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. Such hallucinations are particularly concerning in domains such as healthcare, finance, and customer support, where incorrect information can have severe consequences. 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 to control its behavior. Existing methods to detect hallucinations in settings where the model access is restricted or constrained by resources typically require making multiple calls to the underlying LLM to check and refine the output or to sample many responses to estimate output probabilities. Such a large number of calls significantly increases latency and cost, becoming a bottleneck for adopting these methods in practical scenarios. 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 questions probing factual components of the 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 reveals the strengths of CONFACTCHECK compared to the state-of-the-art baselines. CONFACTCHECK can detect hallucinated facts efficiently using fewer resources and achieves significantly higher accuracy scores compared to existing baselines that operate under similar conditions. We will release the code and resources on acceptance.

1 Introduction

Large Language Models (LLMs) are the goto 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 occur when the model output contradicts the information in the input (or in-context instructions). Such hallucinations are relatively easier to detect by checking if the output is faithful to the provided input (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 either by improving factual accuracy through training or fine-tuning (Tian et al., 2023; Azaria and Mitchell, 2023a; Chuang et al., 2023), or by 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. As a result, recent efforts have focused on iteratively querying or prompting LLMs (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

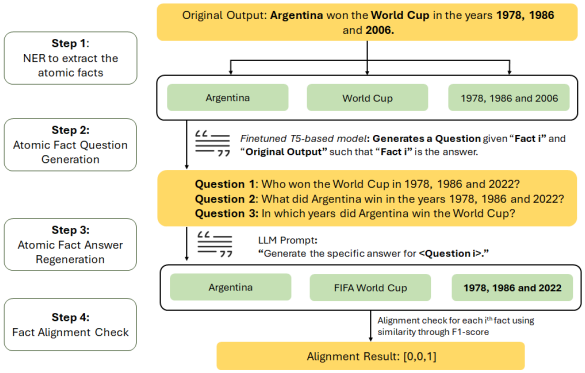

Figure 1: Key fact-based hallucination detection through the Fact Alignment check of our ConFactCheck pipeline. Each fact is used to generate a question, and the fact is regenerated by prompting the question to the LLM. The regenerated facts are compared with the original extracted key facts to check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

method for detecting hallucinations that leverages the LLM's internal knowledge and does not depend on any external knowledge source. Figure 1 illustrates the process with the help of an example. ConFactCheck identifies key entities in the generated output and then formulates contextually relevant questions around these entities. 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, by design, are prone to hallucinations (Xu et al., 2024; Ji et al., 2023) and are observed even in visual and multi-modal LLMs (Bai et al., 2024; Liu et al., 2024). Consequently, significant efforts have been made to detect hallucinations and mitigate their impact (Huang et al., 2023; Zhang et al., 2023b; Tonmoy et al., 2024). We briefly review and summarize the various techniques for hallucination detection covering methods that follow a prompt-based *self-checking* paradigm as well as methods that need access to model weights or external knowledge sources.

Zhang et al. (2023a) propose Semantic-Aware Cross-Check Consistency (SAC^3), which is a sampling-based method that checks for self-consistency of the model across multiple generations. Similarly, Manakul et al. (2023) present SelfCheckGPT, another sampling-based approach for fact-checking, which uses an LLM to generate stochastically similar outputs and scores the similarity of sampled responses with the original to self-check the LLM's confidence over the original generation. 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) analyze self-contradiction in instruction-tuned LMs by employing two separate LLMs for text generation and contradiction analysis for hallucination detection. Honovich et al. (2022) introduce TRUE, an evaluation of factual consistency measures on pre-existing texts manually annotated for factual consistency. Their study employs a range of metrics, including n-gram-based, model-based, and NLI-based evaluations, conducted on the FEVER dataset (Thorne et al., 2018). Liu et al. (2022) propose a reference-free, token-level method for detecting hallucinations. The work is supported by an innovatively curated 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) is another consistency-based approach, which breaks outputs into atomic facts, and score those through reliable *external* knowledge sources.

