FactSelfCheck: Fact-Level Black-Box Hallucination Detection for LLMs

Albert Sawczyn[1] **Jakub Binkowski**[1] **Denis Janiak**[1]

Bogdan Gabrys[2] **Tomasz Kajdanowicz**[1]

[1]Wrocław University of Science and Technology
[2]University of Technology Sydney
albert.sawczyn@pwr.edu.pl

Abstract

Large Language Models (LLMs) frequently generate hallucinated content, posing significant challenges for applications where factuality is crucial. While existing hallucination detection methods typically operate at the sentence level or passage level, we propose FactSelfCheck, a novel black-box sampling-based method that enables fine-grained fact-level detection. Our approach represents text as knowledge graphs consisting of facts in the form of triples. Through analyzing factual consistency across multiple LLM responses, we compute fine-grained hallucination scores without requiring external resources or training data. Our evaluation demonstrates that FactSelfCheck performs competitively with leading sampling-based methods while providing more detailed insights. Most notably, our fact-level approach significantly improves hallucination correction, achieving a 35% increase in factual content compared to the baseline, while sentence-level SelfCheckGPT yields only an 8% improvement. The granular nature of our detection enables more precise identification and correction of hallucinated content.

1 Introduction

Large Language Models (LLMs) have gained significant attention from academia and industry recently. However, a major limitation of LLMs is their tendency to generate hallucinated information (Farquhar et al., 2024; Huang et al., 2025), posing significant challenges for applications where factual correctness is crucial, such as healthcare (Sallam, 2023). Although numerous methods have been proposed to reduce hallucinations (Zhang et al., 2023), it is not possible to eliminate them, and LLMs will constantly hallucinate (Lee, 2023; Xu et al., 2024). Therefore, there remains a critical need for reliable hallucination detection in LLM responses. Effective detection enables system interventions by either preventing the transmission of hallucinated content to users or facilitating its correction (Zhang et al., 2023).

Previous approaches to hallucination detection have primarily focused on classifying hallucinations at either the passage or sentence level (Huang et al., 2025). While valuable, these approaches are limited in their granularity, as they do not provide detailed information about specific hallucinated facts. To address this limitation, we propose a novel method for hallucination detection that operates at the fact level, offering finer-grained analysis. In our approach, we define a fact as a triple consisting of a head, relation, and tail – a standard representation in knowledge graphs (e.g., (*Robert Smith*, *member of*, *The Cure*)) (Hamilton et al., 2017). Our method provides more precise and actionable information by computing hallucination scores for individual facts than traditional passage-level or sentence-level classification approaches. Moreover, our method represents facts in a structured way through knowledge graphs (KGs), enabling more straightforward interpretation, processing, and verification of factual content (Pan et al., 2024).

Our granular approach is motivated by two key observations. First, a single sentence or text passage can contain multiple facts, with the number of facts varying significantly across sentences, contexts, and domains. This variability makes it challenging to identify hallucinated aspects of generated output precisely when using sentence-level or passage-level detection. Second, false information can be dispersed throughout a text, as a single fact may appear across multiple sentences. That can mislead the sentence-level detection, as the factuality of a sentence is dependent on the previous sentences (Zhang et al., 2024). These challenges become particularly acute when analyzing long texts generated by LLMs. Fine-grained fact-level detection provides a more precise understanding of text factuality than a sentence or passage analysis. It enables better assessment of content reliability

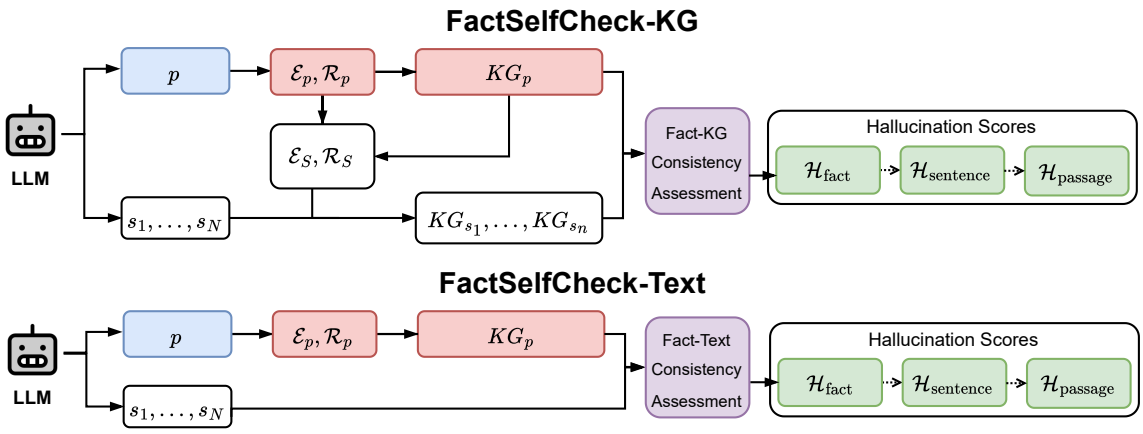

Figure 1: The pipeline of FactSelfCheck in two variants. For response p, entities $\mathcal{E}_p$ and relations $\mathcal{R}_p$ are extracted, followed by the construction of knowledge graphs KG_p, for which hallucination scores $\mathcal{H}_{\text{fact}}$ are calculated. Samples' entities $\mathcal{E}_S$ and relations $\mathcal{R}_S$ are created by merging $\mathcal{E}_p$ and $\mathcal{R}_p$ with entities and relations from KG_p. For each sample s, the knowledge graph KG_s is extracted. FactSelfCheck-KG assesses the consistency between a fact and all KG_s. FactSelfCheck-Text assesses the consistency between a fact and all s directly. To obtain sentence-level $\mathcal{H}_{\text{sentence}}$ and passage-level $\mathcal{H}_{\text{passage}}$ scores, fact-level scores are aggregated, as indicated by dashed arrows.

and, as we show later, more effective factuality correction.

We propose FactSelfCheck, a black-box method for fact-level hallucination detection, meaning it does not require access to the model's internal parameters. This design choice makes our approach universally applicable across any LLM, including closed models, like GPT (OpenAI et al., 2024). Following a sampling-based detection paradigm, introduced by Manakul et al., 2023, our method utilizes multiple response generations and analyzes the factual consistency of extracted facts across these samples. This paradigm is based on the phenomenon that factual information remains largely consistent across different generations, while hallucinated content tends to vary or contradict itself between samples (Manakul et al., 2023; Wang et al., 2023). This way, we can effectively identify hallucinated facts without relying on external resources (zero-resource) or access to the model's internal parameters (black-box). Moreover, our method is non-parametric, as it does not require any training, making it easy to implement in any domain without the need for training data. The FactSelfCheck pipeline consists of three main steps: knowledge graph extraction, which extracts sets of facts from the initial response and samples; fact-level hallucination scoring; and calculating sentence-level and passage-level scores by aggregating fact-level scores.

We evaluated our method using the WikiBio GPT-3 Hallucination Dataset (Manakul et al., 2023), aggregating fact-level scores into sentence-level and passage-level scores. Our approach achieves performance comparable to leading sampling-based methods while providing more detailed information about hallucinations. Additionally, we demonstrate that our fact-level approach significantly improves hallucination correction. Compared to a baseline, providing incorrect facts to the correction method leads to a 35% increase in factual content, while passing incorrect sentences leads to an 8% increase.

Our key contributions are as follows:

1. The novel black-box sampling-based method for fact-level hallucination detection – FactSelfCheck. It enables fine-grained hallucination detection in LLM responses without requiring training data or external resources, as it is both non-parametric and zero-resource.

2. The effective approaches for measuring factual consistency across multiple samples: FactSelfCheck-KG using knowledge graph comparisons and FactSelfCheck-Text using direct text comparison.

3. Comprehensive evaluation of our method, which shows competitive performance with leading sampling-based methods while providing more detailed insights.

4. Demonstration that fact-level detection significantly improves hallucination correction compared to sentence-level approaches.

Our code is available on GitHub [1].

2 Related work

Xu et al., 2024 have proven that hallucinations are inevitable in LLMs. As LLMs are powerful tools, many recent studies have been conducted regarding hallucination mitigation and detection (Zhang et al., 2023; Huang et al., 2025). The detection methods can be divided into two groups: white-box and black-box.

White-box methods analyze LLMs' internal states (Farquhar et al., 2024; Azaria and Mitchell, 2023). While these methods are universal across all LLMs, they often require multiple generations, similar to sampling-based methods. Notable approaches include: SAPLMA (Azaria and Mitchell, 2023), which predicts from hidden states whether generated text is correct or incorrect; INSIDE (Chen et al., 2024), which evaluates hidden state consistency across generations; SEPs (Kossen et al., 2024) that predict entropy directly from model hidden states; Lookback Lens (Chuang et al., 2024) and AttentionScore (Sriramanan et al., 2024) that uses attention maps to detect hallucinations.

