Project Overview on Replication and Extension of SAC3

Xiaojing Zhang
olatname: xiaojing
Matriculation No.: 22741763

Introduction

Detecting hallucinations in large language models (LLMs) is a critical challenge in natural language processing (NLP). Hallucinations, which occur when models generate confident but incorrect or unverifiable responses, can severely impair model performance and pose significant risks in applications that depend on high factual accuracy, such as machine translation, summarization, and question-answering. While various methods have been proposed to address this issue, ranging from confidence-based evaluations to external resource checks, many are not suitable for use with commercial black-box models that provide only API access. Recent studies have employed self-consistency to detect hallucinations based on pretrained LMs (Manakul et al., 2023) and instruction-tuned LMs (Mündler et al., 2023), but it often fails because models often generate consistently incorrect answers or generate varied, non-factual responses from stochastic sampling.

Addressing these drawbacks, the paper "SAC3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency" presents SAC3, an innovative method that enhances traditional self-consistency checks (SC2) by integrating semantically equivalent question perturbations and cross-model response consistency. This approach seeks to boost the trustworthiness of language models by more effectively detecting hallucinations at both the question and model levels (Zhang et al., 2023).

In this project, I replicated and expanded the SAC3 approach detailed in the paper, applying it to a broader range of models, including those with different parameter sizes not originally covered by the study. This extension tested SAC3's robustness and scalability. The results confirmed that the SAC3 method effectively detects hallucinations across different models, demonstrating its generalizability and reliability.

My Work

My work consists of three parts:

I. Preparation Work

Before replicating and extending the SAC3 method, I took several preparatory steps to ensure a thorough understanding and effective application of the study:

• **Literature Review and Testing:** I read the paper repeatedly to fully grasp the SAC3 methodology and its implications. To understand how SAC3 works with different types of questions, I tested ChatGPT with some questions provided in the paper, as well as additional questions from diverse fields such as sports, law and literature. This helped me understand hallucination better and confirmed the general applicability of SAC3.

• **Technical Setup and Code Familiarization:** I studied the code provided by the authors to understand its structure and functionality. Then I made some necessary modifications to integrate their scripts into a Google Colab notebook, which involved adjusting file handling and execution environments to suit the cloud-based platform.

II. Replication Work

The replication phase was critical in assessing the robustness and applicability of the SAC3 methodology outlined in the paper. In this phase, I focused on maintaining the comparability

of the original experiments using GPT-3.5-turbo while exploring the methodology's scalability and effectiveness in new scenarios.

Step 1: Initial Dataset Testing
Initially, I tested two datasets designed for binary classification QA tasks—prime number identification and senator search tasks—using both AUROC and accuracy metrics. The testing with AUROC employed a balanced dataset comprising 50% factual and 50% hallucinated samples, while accuracy testing utilized an imbalanced dataset with 100% hallucinated samples. Additionally, I assessed an open-domain QA dataset, HotpotQA-halu, for generation QA tasks using AUROC with a balanced dataset setup. Preliminary tests with 10 data points per dataset closely matched the results reported in the original paper, confirming the method's robustness.

Step 2: Deepening Method Understanding
To further understand SAC3's efficacy, I tested ChatGPT-3.5-turbo under various settings—varying the number of self-responses (3, 5, 10, and 15 responses per question in self-checking) and the number of perturbed questions (5 and 10 semantically equivalent questions in cross-checking). While the performance of sampling-based methods typically improves with sample size, I observed diminishing returns after certain thresholds, which aligns with theoretical expectations but also highlights practical limitations in terms of time and computational resources.

Step 3: Expanding Model Testing
While attempting to test larger models like Guanaco-33b and Falcon-7b, I encountered hardware limitations, with sessions frequently crashing due to insufficient RAM. Fortunately, Falcon-7B tests were successful.

Given fluctuating performances in the initial tests with 10 data points, I expanded the dataset to 50 questions, focusing on the open-domain QA dataset (HotpotQA-halu). This expanded testing revealed that cross-checking significantly outperformed self-checking.

Table 1 and Table 2 below illustrate how the AUROC scores varied with different configurations. Notably, the AUROC score for cross-checking with 10 perturbed questions was substantially higher than the score reported in the original study (86.96 compared to 81.3), suggesting potential overfitting with smaller datasets.