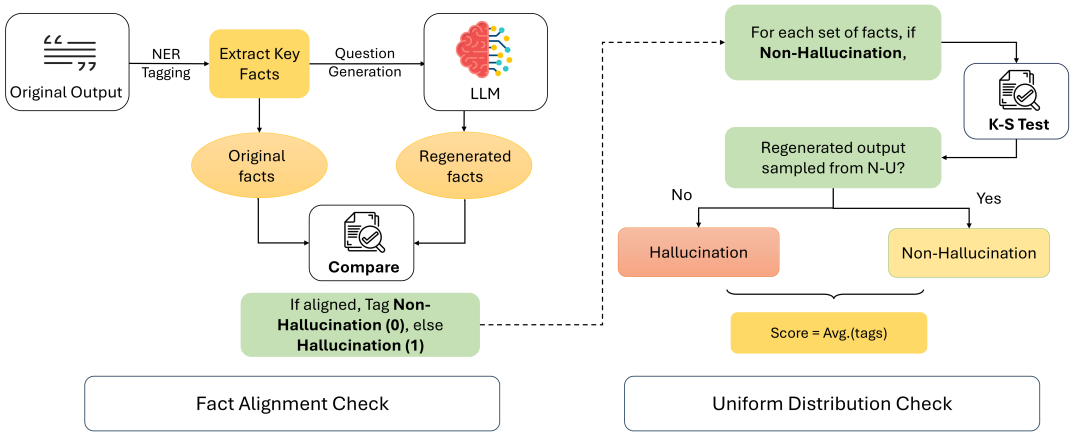

Figure 2: Pipeline of the CONFACTCHECK approach, with NER tagging of outputs followed by the first comparison-based check (Fact Alignment Check) and the secondary KS test-based probability check (Uniform Distribution Check) for rechecking the classfied non-hallucinations, result in the final tagging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

Methods Requiring Access to Model Weights and External Sources: Tian et al. (2023) showed that fine-tuning the model using factuality preference rankings can help improve factual correctness in LLM output. Azaria and Mitchell (2023b) suggest a method to assess the veracity of outputs and detect hallucinations by passing the internal states/activations of an LLM through a trained classifier to output its probabilities of truthfulness. Chen et al. (2024) propose INSIDE, an approach that leverages the internal 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. 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.

3 The CONFACTCHECK Approach

Figure 2 summarizes our proposed approach for hallucination detection. Our solution consists of two main steps – *(i)* a *fact alignment* check step where key facts present in the output text are compared with facts obtained by the targeted probing of the LLM; and *(ii)* a *uniform distribution check* step that serves as an additional check, filtering 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 that can be answered based on the information in $\mathcal{A}$. 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 dis-

cussed in Section 5.4.2. 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 $a = [a_{11} = \textit{Argentina}, a_{12} = \textit{World Cup}, a_{13} = \textit{1978, 1986 and 2006}]$.

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 $\mathcal{M}'$ generate regenerated facts f_{ij}, which are then checked for consistency with a_{ij}. This is done using F1 score-based matching to measure the similarity between f_{ij} and a_{ij}.

Let us understand this better with the help of example in Figure 1 where the set of regenerated facts is given by $f = [f_{11} = \textit{Argentina}, f_{12} = \textit{FIFA World Cup}, f_{13} = \textit{1978, 1986, and 2022}]$, while the original key facts are $a = [a_{11} = \textit{Argentina}, a_{12} = \textit{World Cup}, a_{13} = \textit{1978, 1986, and 2006}]$. In this example, f_{13} and a_{13} yield a low F1-score and are therefore classified as non-aligned, whereas the first two facts are considered aligned due to their higher scores. It should also be noted that the number of key facts per sentence varies depending on the factual content, which in turn affects the number of generated questions. Broadly speaking, this approach enables the decomposition of sentence-level information into discrete factual elements. This method operates under the assumption that the LLM's responses will remain consistent for factual information when sampled at a low temperature.

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, we use a Kolmogorov–Smirnov test on the top-5 tokens (selected heuristically) for each regenerated fact f_{ij}. 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, the second check concludes 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. In addition, this setting also tests the ability of the LLM to generate text that is *faithful* to the input context (text to be summa-

rized). We use the following four datasets for our experiments.