Black-box approaches operate without access to the model's internal states and aim to detect hallucinations based solely on the text generated by LLMs. Some of these methods use external resources to collect evidence (Min et al., 2023; Chern et al., 2023). Others leverage LLMs to detect hallucinations like CoVe (Dhuliawala et al., 2024), which utilizes the chain-of-thought paradigm for detection. Another category is sampling-based methods, such as SelfCheckGPT (Manakul et al., 2023), which evaluate factuality by generating multiple responses (stochastic samples) and assessing consistency between the original response and these samples.

The most popular approach is to classify hallucinations at sentence-level or passage-level (Huang et al., 2025). Few methods have been specifically designed to detect hallucinations at the fact level (where facts are defined as triples). GraphEval (Sansford et al., 2024) generates a KG from LLM output and compares it with the context provided in the LLM input. FactAlign (Rashad et al., 2024) builds KGs from input and output, then compares them after performing entity alignment, a technique that pairs the same entity in different KGs. Most similar to our approach is GCA (Fang et al., 2024), which constructs KGs from the response and samples and then compares them by aggregating multiple scores. GCA has significant methodological concerns - they tuned several parameters directly on the evaluation set (which is the only set in this dataset), thus we do not compare against it despite using the same dataset. Moreover, our method belongs to a fundamentally different class as it is zero-shot, requiring no parameter tuning.

[1] https://github.com/graphml-lab-pwr/FactSelfCheck license: CC BY-SA 4.0

3 Method

We propose FactSelfCheck, a black-box sampling-based method for fact-level hallucination detection, as illustrated in Figure 1. We list used prompts in Appendix F.

3.1 Notation

Let p denote the initial response passage generated by the LLM to a user query, which we aim to evaluate for hallucinations. Let $S = \{s_1, \ldots, s_N\}$ represent a set of N stochastic LLM response samples. The text passage p consists of a set of sentences U. For each sentence $u \in U$ and each sample $s \in S$, we extract knowledge graphs KG_u and KG_s, respectively. Each knowledge graph comprises a set of facts, where a fact f is defined as a triple (h, r, t) consisting of a head h, relation r, and tail t, e.g. (*Robert Smith*, *member of*, *The Cure*). We define $KG_p = \bigcup_{u \in U} KG_u$ as the knowledge graph consisting of all facts from the passage p.

Our objective is to compute a fact-level hallucination score $\mathcal{H}_{\text{fact}}$ for each fact f in KG_p. Subsequently, to facilitate comparisons with other methods, we aggregate these scores to obtain a sentence-level hallucination score $\mathcal{H}_{\text{sentence}}$ for each sentence u and a passage-level hallucination score $\mathcal{H}_{\text{passage}}$ for the entire passage p.

3.2 FactSelfCheck pipeline

As shown in Figure 1, the pipeline of FactSelfCheck consists of three main steps: (1) **Knowledge Graph Extraction** that extracts sets of entities, relations, and finally, knowledge graph from the initial response p and samples S; (2) **Fact-level Hallucination Scores**, that score facts by measuring factual consistency between facts in KG_p and, depending on the variant, KG_s in FactSelfCheck-KG or directly s in FactSelfCheck-Text; (3) **Sentence-Level and Passage-Level Scores** calculation by aggregation of the fact-level scores.

3.3 Knowledge Graph Extraction

We adopt an approach that decomposes the knowledge graph extraction task into simpler subtasks, similarly to Edge et al., 2024. This process involves three primary steps: extracting entities, identifying relations, and formulating facts, which are implemented as a sequence of LLM prompts.

For each instance, we extract a list of entities $\mathcal{E}_p$ by passing p to the LLM (Prompt 1).

$$\mathcal{E}_p = LLM_{\text{entities}}(p) \tag{1}$$

Next, we provide p and $\mathcal{E}_p$ to the LLM to extract relation types between entities, resulting in $\mathcal{R}_p$ (Prompt 2).

$$\mathcal{R}_p = LLM_{\text{relations}}(p, \mathcal{E}_p) \tag{2}$$

We then input $\mathcal{E}_p$, $\mathcal{R}_p$, and each sentence $u \in U$ into the LLM to extract the knowledge graph KG_u (Prompt 3). We also provide an initial response p to add contextual information. The output is a set of facts:

$$KG_u = LLM_{\text{sentence\_facts}}(u, p, \mathcal{E}_p, \mathcal{R}_p) \tag{3}$$

After extracting KG_u for each sentence $u \in U$, we compile the sets of entities $\mathcal{E}_S$ and relations $\mathcal{R}_S$ required for extracting knowledge graph KG_s from each sample s (Prompt 4)[2]:

$$\mathcal{E}_S = \mathcal{E}_p \cup \bigcup_{u \in U} \{h, t \mid (h, r, t) \in KG_u\} \tag{4}$$

$$\mathcal{R}_S = \mathcal{R}_p \cup \bigcup_{u \in U} \{r \mid (h, r, t) \in KG_u\} \tag{5}$$

$$KG_s = LLM_{\text{sample\_facts}}(s, \mathcal{E}_S, \mathcal{R}_S) \tag{6}$$

Extracting KG_s by utilizing $\mathcal{E}_S$ and $\mathcal{R}_S$ is more convenient than extraction without them, as it eliminates the need for entity alignment and ensures that the KG is built using the same schema as KG_p.

[2]Although we could theoretically use $\mathcal{E}_p$ and $\mathcal{R}_p$ for KG_s extraction, in practice, LLMs are not sufficiently accurate to extract all entities and relations from the response p when calculating $\mathcal{E}_p$ and $\mathcal{R}_p$. This results in KG_u containing entities and relations not present in $\mathcal{E}_p$ and $\mathcal{R}_p$, even if the prompt restricts them. Empirical tests showed that extending $\mathcal{E}_S$ and $\mathcal{R}_S$ by adding entities and relations from all KG_u improved the results.

3.4 Fact-Level Hallucination Scores

We define two variants for measuring fact-level hallucination scores. The first variant, FactSelfCheck-KG, assesses the consistency between a fact and the knowledge graphs extracted from samples. The second variant, FactSelfCheck-Text, evaluates the consistency between a fact and the samples directly.

3.4.1 FactSelfCheck-KG

In the FactSelfCheck-KG variant, we introduce two metrics to assess the reliability of each fact.

Frequency-Based Hallucination Score The frequency-based fact-level hallucination score is based on the intuition that the probability of a fact being hallucinated is inversely proportional to the fraction of samples containing the same fact.

$$\mathcal{H}_{\text{fact}}(f) = 1 - \frac{1}{|S|} \sum_{s \in S} \mathbb{I}\{f \in KG_s\} \tag{7}$$

where $\mathcal{H}_{\text{fact}}(f)$ is the hallucination score for fact f, and $\mathbb{I}\{f \in KG_{s_n}\}$ is an indicator function that equals 1 if fact f appears in KG_{s_n} and 0 otherwise. The higher the $\mathcal{H}_{\text{fact}}$ value, the higher the plausibility of hallucination.

LLM-Based Hallucination Score To allow semantic matching and reasoning over knowledge graphs, rather than only exact fact matching, we introduce the LLM-based fact-level hallucination score. We instruct the LLM to determine whether each fact is supported by the knowledge graphs extracted from the samples (Prompt 5). The LLM is expected to respond with 'yes' or 'no'. We then average the valid responses from the LLM to get the final score as in Equation (8). Any invalid responses are not included in the averaging.

$$\mathcal{H}_{\text{fact}}(f) = \frac{1}{|V_f|} \sum_{s \in V_f} \Psi(f, KG_s) \tag{8}$$

where V_f represents the set of samples with valid LLM responses for the fact f, and the function Ψ is defined as follows:

$$\Psi(\cdot) = \begin{cases} 0 & \text{if the LLM returns 'yes'} \\ 1 & \text{if the LLM returns 'no'} \end{cases} \tag{9}$$

3.4.2 FactSelfCheck-Text

In the FactSelfCheck-Text variant, we check if a fact is supported by each textual sample directly

without using the knowledge graphs. We prompt the LLM to evaluate whether a fact f is supported by the textual sample s (Prompt 5). As in the previous variant, we average the valid LLM responses using the Ψ function:

$$\mathcal{H}_{\text{fact}}(f) = \frac{1}{|V_f|} \sum_{s \in V_f} \Psi(f, s) \tag{10}$$

3.5 Sentence-Level and Passage-Level Hallucination Scores

While detecting hallucinations at the fact level offers fine-grained insights, there are scenarios where sentence-level or passage-level detection is necessary. To achieve this, we aggregate fact-level scores to compute sentence-level scores $\mathcal{H}_{\text{sentence}}(u)$ and a passage-level score $\mathcal{H}_{\text{passage}}(p)$.

$$\mathcal{H}_{\text{sentence}}(u) = \text{Agg}_{f \in KG_u} \mathcal{H}_{\text{fact}}(f) \tag{11}$$

$$\mathcal{H}_{\text{passage}}(p) = \text{Agg}_{u \in U} \mathcal{H}_{\text{sentence}}(u) \tag{12}$$

where u represents a single sentence, and U is the set of sentences of the response p. The aggregation function Agg can be based on operations such as $mean$ or max. The $mean$ function provides a smoothed factuality score, whereas max is sensitive to the most hallucinated facts or sentences. Additionally, sentences without facts are excluded from the aggregation when calculating the passage-level score.

