction
Perform coreference resolution on A and split into sentences {S_1, S_2, ..., S_N}
foreach sentence S_i in A do
    Extract atomic facts {a_ij} from S_i using NER
    foreach fact a_ij do
        Generate question q_ij ← Q(a_ij | S_i)  Get answer f_ij ← M'(q_ij)
        if Align(f_ij, a_ij) then
            Set s_ij ← 0  ▷ Fact is consistent
        else
            Set s_ij ← 1  ▷ Fact is hallucinated
return [s_ij]
```

Algorithm 2 Uniformity Check Phase (via KS Test)

Input: Initial Score List $[s_{ij}]$, Corresponding Answer Logits s_{ijk}
Output: Final Sentence Scores $[Score(S_1), \ldots, Score(S_N)]$

```
foreach sentence S_i do
    Initialize Score(S_i) ← 0
    foreach fact a_ij in S_i do
        if s_ij == 0 then
            Compute normalized probabilities:
                p(w_ijk) = e^{s_ijk} / Σ_{m=1}^{k} e^{s_ijm}
            ▷ Compare with uniform distribution
            Perform KS test between p(w_ijk) and uniform distribution
            if p-value ≥ 0.05 then
                Set s_ij ← 1  ▷ Mark as hallucinated
        Add s_ij to Score(S_i)
    Normalize: Score(S_i) ← Score(S_i) / #facts in S_i
return [Score(S_1), ..., Score(S_N)]
```

E Prompting Format

Prompt Templates Used in the Pipeline

1. Fact Regeneration Prompt (Manually Constructed Chat Format):

This prompt is used to generate fact-based questions from the given sentence. The prompt follows a constructed chat format, to be manually customized for the model in use (e.g., LLaMA3.1, Qwen2.5). It is used for each of the questions generated by the T5-finetuned model on the extract key facts.

i) Example format for `LLaMA3-8B-Instruct`:

```
'''<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a Question-answering assistant, only answer the question.
<|eot_id|><|start_header_id|>user<|end_header_id|>
Question: <insert question here>
<|eot_id|><|start_header_id|>assistant<|end_header_id|>'''
```

2. Fact Alignment Prompt (used with the judge LLM):

Few-Shot prompt used to check for alignment between extract and regenerated facts using LLM-as-a-judge. This prompt is well-structured to give the judge LLM complete understanding of how to generate the alignment output for the pairs of facts that it is applied on.

```
”’You are a fact comparison expert. Your task is to determine whether pairs of extracted and regenerated facts refer to the same real-world entity, concept, or meaning.
For each pair:
- Return ‘0‘ if the two facts refer to the same thing, even if the wording, specificity, or structure is different.
- Return ‘1‘ if the two facts do not refer to the same thing, or if their meanings conflict.
Guidelines:
- Minor differences in wording, grammar, or capitalization should be ignored.
- Partial vs full names (e.g., "Vancouver" vs "Vancouver, British Columbia") should match if they refer to the same entity.
- Aliases and synonyms (e.g., "Roger Pirates" vs "Roger crew") should count as a match.
- Abbreviations (e.g., "UCLA" vs "University of California, Los Angeles") are also matches.
- Return ‘1‘ only if clearly unrelated or ambiguous.
Format:
Return a Python-style list of exactly {n} binary values (0 or 1), corresponding to each fact pair in order.
Do not output anything else. If unsure, still return a complete list.
Examples:

  • "President Donald J. Trump" vs "Donald Trump" → 0
  • "Vancouver, British Columbia" vs "Vancouver" → 0
  • "five" vs "5 seasons" → 0
  • "UCLA" vs "University of California, Los Angeles" → 0
  • "Microsoft" vs "Apple" → 1

Now judge the following fact pairs: {pairs}
Output: ”’
```

F Step-by-Step ConFactCheck Example

Example: Question and Answer Processing Step-by-Step

Input:

`Question: Who won the FIFA World Cup in 2018?`
`Answer: The FIFA World Cup 2018 was won by France.`

Step 1: Extract sentences from the original answer

- The sentence splitter extracts:

`"The FIFA World Cup in 2022 was won by Argentina."`

Step 2: Extract Key facts using NER

- Named entities detected: "`FIFA World Cup`", "`Argentina`", "`2022`".
- Generated questions using T5-finetuned model for each key fact:

$$\begin{cases} \texttt{FIFA World Cup} & \rightarrow \text{Q1: Which tournament did Argentina win in 2022?} \\ \texttt{Argentina} & \rightarrow \text{Q2: Who won the FIFA World Cup in 2022?} \\ \texttt{2022} & \rightarrow \text{Q2: When did Argentina win the FIFA World Cup?} \end{cases}$$

Step 3: Generate pinpointed answers

- Using the LM to answer the generated questions:

Answers = ["`FIFA World Cup`", "`Argentina`", "`1978, 1986 and 2022`"]

Step 4: Compare original and regenerated answers

- `Use Huggingface QA pipeline to extract shortened pinpointed answers from original and regenerated contexts.`
- Judge if answers match (0 = match, 1 = hallucination):
  `Initial hallucination flags = [0, 0, 1]`

Step 5: Final hallucination check with probability

- `Use token-level probabilities and KS-test to confirm hallucination.`
- `Final hallucination flags remain: [0, 1, 1]`

Figure 3: Step-by-step example explaining the methodology of CONFACTCHECK