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 summaries 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 Mistral-7B-Instruct and LLaMA2-7B-Inst 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 implementations of HaDes[1], SelfCheckGPT[2], SAC^3[3], and INSIDE[4] for +our experiments. For SAC^3, we compute the question-level consistency SAC^3-Q score and employ thresholds as recommended in the original work 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 F1 score-based similarity. 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 Accuracy scores on the 3 open-domain QA datasets, as well as the WikiBio summarization dataset. Note that the SelfCheckGPT baseline is applicable on the WikiBio dataset, as the others deal with only the QA task and require questions as part of their input.

5 Empirical Results and Discussions

We now present results of experiments to understand how ConFactCheck compares with the SoTA baselines as well as ablation studies revealing the role of different components of the proposed pipeline.

5.1 ConFactCheck for Hallucination Detection

Table 1 summarizes the results of different methods for the four datasets and across two LLM backbones (Mistral and Llama-2). We observe that ConFactCheckoutperforms 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 HaDes achieves the second-best performance on HotpotQA (with LLaMa2) and WebQA for both LMs, SAC^3 achieves the best performance on the three QA datasets with

[1] https://github.com/microsoft/HaDes
[2] https://github.com/potsawee/selfcheckgpt
[3] https://github.com/intuit/sac3
[4] https://github.com/alibaba/eigenscore

Model	NQ Open		HotpotQA		WebQA		WikiBio	
	Mistral	LLaMA2	Mistral	LLaMA2	Mistral	LLaMA2	Mistral	LLaMA2
HaDes (Liu et al., 2022)	0.52	0.57	0.59	0.67	0.64	0.63	—	—
SAC^3 (Zhang et al., 2023a)	0.69	0.51	0.71	0.51	0.64	0.44	—	—
SelfCheckGPT (Manakul et al., 2023)	0.59	0.58	0.61	0.55	0.54	0.50	0.57	0.61
INSIDE (Chen et al., 2024)	0.67	0.59	0.59	0.57	0.60	0.47	—	—
CONFACTCHECK	**0.75**	**0.86**	**0.73**	**0.87**	**0.69**	**0.77**	**0.72**	**0.71**
% gain over best baseline	+6%	+27%	+2%	+20%	+5%	+14%	+15%	+10%

Table 1: Model Accuracy for NQ Open, HotpotQA, WebQA, and WikiBio datasets. We compare ConFactCheck in the same settings as the baselines, using Mistral-7B-Inst and LLaMa2-7B-Inst as the base models. Accuracy for CONFACTCHECKis reported for the Fact Alignment check and when beam decoding is used on the whole pipeline (this yields best possible results). The best performing method in a given column is in **bold** and the second best performing model is underlined.

Mistral as the backbone. 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 double-digit absolute percentage gains in five of the eight settings. Such strong performance of CONFACTCHECKcan be attributed to the fact that it identifies the key atomic facts in the generated text and probes the LLM regarding its knowledge around these key facts.

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. Further, 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 the fastest inference time for both the LLaMA2 and Mistral backbones. Compared to INSIDE, CONFACTCHECK offers upto ≈2x speedup (7.09s vs. 15.15s for LLaMA2; 3.53s vs. 6.21s for Mistral), and up to ≈10x speedup compared to SelfCheckGPT () (7.09s vs. 69.59s for LLaMA2; 3.53s vs. 30.76s for Mistral). 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. 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 with 10 and 5 LLM calls per question and SAC^3 with 5 and 2 LLM calls, 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 the higher number of LLM calls (i.e 10 for SelfCheckGPT and 5 for SAC^3) to exhibit their best performance.

Method	# LLM calls	Mistral	LLaMA2
SelfCheckGPT	10	30.76 s	69.59 s
SelfCheckGPT	5	8.91 s	24.81 s
SAC^3	5	13.24 s	31.85 s
SAC^3	2	7.94 s	22.7 s
INSIDE	10	6.21 s	15.15 s
CONFACTCHECK	2.6	**3.53 s**	**7.09 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 Mistral and LLaMA2 models. CONFACTCHECK offers significant speedups over the baselines.

5.3 ConFactCheck with 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 accuracy 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 all three datasets, there is a significant amount of hallucinated outputs, with only a slight decrease 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 consistent across different model sizes. While the gains, as we go from 3.8B to 13 B models, are moderate for HotpotQA and NQ-open (one and two percentage points, respectively), they are substantial for WebQA dataset (absolute gains of six percentage points).