As discussed in Section 4.1, the dataset choice influences the calculation of the passage-level score in Equation 12. Since the ground truth scores are derived by averaging sentence-level scores, we also employ the $mean$ function for aggregation. It is worth mentioning that a more intuitive approach would be to average the fact-level scores from KG_p, but such an approach cannot be evaluated with the current dataset. Nevertheless, this technique could be promising for real-world applications and future research.

4 Experimental Setup

In this section, we describe the experimental setup, including the used data, implementation details, and aspects we investigated.

4.1 Evaluation Data

We utilized the WikiBio GPT-3 Hallucination Dataset (Manakul et al., 2023) [3] for all experiments. It is designed to evaluate sampling-based hallucination detection methods. It consists of 238 passage-level annotations and 1908 sentence-level annotations. As it contains only test data, tuning parameters on this dataset would not be methodologically correct. While the dataset focuses on sentence-level and passage-level evaluation, our approach goes further by providing more fine-grained insights. To ensure a meaningful comparison with SelfCheckGPT, we evaluated these levels using aggregation approaches (see Section 3.5). Following the protocol established by Manakul et al., 2023, we merged the labels *major-inaccurate* and *minor-inaccurate* into a single *hallucinated* class.

Additionally, we conducted a fact-level evaluation to provide a more comprehensive analysis, with the detailed description and results presented in Appendix B.

4.2 Implementation Details

We employed the Llama-3.1-70B-Instruct model (Grattafiori et al., 2024) as the LLM in our method, hosting it locally using vLLM (Kwon et al., 2023) on a server with 2xNvidia H100 94GB. The sole exception was the hallucination correction experiment, for which we utilized GPT-4o (OpenAI et al., 2024), as detailed in Section 4.5. We set the LLM's temperature to 0.0 for all calls, except during hallucination correction (see Section 4.5), where we set it to 0.5. We implemented the methods and experiments using LangChain (Chase, 2022) and Hugging Face Datasets (Lhoest et al., 2021). All pipeline steps were defined using DVC (Kuprieiev et al., 2025) to facilitate reproducibility.

To determine whether a language model's response was 'yes' or 'no' in Equation 9, we parsed the text into individual words and verified the presence of the words 'yes' or 'no'. If either word was detected, the response was excluded from the averaging process in Equations 8 and 10.

4.3 Evaluation of Sentence-Level and Passage-Level Hallucination Detection

For a fair comparison, we employed the same evaluation protocol as SelfCheckGPT. We reported area under the precision-recall curve (AUC-PR) for

[3] huggingface.co/datasets/potsawee/wiki_bio_gpt3_hallucination (cc-by-sa-3.0 license)

the sentence-level task and Pearson and Spearman correlation coefficients for the passage-level task. We evaluated our method against all variants of SelfCheckGPT, including Prompt, NLI, Unigram (Max), QA, and BERTScore. We reproduced the results of the Prompt variant using the same LLM employed in our study, namely Llama-3.1-70B-Instruct, whereas the original method utilized an unspecified release of GPT-3.5-turbo. We obtained the remaining results of SelfCheckGPT from the original paper. We ensured consistency in the evaluation protocol by reviewing their source code [4].

4.4 Effect of Sample Size on Detection Performance

We investigated the impact of varying the number of samples on the detection performance of our method. Specifically, we evaluated the performance at both the sentence and passage levels by changing the number of samples from 1 to 20. We compared our method with the SelfCheckGPT (Prompt) approach.

4.5 Role of Fact-Level Detection in Hallucination Correction

One potential application of hallucination detection methods is their use in correcting hallucinated responses. In this experiment, we investigate the effectiveness of our fact-level detection approach in enhancing hallucination correction and compare the results with those obtained using sentence-level detection and a baseline method. Each of the three tested approaches uses different input for the LLM: **(1) Baseline**: the original prompt and the generated response (Prompt 7). **(2) Sentence-level**: the original prompt, the generated response, and a list of hallucinated sentences (Prompt 8). **(3) Fact-level**: the original prompt, the generated response, and a list of hallucinated facts (Prompt 9).

The original prompt is the one used during the creation of the dataset: *"This is a Wikipedia passage about {concept_name}:"*. We instructed the LLM to return a list of sentences, allowing it to correct each sentence or leave it unchanged if no hallucinations were detected. We obtained the lists of incorrect sentences/facts using the best variants of models (see Section 5.1) with thresholds that achieved the highest F1-scores on the dataset (0.3 for FactSelfCheck and 0.75 for SelfCheckGPT).

Subsequently, we evaluated the factuality of the corrected responses using the LLM-as-judge approach (Zheng et al., 2023) using Prompt 10. For each corrected sentence, we provided the external biography from Wikipedia. We instructed LLM-as-judge to return 'yes' if the source supported the sentence, 'no' if it was not, or 'refused' if the LLM declined to correct the sentence (e.g., due to insufficient knowledge). We then categorized the responses into three labels: 'factual', 'non-factual', 'refused'.

As mentioned in Section 4.2, we utilized GPT-4o for this experiment instead of Llama-3.1-70B-Instruct (used in detection). This choice was motivated by the challenging nature of the correction task – the model needs to correct hallucinations using only its internal knowledge, without access to external references. While the model knows hallucinated parts, it must rely on its knowledge to determine the correct information.

While the described correction method is not our main contribution, we used it to study the potential benefits of fact-level detection. Although the correction method employed here may not be the most sophisticated, the key takeaway is the observed difference in performance.

5 Results

This section presents the results of the experiments described in the previous section. The results of the fact-level evaluation are given in Appendix B.

5.1 Evaluation of Sentence-Level and Passage-Level Hallucination Detection

Tables 1 and 2 present a comparative analysis of our method against SelfCheckGPT (SCGPT). In the sentence-level evaluation, FactSelfCheck-Text utilizing max as an aggregation function achieves an AUC-PR score of 92.45. It demonstrates that our approach is comparable in performance to the leading SCGPT variants – Prompt (93.60) and NLI (92.50). Notably, while our method operates at a more granular level, it maintains competitive performance with a marginal decrease of 1.2% compared to the best SCGPT. Figure 3 illustrates the precision-recall trade-off across various detection thresholds.

In the passage-level evaluation, our method achieves Pearson and Spearman correlation coefficients of 79.69 and 76.68, respectively. Although these results are lower than SCGPT (Prompt) (83.12 and 78.65), they demonstrate the effec-

[4] github.com/potsawee/selfcheckgpt

Method	Agg.	AUC-PR
Sentence-level methods		
SCGPT (Prompt)	-	93.60
SCGPT (NLI)	-	92.50
SCGPT (Unigram-max)	-	85.63
SCGPT (QA)	-	84.26
SCGPT (BERTScore)	-	81.96
RandomSentence	-	72.96
Fact-level methods (ours)		
FSC-Text	max	92.45
FSC-KG (LLM-based)	max	91.82
FSC-Text	mean	91.01
FSC-KG (LLM-based)	mean	90.24
FSC-KG (Freq.-based)	max	88.48
FSC-KG (Freq.-based)	mean	88.25
RandomFact	mean	74.22

Table 1: Results on the sentence-level hallucination detection task. Comparison of sentence-level and fact-level methods based on AUC-PR scores. SCGPT stands for SelfCheckGPT, FSC represents FactSelfCheck, and Agg denotes the aggregation method used for calculating sentence-level scores.

Method	Agg.	Pear.	Spear.
Sentence-level methods			
SCGPT (Prompt)	-	83.12	78.65
SCGPT (NLI)	-	74.14	73.78
SCGPT (Unigram-max)	-	64.71	64.91
SCGPT (QA)	-	61.07	59.29
SCGPT (BERTScore)	-	58.18	55.90
Fact-level methods (ours)			
FSC-Text	max	79.69	76.68
FSC-KG (LLM-based)	max	77.30	75.37
FSC-Text	mean	77.03	75.89
FSC-KG (LLM-based)	mean	73.92	72.28
FSC-KG (Freq.-based)	mean	71.50	70.45
FSC-KG (Freq.-based)	max	64.05	66.09

Table 2: Results on the passage-level hallucination detection task. Comparison of sentence-level and fact-level methods based on Pearson and Spearman correlation scores. Results are sorted by the Pearson correlation score. SCGPT stands for SelfCheckGPT, FSC represents FactSelfCheck, and Agg denotes the aggregation method used for calculating sentence-level scores. As mentioned in Section 4, passage-level scores were calculated using the mean of sentence-level scores.

tive scalability of the fact-level detection to the sentence-level and passage-level detections.