Method	AUROC Score (SC2)
Reported SC2 (self-checking)	74.2
SC2 for 3 self-responses	74.4
SC2 for 5 self-responses	75.36
SC2 for 10 self-responses	74.08
SC2 for 15 self-responses	78.24

Table 1: Self-Checking AUROC Score Comparison illustrates the scores obtained when varying the number of self-responses in the self-checking setup.

Method	AUROC Score (SC2)
Reported SAC3 (cross-checking) for 10 perturbed questions	81.3
SAC3 for 5 perturbed questions	81.5
SAC3 for 10 perturbed questions	86.96

Table 2: Cross-Checking AUROC Score Comparison shows the outcomes when varying the number of perturbed questions in the cross-checking setup.

This discrepancy suggested possible overfitting, which seemed more apparent when testing was expanded to 100 questions, with AUROC scores dropping to between 77 and 79, much lower than the score reported in the paper (81.3).

Subsequent testing with GPT-4.0 yielded similar results, reinforcing the likelihood of overfitting. Nevertheless, cross-checking always outperformed self-checking, showing that SAC3 can be replicated.

Given these findings, and considering the substantial costs and computational demands of testing more data points, I focused on evaluating open-source large language models in the next phase to further validate the SAC3 method under varying conditions and with potentially more scalable configurations.

III. Extended Work

In my exploration of various language models, I encountered significant memory constraints with models exceeding 10 billion parameters (e.g., Guanaco-13b, Llama-3-13b, Command-R+). Attempts to employ vLLM, a memory-efficient serving engine, failed due to its requirements for a GPU with computing capability >=7.0, which exceeded the capabilities of my setup. Consequently, I focused on models with fewer than 10 billion parameters that could be managed on CPU.

Testing Smaller Models:

I proceeded to test smaller language models, but some revealed unique challenges:

- **Gemma-2B:** Generated three options for user to choose instead of a single response.
- **Vicuna:** Generated outputs with repeated characters and special tokens without meaning.
- **Zephyr:** Automatically added irrelevant questions to the responses.
- **DeciLM:** Testing sessions frequently crashed.

Despite these obstacles, I managed to successfully conduct tests on models such as the latest Llama-3-8B-Instruct, Mistral-7B-Instruct-v0.2, Phi-3-mini-128k-instruct, Gemma-7B-Instruct, and Qwen1.5-7B-Chat, with parameters ranging from 3.8 billion to 8 billion. They are all open-source language models on HuggingFace, released by different companies. These models were evaluated using the HotpotQA-halu dataset, measuring performance through the AUROC metric.

I did the following to test these five models:

Step 1: Initial Testing with Single Questions

I modified the codes for each model, i.e., adding an API call function to query the model and get response based on the example code on HuggingFace and revising relevant codes in consistency check and evaluation. Then I tested the model with single questions, one from binary classification task and one from open-domain generation task, to ensure the model work as expected and to see its response format.

Step 2: Further Testing with Dataset (10 Data Points)

Based on the response format, I further modified the codes as the format varies from model to model. Some models included the prompt in its response, such as Mistral-7B; some models added other content before the relevant answer; some models used special tokens like "[/INST] in its response. To solve this issue, I either modified the function to call the model's API (such as Gemma-7B and Mistral-7B) or the relevant codes in consistency check to extract the answer (e.g., Phi-3-mini and Qwen1.5-7B). Initially, the AUROC score of Phi-3-mini in self-checking was only 50, but after the modification in consistency check to extract the response accurately, the AUROC score increased to 80.

Furthermore, considering model-specific characteristics, I customized the prompt for each model in the API call function and consistency check. In the API call function, I provided example question and answer in the prompt. For some models, I gave concrete example question and answer such as “Between Quentin Tarantino and Sofia Coppola, who has directed more films?”, while for models like Mistral-7B, Phi-3-mini and Qwen1.5-7B, I just provided vague example question like “Between XXX and XXX, which is better?”; otherwise, these models would generate example answers in subsequent testing.