Model	NQ Open		HotpotQA		WebQA	
	Acc	%Hall.	Acc.	%Hall.	Acc.	%Hall.
Phi-3-4b	0.54	0.75	0.62	0.76	0.47	0.81
Phi-3-7b	0.56	0.79	0.57	0.74	0.47	0.74
Phi-3-13b	**0.56**	0.71	**0.63**	0.75	**0.53**	0.78

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

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* 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 accuracy achieved by just the fact-alignment step 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 gains ranging from 16 to 38%.

Component	LLM	NQ Open	HotpotQA	WebQA
Fact Alignment	Mistral	0.59	0.60	0.50
+ Distribution Check	Mistral	**0.75**	**0.73**	**0.69**
% gain		27%	21%	38%
Fact Alignment	LLaMA2	0.72	0.75	0.60
+ Distribution Check	LLaMA2	**0.86**	**0.87**	**0.77**
% gain		19%	16%	28%

Table 4: Accuracies 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 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. This setting provides a lower bound for key fact identification accuracy. Table 5 summarizes the results for the three strategies and reveals that NER significantly outperforms both POS tagging and random token sampling to identify which tokens contribute to the factuality of a

sentence or paragraph.

Tagging	NQ Open		HotpotQA		WebQA	
	Mistral	LLaMA2	Mistral	LLaMA2	Mistral	LLaMA2
Random	0.58	0.62	0.56	**0.59**	0.49	**0.63**
POS	0.62	0.62	0.52	0.54	0.61	0.59
NER	**0.67**	**0.63**	**0.56**	0.57	**0.67**	0.61

Table 5: The Accuracy scores while using different tagging strategies on Mistral-7B and LLaMA2-7B for identifying key facts in the sentence. NER is observed to perform better in more cases over these three QA datasets.

5.4.3 Effect of Decoding Strategies

Model	NQ Open	HotpotQA	WebQA	WikiBio
Mistral (Greedy)	0.69	0.69	0.58	0.69
Mistral (Beam)	**0.75**	**0.73**	**0.69**	**0.72**
LLaMA2 (Greedy)	0.83	0.84	0.73	0.69
LLaMA2 (Beam)	**0.86**	**0.87**	**0.77**	**0.71**

Table 6: The Accuracy scores of CONFACTCHECK with Mistral-7b and LLaMA2-7b models using different decoding strategies for fact regeneration on the four datasets. Beam decoding (beam size = 5) outperforms Greedy Decoding in all of the settings.

After identifying the key facts in the generated output, the next important step in the pipeline is regenerating these key facts by targeted probing of the LLM. To generate answers to the key fact-oriented questions, we experimented with greedy and beam decoding strategies. Since greedy decoding involves selecting the token from the vocabulary with the highest conditional probability at each step, this strategy prioritizes key facts (as answers) for which the model has the highest immediate confidence. Beam decoding, on the other hand, explores multiple possible generation paths (number of paths determined by the beam_size parameter), and can thus generate key facts with the overall highest probabilities. Table 6 reports the results obtained by using the two decoding strategies, and we note that for all four datasets and the two LLM backbones, beam decoding outperforms greedy decoding. This outperformance can be attributed to the exploration of multiple possible answer paths before selecting the most likely one by beam decoding. As a result, when regenerating key facts, beam decoding ensures a more informed selection of entities, leading to a reduction in the likelihood of factual errors and ensuring better regeneration of facts. However, we do note that this decoding strategy does involve a trade-off with computational efficiency compared to greedy decoding as it now needs to evaluate multiple generation paths.

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 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.

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.

Consistent Sample Scoring: Unlike previous stochastic hallucination detection methods, such as SelfCheckGPT (Manakul et al., 2023), CONFACTCHECK does not rely on multiple LLM outputs. 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. 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

We presented CONFACTCHECK, a novel fact-based hallucination detection pipeline that does not rely on access to model weights or external knowledge sources. Empirical evaluations reveal that despite being less computationally expensive and not requiring any training, CONFACTCHECK outperforms baseline methods for both QA as well as free-form text generation tasks.

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. Consider the following question 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 Mistral-7B-Instruct 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. However, due to the presence of other hallucinated facts, our pipeline assigns a hallucinated tag to the paragraph. Moreover, we have observed that oftentimes, the generated questions may be vague, such as "Who was the building named after?" This ambiguity can result in inaccuracies when regenerating facts. The first error can be mitigated if the relevant fact to examine is known a priori. As for the second, developing an improved reverse question-answering system can help mitigate such errors.