Among the FactSelfCheck (FSC) variants, FSC-Text, which performs the direct comparison between facts and samples, consistently outperforms FSC-KG, which relies on knowledge graph comparisons. FSC-Text offers computational advantages by eliminating the knowledge graph extraction step. However, FSC-KG may be more computationally efficient for processing longer samples with a lower fact density because of reduced token usage. Both variants perform better than the Frequency-based FSC, which employs a simple fact occurrence counting mechanism. This performance differential highlights the benefits of incorporating semantic matching and knowledge graph reasoning in hallucination detection. Nevertheless, the frequency-based method's computational efficiency may make it suitable for applications where its reduced accuracy is acceptable. Regarding aggregation functions, max consistently yields better results than $mean$, as it is more sensitive to the presence of hallucinated facts in the text.

An interesting side observation is that our reproduced SCGPT on Llama-3.1-70B-Instruct marginally surpassed the original implementation using GPT-3.5-turbo (93.60 vs 93.42 [5]). Furthermore, for passage-level detection, Llama achieved a better Pearson correlation coefficient than the GPT-3.5-turbo (83.12 vs. 78.32 [5]), while the Spearman correlation coefficient remained at a similar level (78.65 vs. 78.30 [5]). This indicates that contemporary open-source models can exceed the performance of previous closed models.

5.2 Effect of Sample Size on Detection Performance

Figures 2 and 4 illustrate the relationship between the number of samples and detection performance for both FactSelfCheck variants and SelfCheckGPT (Prompt). The results show that FactSelfCheck exhibits similar behavior to SelfCheckGPT regarding sample requirements. The performance improves dramatically with up to 5 samples, after which the improvement curve flattens. While additional samples continue to yield benefits, these improvements become incremental, with modest gains observed up to 20 samples.

5.3 Role of Fact-Level Detection in Hallucination Correction

Table 3 presents the results of our hallucination correction experiment. The fact-level approach

[5]The scores on GPT-3.5-turbo were obtained from the original paper (Manakul et al., 2023).

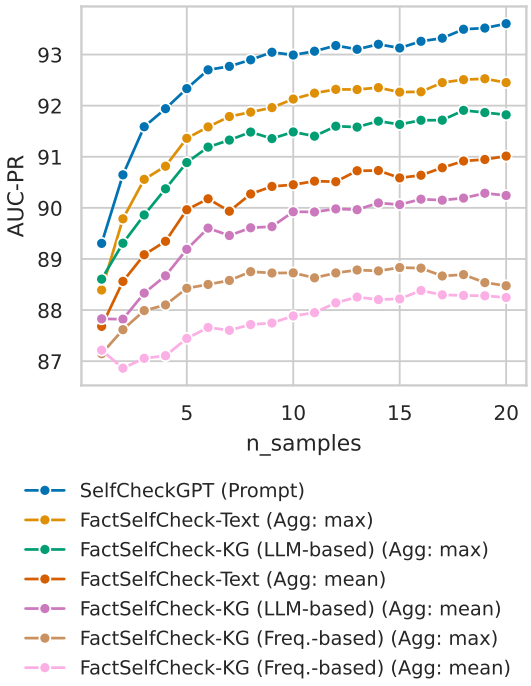

Figure 2: Impact of the number of samples on sentence-level detection task.

Level	Factual ↑	Non-Factual ↓	Refused
baseline	0.23	0.74	0.04
sentence	0.25 (+8%)	0.71 (-4%)	0.05 (+30%)
fact	0.31 (+35%)	0.64 (-12%)	0.05 (+30%)

Table 3: Effectiveness of hallucination correction by providing detected hallucinations at sentence-level, fact-level, and a baseline (without providing any hallucinations). The table presents the proportions of factual, non-factual sentences, and refused corrections. Percentages in parentheses indicate the relative change compared to the baseline. Arrows ↑ and ↓ denote whether a higher or lower value is better.

shows substantial improvements over the baseline and sentence-level methods. We observed a 35% increase in factual content and 12% reduction in non-factual content compared to the baseline. In contrast, the sentence-level detection achieves only modest improvements of 8% and 4%, respectively, indicating that pointing out hallucinations at the fact level enables more effective corrections and underscoring the importance of our study and contributions.

The overall rate of refusals remains low, increasing only marginally from 0.04 in the baseline to 0.05 with fact-level and sentence-level detection. We hypothesize that the model becomes more cautious with provided information about hallucinations and more likely to know its limitations.

6 Conclusion

Our study introduces FactSelfCheck, a novel approach to hallucination detection that operates at the fact level rather than the sentence level or passage level. The evaluation results demonstrate that this granular approach achieves competitive performance with methods like SelfCheckGPT, with the AUC-PR score of 92.45 on sentence-level detection compared to SelfCheckGPT (Prompt)'s 93.60, while offering more detailed insights into the nature of hallucinations.

Our fact-level approach improves hallucination correction. Providing the correction system with a list of incorrect facts rather than incorrect sentences allows for more effective corrections. Compared to baseline, we achieved 35% increase in factual content and 12% reduction in non-factual content. In contrast, the sentence-level approach achieved a 8% increase and 4% reduction, respectively. It underscores the importance of fine-grained hallucination detection in enhancing the reliability and trustworthiness of systems that use LLMs, particularly in applications requiring high factual accuracy.

Limitations

Our study faces two primary limitations. First, we are constrained by the availability of suitable datasets. Currently, the WikiBio GPT-3 Hallucination Dataset, to the best of our knowledge, is the only dataset designed for evaluating sampling-based methods, and it only provides sentence-level and passage-level annotations. It forced us to evaluate our method through aggregation rather than directly assessing our fact-level detection capabilities. Second, while the granular approach of FactSelfCheck justifies its increased complexity, the multiple LLM-based steps make it more computationally intensive compared to more straightforward methods like SelfCheckGPT.

Several promising directions could address these limitations. An important step would be creating new datasets with fact-level hallucination annotations, enabling direct evaluation of our method's core capabilities. Additionally, we see significant potential for improving computational efficiency. Our current prompt engineering was largely empirical and not optimized for token usage. Future work could focus on reducing prompt lengths and merg-

ing steps, such as simultaneously assessing support for multiple facts or merging the KG extraction steps. Finally, incorporating external knowledge graphs could enhance fact verification while potentially reducing our reliance on stochastic sampling.

Ethical Considerations

Like all machine learning methods, FactSelfCheck can produce false positives and false negatives. Therefore, it should not completely replace human verification of factual correctness in LLM responses. Additionally, since the fact-level annotations (Appendix B) are generated by an LLM, there remains a risk of fact mislabeling.

Acknowledgments

This work was funded by the European Union under the Horizon Europe grant OMINO – Overcoming Multilevel INformation Overload (grant number 101086321, http://ominoproject.eu/). Views and opinions expressed are those of the authors alone and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the European Research Executive Agency can be held responsible for them. It was also co-financed with funds from the Polish Ministry of Education and Science under the programme entitled International Co-Financed Projects, grant no. 573977. This work was co-funded by the National Science Centre, Poland under CHIST-ERA Open & Reusable Research Data & Software (grant number 2022/04/Y/ST6/00183).

References

Amos Azaria and Tom Mitchell. 2023. 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.

Harrison Chase. 2022. Langchain.

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. *Preprint*, arXiv:2402.03744.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. Factool: Factuality detection in generative ai – a tool augmented framework for multi-task and multi-domain scenarios. *Preprint*, arXiv:2307.13528.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James R. Glass. 2024. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1419–1436, Miami, Florida, USA. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Xinyue Fang, Zhen Huang, Zhiliang Tian, Minghui Fang, Ziyi Pan, Quntian Fang, Zhihua Wen, Hengyue Pan, and Dongsheng Li. 2024. Zero-resource hallucination detection for text generation via graph-based contextual knowledge triples modeling. *arXiv preprint arXiv:2409.11283*.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630. Publisher: Nature Publishing Group.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang,

Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj

Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. In *IEEE Data Eng. Bull.*, volume 40, pages 52–74.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *Preprint*, arXiv:2406.15927.

Ruslan Kuprieiev, skshetry, Peter Rowlands, Dmitry Petrov, Paweł Redzyński, Casper da Costa-Luis, David de la Iglesia Castro, Alexander Schepanovski, Ivan Shcheklein, Gao, Batuhan Taskaya, Jorge Orpinel, Fábio Santos, Daniele, Ronan Lamy, Aman Sharma, Zhanibek Kaimuldenov, Dani Hodovic, Nikita Kodenko, Andrew Grigorev, Earl, Nabanita Dash, George Vyshnya, Dave Berenbaum, maykulkarni, Max Hora, Vera, and Sanidhya Mangal. 2025. Dvc: Data version control - git for data & models.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Minhyeok Lee. 2023. A mathematical investigation of hallucination and creativity in gpt models. *Mathematics*, 11(10).

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow,

Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

Mohamed Rashad, Ahmed Zahran, Abanoub Amin, Amr Abdelaal, and Mohamed Altantawy. 2024. FactAlign: Fact-level hallucination detection and classification through knowledge graph alignment. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*, pages 79–84, Mexico City, Mexico. Association for Computational Linguistics.

Malik Sallam. 2023. Chatgpt utility in healthcare education, research, and practice: Systematic review on the promising perspectives and valid concerns. *Healthcare*, 11(6).