Since some models are not sensitive to the original prompt in the consistency check part provided by the authors (for example, the prompt requires responding with Guess: Yes/No to the question if the two question-answer pairs are semantically equivalent, but Mistral-7B and Phi-3-mini always respond with answers to the questions (not Yes or No), I modified the original prompt according to the responses of different models, stressing the format of their responses. Gemma-7B always gave “Yes” even if the two question-answer pairs are apparently not semantically equivalent (one answer is “Yes” while the other is “No”). For this, I added two sentences to the original prompt requesting the model to compare both the questions and answers to see if the two pairs are semantically equivalent. The performance improved significantly after the modifications in the prompt.

Subsequently, I proceeded to test these five models with the open-domain QA dataset (HotpotQA-halu). Initially, I tested 10 and 20 data points, respectively, but testing with 20 data points failed with all models due to memory and GPU constraints. Experiments using random sampling with 10 data points were successful for all models.

Results show that SAC3 worked with all five models, achieving higher AUROC scores with cross-checking than with self-checking. AUROC Scores with cross-checking ranged from 56% to 94%. Phi-3-mini performed surprisingly well, with an AUROC score of 80% in self-check and 86-90% in cross-check, at much faster speeds.

I also explored different numbers of self-responses and perturbed questions when testing the five models. Self-checking had the highest AUROC scores with 5 or 10 responses to the original question, while cross-checking had the highest AUROC scores with 10 perturbed questions (except for Mistral-7B, which peaked at 5 and 8).

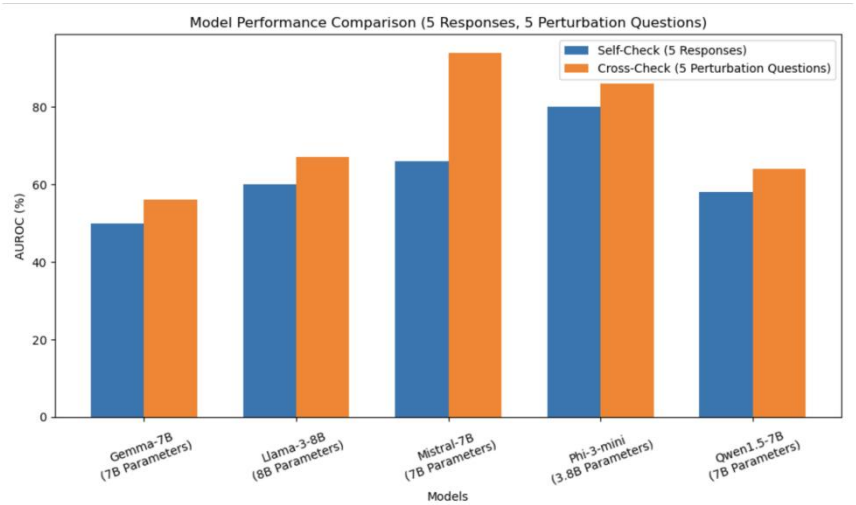

Chart 1: Comparing AUROC scores for 5 responses (self-checking) vs. 5 perturbed questions (cross-checking)

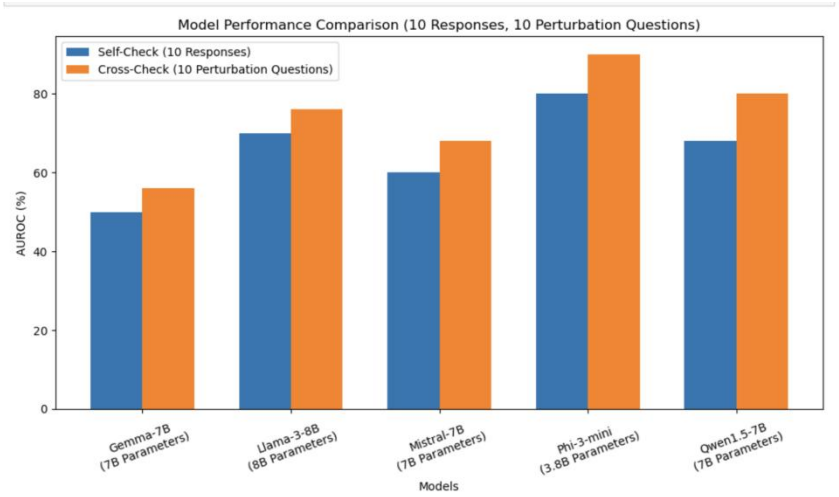

Chart 2: Comparing AUROC scores for 10 responses (self-checking) vs. 10 perturbed questions (cross-checking)

Step 3: Testing with Bootstrap

To test more data, I went on to use bootstrap sampling to achieve more reliable results. Each model was tested by randomly sampling 10 questions at a time, repeated five times to aggregate 50 data points.