In addition, we also note that the proposed ConFactCheck has only been tested for English language and LLMs trained mostly on English data. Hence, while theoretically, the framework should work for non-English and low-resource languages, this needs to be empirically validated. 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.

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.

A Models and Implementations

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

One of the first works addressing zero-resource hallucination detection, SelfCheckGPT is included in our comparison, with its MQAG scores presented in Table 1. We set the number of questions per sentence to 5, using the Bayes scoring method with Alpha. Both β_1 and β_2 were set to 0.95.

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

As discussed earlier, we include SAC^3 as one of our baselines and evaluate it using the instruction fine-tuned version of Mistral-7B. We compute the question-level consistency score (SAC^3-Q), which is defined in the original study as a measure of cross-check consistency between two types of QA pairs: i) the original question paired with its generated answer, and ii) multiple semantically similar generated questions paired with their respective answers. To align with our computational constraints, we experiment with two perturbed QA pairs. While this number can be adjusted for further comparisons, Zhang et al. (2023a) suggest that using between 2 to 5 perturbed questions per data sample produces comparable quantitative results.

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

HaDeS is a novel token-free hallucination detection dataset designed for free-form text generation. The dataset is constructed by perturbing raw web text using an out-of-the-box BERT model. Human annotators are then employed to assess whether the perturbed text spans constitute hallucinations based on the original text. The final model is a binary classifier that distinguishes between hallucinated and 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

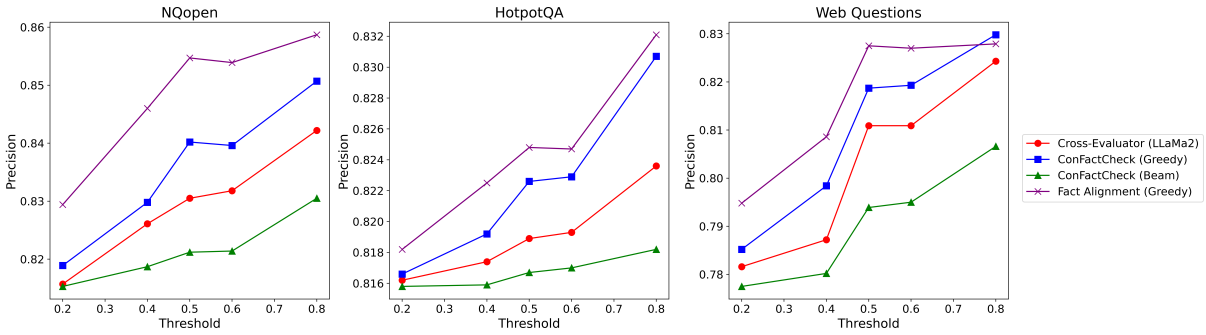

Figure 3: Precision values at varying thresholds of hallucination classification by Mistral-7B, for each of our CONFACTCHECK-based experiments. We observe that the Fact Alignment Greedy significantly outperforms others across all thresholds.

measure consistency in the dense embedding space. Specifically, the authors define a score called *EigenScore*, computed as the logarithmic determinant of the covariance matrix formed from the sentence embeddings of a set of K outputs, thereby quantifying the consistency among these embeddings. Using it as a baseline, we implement it with our settings with Mistral-7b and LLaMA2-7b as the LLMs on the 3 QA datasets and calculate the accuracy scores.

B Effects of Changing Precision Threshold

For further evaluation, we perform a threshold-based analysis on the averaged scores of each sample. Specifically, for various thresholds between 0 and 1, an output is classified as hallucinated if its score exceeds the threshold. We then plot precision values (see Figure 3) for the different settings of our approach. The results show a gradual increase in precision across all three datasets as the threshold rises. Notably, Fact Alignment consistently outperforms the other settings, suggesting that alignment without a probability check is more effective in detecting hallucinations.

C Percentage of Key Facts in Original Outputs

We also analyze the three QA datasets to determine the proportion of textual content in the original generated answers that is represented by key facts, i.e the factual information identified by NER tags in our pipeline. These results are summarized in the table below, providing insight into the representation of key facts in the outputs.