Hannah Sansford, Nicholas Richardson, Hermina Petric Maretic, and Juba Nait Saada. 2024. Grapheval: A knowledge-graph based llm hallucination evaluation framework. *Preprint*, arXiv:2407.10793.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. LLM-check: Investigating detection of hallucinations in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference*

on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *Preprint*, arXiv:2401.11817.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

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. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

A Additional Figures for Results

Figure 3 depicts the precision-recall curve for sentence-level detection (see Section 5.1). Figure 4 shows the impact of the number of samples on the passage-level detection task (see Section 5.2).

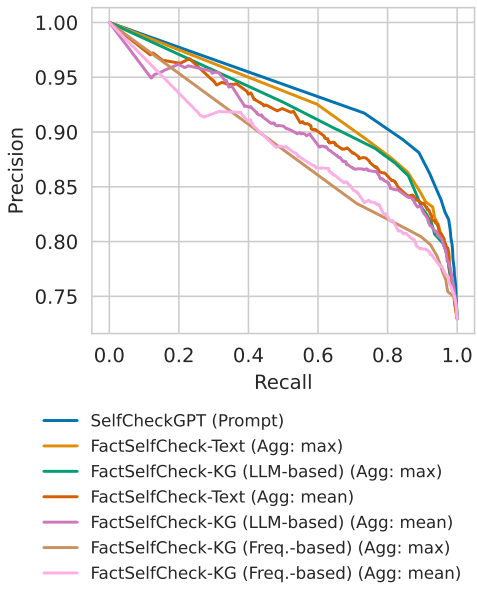

Figure 3: Precision-recall curve for the sentence-level detection task.

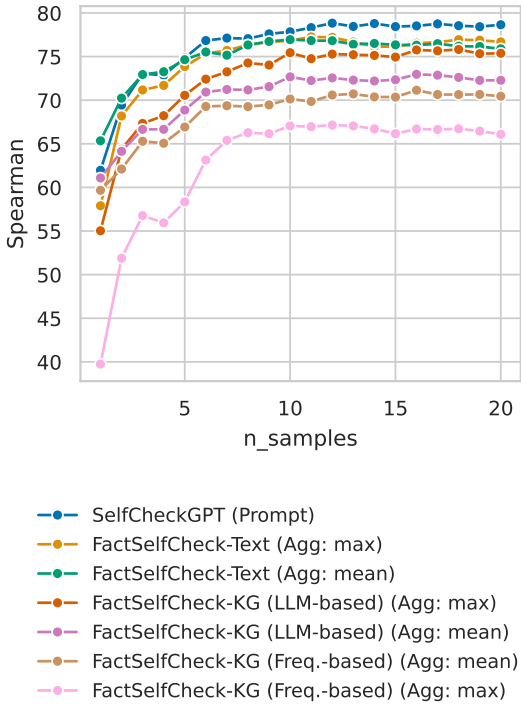

Figure 4: Impact of the number of samples on passage-level detection task.

B Fact-Level Detection Evaluation

While the evaluation in Section 5.1 focused on the sentence-level and passage-level detection due to dataset constraints, we present a complementary evaluation at the fact level here. Since this evaluation is not based on human annotations, we have placed it in the appendix.

Evaluation Data To enable fact-level evaluation, we annotated facts from the extracted response knowledge graph (KG_p) using the LLM-as-judge approach (Zheng et al., 2023). This approach annotated whether each fact is supported by the external Wikipedia biography (Prompt 11). As a result, we obtained 5488 binary annotated facts.

Experimental Setup We evaluated all variants of FactSelfCheck using the fact-level score $\mathcal{H}_{\text{fact}}$. For comparison with SelfCheckGPT, which provides only sentence-level granularity, we derived fact-level scores by averaging the sentence-level scores across all sentences containing each fact.

Results Table 4 presents the comparative results. FactSelfCheck-Text demonstrates superior performance with an AUC-PR score of 93.41, followed by FactSelfCheck-KG (LLM-based) at 92.25. The frequency-based method achieves 87.99, while SelfCheckGPT (Prompt) scores 86.18. These results demonstrate the effectiveness of our method in detecting hallucinations at the fact level. Furthermore, the lower performance of averaging the sentence-level scores highlights the importance of designing fact-level methods and validating our approach.

Method	AUC-PR
FactSelfCheck-Text	93.41
FactSelfCheck-KG (LLM-based)	92.25
FactSelfCheck-KG (Freq.-based)	87.99
SelfCheckGPT (Prompt)	86.18
RandomFact	65.79

Table 4: Results on the fact-level hallucination detection task. Comparison of sentence-level and fact-level methods based on AUC-PR scores.

C Passage-level Score: Size of Knowledge Graphs Extracted from Samples

As a side study, we examined the effectiveness of scoring hallucinations at the passage level based

on the size of knowledge graphs extracted from samples.

Score Definition We propose using the average size of restricted knowledge graphs extracted from samples as a passage-level score. Restriction involves keeping only facts where all entities and relations are present in $\mathcal{E}_S$ or $\mathcal{R}_S$, respectively. This results in KG'_s defined as:

$$KG'_s = \{(h, r, t) \in KG_s \mid h \in \mathcal{E}_S \wedge r \in \mathcal{R}_S \wedge t \in \mathcal{E}_S\} \quad (13)$$

The passage-level score is then calculated as:

$$\mathcal{H}_{\text{passage}}(p) = \frac{1}{|S|} \sum_{s \in S} -|KG'_s| \quad (14)$$

The rationale behind this score is that the size of KG'_s reflects how well the samples align with the original response p. A smaller KG'_s indicates that the sample contains facts than those in the original response, suggesting potential hallucinations. In distinction from the previously defined scores, this metric operates outside the standard $[0, 1]$ range.

Results The method achieved Pearson and Spearman correlation scores of 72.35 and 71.18, respectively. When compared to the main results in Table 2, this approach shows lower correlation scores than LLM-based methods but outperforms the frequency-based method. It is important to note that while it is the passage-level score, frequency-based methods operate at the fact level. Nevertheless, this approach offers the most computationally efficient way to calculate passage-level scores among tested methods utilizing knowledge graphs, as it does not require any LLM calls during factual consistency assessment.

D Case Study of FactSelfCheck vs SelfCheckGPT

Table 5 and Figure 5 present a comparative case study of predictions made by FactSelfCheck and SelfCheckGPT. The table contains an external Wikipedia biography, sentences from the response, facts extracted from the response, and the predictions of both methods. This comparison demonstrates that fact-level detection provides more detailed information about the factuality of the response. We observe that LLM did not hallucinate all facts, as some are consistent with the external Wikipedia biography. However, when using sentence-level detection, we cannot distinguish between correct and hallucinated facts – all sentences are predicted as hallucinated.

E SelfCheckGPT with Enhanced Prompt

To ensure a fair comparison between FactSelfCheck and SelfCheckGPT, we conducted an additional experiment using an enhanced prompt for SelfCheckGPT. The original SelfCheckGPT prompt is relatively simple, while our FactSelfCheck prompts are more elaborate, directly allowing reasoning and inference of new facts, and providing examples. These characteristics could potentially increase the performance of methods. We designed an alternative prompt for SelfCheckGPT, presented in Prompt 12, that incorporates these features, making it similar to our FactSelfCheck prompts.

For sentence-level detection, the enhanced prompt for SelfCheckGPT achieved an AUC-PR score of 93.38, slightly lower than the original prompt's 93.60. This minimal difference indicates that SelfCheckGPT's performance is not significantly affected by prompt design in our experimental setting. In fact, the enhanced prompt even lowered the performance rather than increasing it, despite its more sophisticated design. For passage-level detection, we observed similar results, with the enhanced prompt achieving Pearson and Spearman correlation scores of 82.36 and 76.71, respectively, compared to the original prompt's 83.12 and 78.65.

Due to these findings, we chose to use the original prompt for SelfCheckGPT in our main experiments. These consistent results confirm that our comparison between methods is fair, as the enhanced prompt did not improve the performance of SelfCheckGPT.

F FactSelfCheck Prompts

In this section, we present the prompts used in FactSelfCheck. These include prompts for the entity extraction (Prompt 1), the relation extraction (Prompt 2), and the knowledge graph extraction from a sentence (Prompt 3) or a sample (Prompt 4). Additionally, we provide prompts for assessing factual consistency in FactSelfCheck-KG (Prompt 5) and FactSelfCheck-Text (Prompt 6).

G Correction Prompts

The following prompts are instructions for correcting hallucinated responses generated by the LLM. The prompts are for the baseline (Prompt 7), sentence-level (Prompt 8), and fact-level (Prompt 9) correction. The prompts take several inputs: *{generated_sentences}* containing the response as a numbered list of sentences, *{incorrect_sentences}* listing the hallucinated sentences with their corresponding numbers from the generated text, and *{incorrect_facts}* providing a formatted list of hallucinated facts organized by sentence. The *{format}* field specifies the output format as an empty numbered list with a length equal to the number of sentences in the *{generated_sentences}* field.