I tested 5 and 10 responses in self-checking and 5 perturbed questions in cross-checking. Experiments with 10 perturbed questions were also conducted.

Model	Self-Checking (SC2)	Standard Deviation	Cross-Checking (SAC3)	Standard Deviation
Mistral-7B-Instruct-v0.2	55.20	0.1107	60.80	0.1468
Phi-3-mini-128k-instruct	46.00	0.1625	78.00	0.1513
Gemma-7B-it	62.00	0.0748	69.20	0.0943
Meta-Llama-3-8B-Instruct	64.00	0.1824	70.00	0.1397
Qwen1.5-7B-Chat	61.20	0.1603	65.20	0.1542

Table 3: Comparing AUROC scores for 5 responses (self-checking) vs. 5 perturbed questions (cross-checking) with Bootstrap

Model	Self-Checking (SC2)	Standard Deviation	Cross-Checking (SAC3)	Standard Deviation
Mistral-7B-Instruct-v0.2	62.80	0.2160	70.80	0.1129
Phi-3-mini-128k-instruct	58.00	0.0400	68.80	0.1024
Gemma-7B-it	46.00	0.1625	58.80	0.1001
Meta-Llama-3-8B-Instruct	59.20	0.1281	61.00	0.1130
Qwen1.5-7B-Chat	50.00	0.2067	58.40	0.1015

Table 4: Comparing AUROC scores for 10 responses (self-checking) vs. 10 perturbed questions (cross-checking) with Bootstrap

We can see from Table 3 and 4 that the AUROC scores were significantly lower than the scores with 10 data points. Meanwhile, the standard deviation for both self-checking and cross-checking is very high, ranging from 0.0400 to 0.2160. The variability in performance, as evidenced by high standard deviations in AUROC scores, underscored the need for further testing to achieve more stable and reliable results.

Conclusion and Future Work:

The extensive replication and expansion of the SAC3 method through this project have further confirmed its efficacy in detecting hallucinations across various language models, illustrating its robustness and scalability. The method's ability to consistently outperform traditional self-consistency checks, especially in black-box model scenarios, has been validated across different model sizes and configurations.

Key Findings:

• ***Robustness and Scalability:*** SAC3 proved effective across an expanded range of models, demonstrating its applicability and reliability in various real-world scenarios. This adaptability highlights SAC3's potential as a critical tool for enhancing the trustworthiness of language models in NLP applications.

• ***Performance Variability:*** The project underscored the need for more extensive data to stabilize the detection performance, as evidenced by high standard deviations in AUROC scores across different tests. This variability suggests that larger datasets could help in achieving more consistent and reliable results.

Future Directions:

• ***Optimizing Computational Efficiency:*** Future work should focus on enhancing computational strategies to handle larger datasets and more complex model configurations without compromising performance. This includes optimizing code and possibly integrating more efficient algorithms to reduce computational overhead.

• ***Expanding Data and Model Testing:*** To further validate and refine the SAC3 method, expanding the range of test scenarios and incorporating diverse model architectures will be crucial. This could involve collaborations with other research institutions or industry partners to access a wider array of data and computational resources.

• ***Addressing Overfitting and Response Variability:*** Continued efforts are needed to address issues of overfitting and to adapt the SAC3 methodology for consistent performance across varying conditions. This may involve developing more nuanced perturbation strategies or exploring different ensemble techniques to enhance response consistency.

This project has laid a substantial groundwork for advancing hallucination detection in language models, setting a precedent for future research in this area of NLP. The findings not only reinforce the necessity of robust detection mechanisms but also open avenues for significant improvements in how these models are deployed in high-stakes environments.

References:

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley Malin, Sricharan Kumar. 2023. SAC3: Reliable Hallucination Detection in Black-Box Language Models via Semantic-aware Cross-check Consistency. *arXiv:2311.01740v2.*

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896.*

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