Model Name	% Atomic Facts/Output		
	NQ-Open	HotpotQA	Web Questions
Mistral-7B	27.53%	13.10%	21.61%
LLaMA2-7B	10.23%	10.22%	8.4%

Table 7: Percentage of Key Facts/Output for different models across the Question-Answering datasets.

D Usage of CONFACTCHECK on Datasets

D.1 Open-Domain Question Answering

For this task, we evaluate CONFACTCHECK on three datasets. CONFACTCHECK is applied to the original outputs generated for each question to determine if the LLMs have produced any hallucinations. The process operates at the sentence level: outputs are segmented into individual sentences, key facts are extracted from each sentence, and CONFACTCHECK then assesses the factual consistency of these extracted facts.

D.2 Text-Based Summarization

For text-based summarization, we use the WikiBio dataset, which includes summaries of individuals from Wikipedia alongside synthetic summaries generated by GPT-3. CONFACTCHECK is employed as a sentence-level detector on the synthetic summaries, where each sentence is annotated with a hallucination label provided by the dataset. We obtain sentence-level hallucination scores and compare them against the gold-standard annotations. For passage-level evaluation, the overall hallucination score is computed by averaging the sentence-level scores.

E Pseudocode for the algorithm proposed

For an originally generated LLM output, we first split it into sentences $S_1, S_2, \ldots, S_N$. Then for each sentence, we use NER (POS can also be used) to extract named entities which act as the key facts a_{ij} from each sentence (i and j represent the i^{th} sentence and j^{th} entity of that sentence respectively).

We then proceed with the Fact Alignment Check, where the LLM $\mathcal{M}'$ is prompted with the T5 model-based entity-specific questions to regenerate the key facts as f_{ij}. Then f_{ij} and a_{ij} are compared by F1-score similarity to check if they align or not. If they are aligned we tag a_{ij} as 0 (i.e consistent or a non-hallucination). Then we implement the second probability-based check (i.e the Uniform Distribution Check), where each aligned fact from the first check is rechecked using the KS test as described in Section 3. Following this, we obtain a set of scores for each of the j key facts within the i^{th} sentence. These are then averaged to provide a final score for the particular output sentence.

Algorithm 1 Hallucination detection score

procedure CALCULATESCORE($\mathcal{A}, \mathcal{M}$)
  Perform coreferencing on $\mathcal{A}$ and break it into sentences $S_1, S_2, \ldots, S_N$
  Set $Score(S_i)$ to 0 for $i \in \{1, \cdots, N\}$.
  for $i \leftarrow 1$ **to** N **do**
    Tag each sentence S_i with NER entities to extract atomic facts a_{ij} for j entities
    for all a_{ij} in S_i **do**
      $q_{ij} = Q(a_{ij}|S_i)$
      $f_{ij} = \mathcal{M}'(q_{ij})$
      if align(f_{ij}, a_{ij}) **then**
        Tag a_{ij} as 0 (consistent)
        for token w_{ijk} in f_{ij} **do**

$$s_{ijk} = \text{logitScore}(w_{ijk}|\text{vocab}(\mathcal{M}'))$$

where:

- $|s_{ijk}| = |\text{vocab}(\mathcal{M}')|$
- $w_{ijk} \in \text{vocab}(\mathcal{M}')$.
- logitScores : $\text{vocab}(\mathcal{M}') \to \mathbb{R}$

Compute normalized-probabilities of top-5 tokens:

$$p(w_{ijk}) = \frac{e^{s_{ijk}}}{\sum_{m=1}^{5} e^{s_{ijm}}}, \quad \text{for } k = 1, 2, \ldots, 5$$

if $p_{ijk} \sim$ Uniform **then**
Tag a_{ij} as 1
break
end if
end for
else
Tag a_{ij} as 1 (hallucinated)
end if
$Score(a_{ij}) \leftarrow$ Tag of a_{ij} (0 or 1)
$Score(S_i) \leftarrow Score(S_i) + Score(a_{ij})$
end for
$Score(S_i) \leftarrow \frac{Score(S_i)}{|\text{entities in } S_i|}$ ▷ Normalize score by number of entities
end for
return $[Score(S_1), Score(S_2), \ldots, Score(S_N)]$
end procedure