External Wikipedia Bio	
Kenan Hasagić (born 1 February 1980) is a Bosnian football goalkeeper who plays for Balıkesirspor. His football career began in his hometown with FK Rudar. At the age of 16, he made his debut in a first division match. He was the most promising goalkeeper in Bosnia and Herzegovina; he played for youth selections and was later transferred to Austrian side Vorwärts Steyr. After that, he was a member of Altay SK in Turkey but didn't see much first team football. He went back to Bosnia and played for Bosna Visoko. In 2003, he signed a contract with FK Željezničar. Here he found good form and even became first choice goalkeeper for the Bosnian national team. In the 2004–05 season, he moved to Turkey once again where he signed for Turkish Süper Lig side Gaziantepspor. He made his debut for the national team on 12 February 2003 in a game between Wales and Bosnia and Herzegovina which ended in a 2–2 draw.	
Sentence 1 (SelfCheckGPT: 1.0)	
Kenan Hasagić (born 28 April 1988) is a Bosnian professional footballer who plays as a midfielder for Bosnian Premier League club Željezničar.	
Fact	**FactSelfCheck**
('Kenan Hasagić', 'DATE OF BIRTH', 'April 28, 1988')	1.00
('Kenan Hasagić', 'NATIONALITY', 'Bosnian')	0.00
('Kenan Hasagić', 'OCCUPATION', 'professional footballer')	0.20
('Kenan Hasagić', 'POSITION PLAYED', 'midfielder')	0.55
('Kenan Hasagić', 'LEAGUE', 'Bosnian Premier League')	0.45
('Kenan Hasagić', 'CURRENT CLUB', 'Željezničar')	0.85
Sentence 2 (SelfCheckGPT: 1.0)	
Hasagić started his career at his hometown club Željezničar, where he made his professional debut in 2006.	
Fact	**FactSelfCheck**
('Kenan Hasagić', 'CURRENT CLUB', 'Željezničar')	0.85
('Kenan Hasagić', 'PROFESSIONAL DEBUT', '2006')	0.90
('Kenan Hasagić', 'CURRENT CLUB', 'Željezničar')	0.85
Sentence 3 (SelfCheckGPT: 1.0)	
He has since gone on to make over 200 appearances for the club, winning the Bosnian Premier League title in 2008 and the Bosnian Cup in 2009.	
Fact	**FactSelfCheck**
('Kenan Hasagić', 'TEAM APPEARANCES', 'Željezničar')	0.60
('Željezničar', 'CHAMPIONSHIP WON', 'Bosnian Premier League title')	0.90
('Željezničar', 'YEAR OF CHAMPIONSHIP', '2008')	1.00
('Željezničar', 'CHAMPIONSHIP WON', 'Bosnian Cup')	0.95
('Željezničar', 'YEAR OF CHAMPIONSHIP', '2009')	1.00
Sentence 4 (SelfCheckGPT: 0.9)	
He has also represented Bosnia and Herzegovina at international level, making his debut in 2011.	
Fact	**FactSelfCheck**
('Kenan Hasagić', 'COUNTRY REPRESENTED', 'Bosnia and Herzegovina')	0.05
('Kenan Hasagić', 'INTERNATIONAL DEBUT', '2011')	0.85

Table 5: Comparison of fact-level FactSelfCheck with sentence-level SelfCheckGPT. An external Wikipedia biography is provided to analyse the correctness of the methods. The red value indicates hallucinations, and the green value indicates factual correctness. The facts were classified using a threshold of 0.4 utilizing FactSelfCheck, and the sentences were classified using a threshold of 0.75 with SelfCheckGPT. These thresholds achieved the highest F1-scores in fact-level (Appendix B) and sentence-level (Section 5.1) evaluation, respectively.

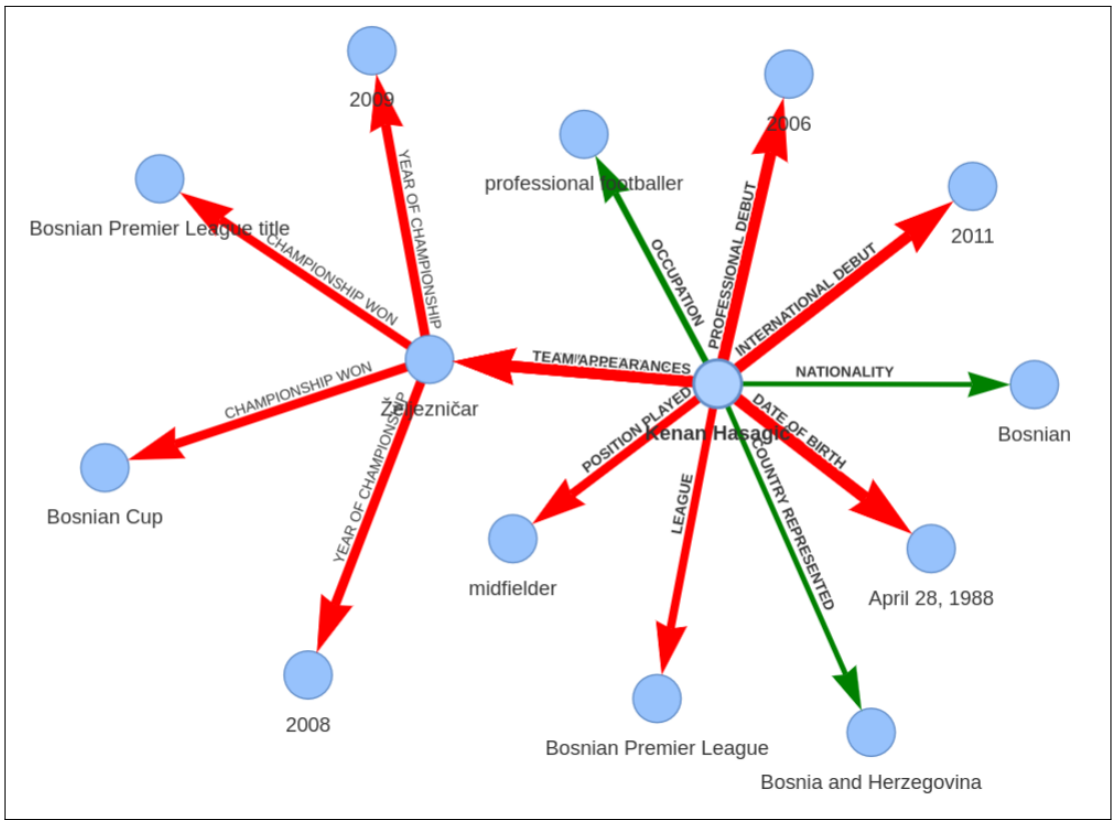

Figure 5: Example of a knowledge graph extracted from a response. Edge width represents the hallucination score for each fact, with red edges indicating hallucinated facts and green edges indicating correct facts. Facts were classified using a threshold of 0.4, which achieved the highest F1-score in the fact-level evaluation (Appendix B).

```
================================ System Message ================================

You are a top-tier algorithm designed for extracting entities in structured formats.
These entities will be used to build a knowledge graph.
Your task is to identify the entities requested with the user prompt from a given text.
You must generate the output in a JSON format containing a list with JSON objects.
Each object should have the key: "entities".
The "entities" key must contain a list of strings, where each string is an extracted entity.
Attempt to extract as many entities as you can.
It is very important to cover all the entities in the text!
Entities can be in many types, such as people, organizations, locations, events, positions, dates, roles, titles,
      objects, and more.
If an entity, such as "John Doe", is mentioned multiple times in the text but is referred to by different names
     or pronouns (e.g., "Joe", "he"), always use the most complete identifier for that entity.
IMPORTANT NOTES:
- Don't add any explanation and text.
- It is very important to cover all the entities in the text!
- Remember to cover dates.
- Remember to cover events eg. publication of book, employment, graduation, marriage.
- All numbers, dates has to be in string format. eg. "2023", "1980s".

================================ Human Message =================================

Below are a number of examples of text and their extracted entities.
[{'text': "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman
     who served as the 45th president of the United States from 2017 to 2021. Having won the 2024 presidential
     election as the nominee of the Republican Party, he is the president-elect and will be inaugurated as the 47
     th president on January 20, 2025. Trump graduated with a bachelor's degree in economics from the University
     of Pennsylvania in 1968.", 'entities': ['Donald John Trump', 'American politician', 'media personality', '
     businessman', '45th president', 'United States', '2017', '2021', '2024 presidential election', 'Republican
     Party', 'president-elect', '47th president', 'January 20, 2025', 'University of Pennsylvania', 'graduation',
      'bachelor', 'degree', 'economics', '1968']}]
For the following text, extract entities as in the provided example.
{format_instructions}
Text: {input}
```

Prompt 1: Instructions for the LLM to extract entities $\mathcal{E}_p$ from the response (*{input}*). The *{format_instructions}* field contains formatting instructions for JSON output, constructed by the LangChain library based on the expected output schema.

```
================================ System Message ================================

You are a top-tier algorithm designed for extracting relationship types in structured formats.
These relationship types will be used to build a knowledge graph in the future.
Your task is to identify the relationship types requested with the user prompt from a given text and list of
     entities.
You must generate the output in a JSON format containing a list with JSON objects.
Each object should have the key: "relationship_types".
The "relationship_types" key must contain a list of strings, where each string is an extracted relationship type.
Attempt to extract as many relationship types as you can.
It is very important to cover all the relationship types in the text!
Ensure the relationships accurately describe the interaction or connection between entities.
Relationship types should be simple and concise, but also specific and informative, to cover all the
     relationships between entities in the text.
If it is possible, use the WikiData relationship types. If not, use the most appropriate relationship type.
IMPORTANT NOTES:
- Don't add any explanation and text.
- It is very important to cover all the relationship types in the text!
- Omit long and complex relationship types.
- Use double quotes, not single quotes for json format.


================================ Human Message =================================

Below are a number of examples of text and their extracted relationship types.
[{'text': "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman
     who served as the 45th president of the United States from 2017 to 2021. Having won the 2024 presidential
     election as the nominee of the Republican Party, he is the president-elect and will be inaugurated as the 47
     th president on January 20, 2025. Trump graduated with a bachelor's degree in economics from the University
     of Pennsylvania in 1968.", 'entities': ['Donald John Trump', 'American politician', 'media personality', '
     businessman', '45th president', 'United States', '2017', '2021', '2024 presidential election', 'Republican
     Party', 'president-elect', '47th president', 'January 20, 2025', 'University of Pennsylvania', 'graduation',
      'bachelor', 'degree', 'economics', '1968'], 'relationship_types': ['DATE OF BIRTH', 'NATIONALITY', '
     OCCUPATION', 'POSITION HELD', 'TERM OF OFFICE', 'MEMBER OF', 'ELECTED AS', 'NOMINEE OF', 'FUTURE POSITION',
     'DATE OF START', 'DATE OF END', 'EDUCATED AT', 'DEGREE IN', 'DEGREE FROM', 'DEGREE TYPE', 'DATE OF
     GRADUATION']}]
For the following text and entities, extract relationship types as in the provided example.
{format_instructions}
Text: {input}
Entities: {entities}
```

Prompt 2: Instructions for the LLM to extract relations $\mathcal{R}_p$ from the response (*{input}*). The *{entities}* field contains entities extracted using Prompt 1. The *{format_instructions}* field specifies JSON output formatting requirements constructed by the LangChain library based on the expected schema.

```
================================ System Message ================================

You are a top-tier algorithm designed for extracting information in CSV format to build a knowledge graph.
In input, you will receive: sentence, full text, list of entities, and list of relationships.
Your task is to identify facts (triples) consisting of a subject, relation, and object from a given sentence.
Do not generate relationships that are defined in the full text but not in the sentence.
You must generate the output in CSV format with semicolon ";" as the separator, and lines separated by "\n".
The first line should be the header: subject;relation;object
Each subsequent line should contain one triple with the following columns:
- subject: the text of the extracted entity which is the subject of the relation
- relation: the type of relation between the subject and object
- object: the text of the extracted entity which is the object of the relation
The extracted entities must be from the list provided by the user.
The relations must be from the list of allowed relation types provided by the user.
Attempt to extract as many facts / triples as you can.
Entities can be in many types, such as people, organizations, locations, dates, roles, titles, objects, and more.
Maintain Entity Consistency: When extracting entities, it's vital to ensure consistency.
If an entity, such as "John Doe", is mentioned multiple times in the text but is referred to by different names
     or pronouns (e.g., "Joe", "he"), always use the most complete identifier for that entity.
The knowledge graph should be coherent and easily understandable, so maintaining consistency in entity references
      is crucial.
The knowledge graph should be complete, so it should cover all the facts in the sentence.
IMPORTANT NOTES:
- Don't add any explanation and text.
- It is very important to cover all triples (facts) in the sentence!
- Full text is only for understanding the context. Do not add any triples that are not in the sentence.
- Output must be in CSV format with semicolon (;) separators. Lines should be separated by "\n".
- Do not include quotes around the values unless they contain semicolons


================================ Human Message =================================

For the following sentence, full text, entities, and relation types, extract facts / triples as in the provided
     sentence.
Your response should be a list of semicolon separated values with header, eg:
```
subject;relation;object
Donald John Trump;DEGREE FROM;University of Pennsylvania
Donald John Trump;DEGREE IN;economics
```
Below are a number of examples of text and their extracted facts.
[{'full_text': "Donald John Trump (born June 14, 1946) is an American politician, media personality, and
     businessman who served as the 45th president of the United States from 2017 to 2021. Having won the 2024
     presidential election as the nominee of the Republican Party, he is the president-elect and will be
     inaugurated as the 47th president on January 20, 2025. Trump graduated with a bachelor's degree in economics
      from the University of Pennsylvania in 1968.", 'sentence': "Trump graduated with a bachelor's degree in
     economics from the University of Pennsylvania in 1968.", 'allowed entities': ['Donald John Trump', 'American
      politician', 'media personality', 'businessman', '45th president', 'United States', '2017', '2021', '2024
     presidential election', 'Republican Party', 'president-elect', '47th president', 'January 20, 2025', '
     University of Pennsylvania', 'graduation', 'bachelor', 'degree', 'economics', '1968'], 'allowed relation
     types': ['DATE OF BIRTH', 'NATIONALITY', 'OCCUPATION', 'POSITION HELD', 'TERM OF OFFICE', 'MEMBER OF', '
     ELECTED AS', 'NOMINEE OF', 'FUTURE POSITION', 'DATE OF START', 'DATE OF END', 'EDUCATED AT', 'DEGREE IN', '
     DEGREE FROM', 'DEGREE TYPE', 'DATE OF GRADUATION'], 'triples': 'subject;relation;object\nDonald John Trump;
     DEGREE FROM;University of Pennsylvania\nDonald John Trump;DEGREE IN;economics\nDonald John Trump;DEGREE TYPE;
     bachelor\nDonald John Trump;DATE OF GRADUATION;1968\n'}]
Use the following entities, don't use other entity that is not defined below:
# ALLOWED ENTITIES:
{allowed_nodes}
Use the following relation types, don't use other relation that is not defined below:
# ALLOWED RELATION TYPES:
{allowed_relationships}
Full text: {input_text}
Sentence: {input_sentence}
```

Prompt 3: Instructions for the LLM to extract knowledge graph KG_u from a sentence (*{input_sentence}*). We also provide a full response (*{input_text}*) to give LLM a context. The *{allowed_nodes}* field contains the list of allowed entities. The *{allowed_relationships}* field contains the list of allowed relations. The LLM is expected to return a list of triples in the CSV format.

```
================================ System Message ================================

You are a top-tier algorithm designed for extracting information in CSV format to build a knowledge graph.
Your task is to identify facts (triples) consisting of a subject, relation, and object from a given text.
You must generate the output in CSV format with semicolon ";" as the separator, and lines separated by "\n".
The first line should be the header: subject;relation;object
Each subsequent line should contain one triple with the following columns:
- subject: the text of the extracted entity which is the subject of the relation
- relation: the type of relation between the subject and object
- object: the text of the extracted entity which is the object of the relation
The extracted entities must be from the list provided by the user.
The relations must be from the list of allowed relation types provided by the user.
Attempt to extract as many facts / triples as you can.
Entities can be in many types, such as people, organizations, locations, dates, roles, titles, objects, and more.
Maintain Entity Consistency: When extracting entities, it's vital to ensure consistency.
If an entity, such as "John Doe", is mentioned multiple times in the text but is referred to by different names
     or pronouns (e.g., "Joe", "he"), always use the most complete identifier for that entity.
The knowledge graph should be coherent and easily understandable, so maintaining consistency in entity references
      is crucial.
The knowledge graph should be complete, so it should cover all the facts in the text.
IMPORTANT NOTES:
- Don't add any explanation and text.
- It is very important to cover all triples (facts) in the text!
- Output must be in CSV format with semicolon (;) separators. Lines should be separated by "\n".
- Do not include quotes around the values unless they contain semicolons


================================ Human Message =================================

For the following text, allowed entities and allowed relation types, extract facts / triples as in the provided
     example.
Your response should be a list of semicolon separated values with header, eg:
```
subject;relation;object
Donald John Trump;DATE OF BIRTH;June 14, 1946
Donald John Trump;NATIONALITY;American politician
```
Below are a number of examples of text and their extracted facts.
[{'text': "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman
     who served as the 45th president of the United States from 2017 to 2021. Having won the 2024 presidential
     election as the nominee of the Republican Party, he is the president-elect and will be inaugurated as the 47
     th president on January 20, 2025. Trump graduated with a bachelor's degree in economics from the University
     of Pennsylvania in 1968.", 'allowed entities': ['Donald John Trump', 'American politician', 'media
     personality', 'businessman', '45th president', 'United States', '2017', '2021', '2024 presidential election',
      'Republican Party', 'president-elect', '47th president', 'January 20, 2025', 'University of Pennsylvania',
     'graduation', 'bachelor', 'degree', 'economics', '1968'], 'allowed relation types': ['DATE OF BIRTH', '
     NATIONALITY', 'OCCUPATION', 'POSITION HELD', 'TERM OF OFFICE', 'MEMBER OF', 'ELECTED AS', 'NOMINEE OF', '
     FUTURE POSITION', 'DATE OF START', 'DATE OF END', 'EDUCATED AT', 'DEGREE IN', 'DEGREE FROM', 'DEGREE TYPE',
     'DATE OF GRADUATION'], 'triples': 'subject;relation;object\nDonald John Trump;DATE OF BIRTH;June 14, 1946\
     nDonald John Trump;NATIONALITY;American politician\nDonald John Trump;OCCUPATION;media personality\nDonald
     John Trump;OCCUPATION;businessman\nDonald John Trump;POSITION HELD;45th president\n45th president;TERM OF
     OFFICE;2017\n45th president;TERM OF OFFICE;2021\nDonald John Trump;ELECTED AS;2024 presidential election\
     n2024 presidential election;NOMINEE OF;Republican Party\nDonald John Trump;FUTURE POSITION;47th president\
     n47th president;DATE OF INAUGURATION;January 20, 2025\n47th president;DATE OF START;January 20, 2025\nDonald
      John Trump;EDUCATED AT;University of Pennsylvania\nDonald John Trump;DEGREE FROM;University of Pennsylvania\
     nDonald John Trump;DEGREE IN;economics\nDonald John Trump;DATE OF GRADUATION;1968\n'}]
Use the following entities, don't use other entity that is not defined below:
# ALLOWED ENTITIES:
{allowed_nodes}
Use the following relation types, don't use other relation that is not defined below:
# ALLOWED RELATION TYPES:
{allowed_relationships}
Text: {input}
```

Prompt 4: Instructions for the LLM to extract a knowledge graph KG_s from a sample (*{input}*). The *{allowed_nodes}* field contains the list of allowed entities. The *{allowed_relationships}* field contains the list of allowed relations. The LLM is expected to return a list of triples in the CSV format.

```
================================ System Message ================================

You are a top-tier algorithm designed for verification whether fact is supported by the provided knowledge graph.
Your task is to answer whether the given fact is supported by the knowledge graph.
Return `yes` if the fact is supported, `no` otherwise.
You can reason over the knowledge graph and the fact to answer.
You can infer the answer from the knowledge graph, e.g. by inferring new facts from the knowledge graph.
Do not add new facts other than by inference from the knowledge graph.
Fact is a triple of the form `(subject, predicate, object)`. Knowledge graph is a list of facts.
You must return a single word, either `yes` or `no`.
You must not return any other text or explanation.


================================ Human Message =================================

Below are a number of examples of facts, knowledge graph and their verification results.
- Fact: (Donald Trump, president, United States)
  Knowledge graph: [(Donald Trump, president, United States), (Donald Trump, born, 1946), (Donald Trump, graduated
       , University of Pennsylvania)]
  Is the fact supported by the knowledge graph?: yes
- Fact: (Joe Biden, born, 1942)
  Knowledge graph: [(Joe Biden, proffesion, politician), (Joe Biden, president, United States), (Joe Biden,
       graduated, University of Delaware)]
  Is the fact supported by the knowledge graph?: no
- Fact: (Michael Jordan, played, basketball)
  Knowledge graph: [(Michael Jordan, CEO, PepsiCo), (Michael Jordan, born, 1936), (Michael Jordan, graduated, Yale
        University)]
  Is the fact supported by the knowledge graph?: no
- Fact: (Kobe Bryant, played in, NBA)
  Knowledge graph: [(Kobe Bryant, born, 1978), (Kobe Bryant, played in, Los Angeles Lakers)]
  note: Los Angeles Lakers is a part of NBA
  Is the fact supported by the knowledge graph?: yes
- Fact: (Bryant, profession, basketball player)
  Knowledge graph: [(Kobe Bryant, born, 1978), (Kobe Bryant, played in, Los Angeles Lakers)]
  note: In this context, Bryant is Kobe Bryant, and Los Angeles Lakers is a basketball team, so Kobe Bryant is a
       basketball player
  Is the fact supported by the knowledge graph?: yes

===
For the following fact and knowledge graph, verify if the fact is supported by the knowledge graph.
Fact: {input}
Knowledge graph: {knowledge_graph}
Is the fact supported by the knowledge graph?
```

Prompt 5: FactSelfCheck-KG: Instructions for the LLM to assess support of fact (*{input}*) by a knowledge graph (*{knowledge_graph}*). The LLM is expected to return 'yes' if the fact is supported by the knowledge graph, 'no' otherwise.

```
================================ System Message ================================

You are a top-tier algorithm designed for verification whether fact is supported by the provided context.
Your task is to answer whether the given fact is supported by the context.
Return `yes` if the fact is supported, `no` otherwise.
You can reason over the context and the fact to answer.
You can infer the answer from the context, e.g. by inferring new facts from the context.
Do not add new facts other than by inference from the context.
Fact is a triple of the form `(subject, predicate, object)`. Context is a text.
You must return a single word, either `yes` or `no`.
You must not return any other text or explanation.


================================ Human Message =================================

Below are a number of examples of facts, context and their verification results.
- Fact: (Donald Trump, president, United States)
  Context: Donald Trump was president of the United States. Donald Trump was born in 1946 and graduated from the
       University of Pennsylvania.
  Is the fact supported by the context?: yes
- Fact: (Joe Biden, born, 1942)
  Context: Joe Biden is a politician and president of the United States. He graduated from the University of
       Delaware.
  Is the fact supported by the context?: no
- Fact: (Michael Jordan, played, basketball)
  Context: Michael Jordan is the CEO of PepsiCo. He was born in 1936 and graduated from Yale University.
  Is the fact supported by the context?: no
- Fact: (Kobe Bryant, played in, NBA)
  Context: Kobe Bryant was born in 1978 and played for the Los Angeles Lakers.
  note: Los Angeles Lakers is a part of NBA
  Is the fact supported by the context?: yes
- Fact: (Bryant, profession, basketball player)
  Context: Kobe Bryant was born in 1978 and played for the Los Angeles Lakers.
  note: In this context, Bryant is Kobe Bryant, and Los Angeles Lakers is a basketball team, so Kobe Bryant is a
       basketball player
  Is the fact supported by the context?: yes

===
For the following fact and context, verify if the fact is supported by the context.
Fact: {input}
Context: {context}
Is the fact supported by the context?
```

Prompt 6: FactSelfCheck-Text: Instructions for the LLM to assess support of fact (*{input}*) by a sample (*{context}*). The LLM is expected to return 'yes' if the fact is supported by the sample, 'no' otherwise.

````
================================ System Message ================================

You are a top-tier algorithm designed for correcting sentences from the generated text generated by the LLM.
You will be given a original prompt, a generated text in form of a list of sentences.
We don't know whether the given sentences are correct or incorrect.
You need to correct the incorrect sentences to make it more factual and coherent.
If the sentence is correct, you should return the original sentence.
If you don't know how to correct the sentence, you should return the original sentence.
Put the corrected sentences in the following format, each sentence separated by a new line:
```
1. Corrected sentence 1.
2. Corrected sentence 2.
:::
You must return only the corrected or original sentences, not return any other text or explanation.
Important: The number of corrected sentences should be the same as the number of sentences in the generated text.


================================ Human Message =================================

Below are a number of examples of original prompt, list of sentences from the generated text.
- Original prompt: ```Donald Trump is```
  Generated sentences:
  ```
  1. Donald Trump is the president of Canada.
  2. He was born in 1946 and graduated from the University of Pennsylvania.
  ```
  Corrected sentences:
  ```
  1. Donald Trump is the president of the United States.
  2. Donald Trump was born in 1946 and graduated from the University of Pennsylvania.
  ```
- Original prompt: ```Kobe Bryant is```
  Generated sentences:
  ```
  1. Kobe Bryant (February 17, 1963 - January 26, 2020) was an American professional basketball player.
  2. Bryant married Vanessa Laine in 2001.
  ```
  Corrected sentences:
  ```
  1. Kobe Bryant (August 23, 1978 - January 26, 2020) was an American professional basketball player.
  2. Kobe Bryant married Vanessa Laine in 2001.
  ```

===
For the following original prompt, generated text, and incorrect sentences, correct the generated text.
Original prompt: ```{original_prompt}```
Generated sentences:
```
{generated_sentences}
```
Answer in format:
```
{format}
```
Corrected sentences:
````

Prompt 7: Instructions for the LLM to correct a hallucinated response without pointing out hallucinations. For a description of input fields { }, see Appendix G.
````

```
================================ System Message ================================

You are a top-tier algorithm designed for correcting sentences from the generated text generated by the LLM.
You will be given a original prompt, sentences from the generated text, and a list of incorrect sentences from
     the generated text.
Empty list means that no incorrect sentences were found.
You need to correct the sentences to make it more factual and coherent.
If the sentence is correct, you should return the original sentence.
If you don't know how to correct the sentence, you should return the original sentence.
Put the corrected sentences in the following format, each sentence separated by a new line:
```

1. Corrected sentence 1.
2. Corrected sentence 2.
3. Corrected sentence 3.

```
You must return only the corrected or original sentences, not return any other text or explanation.
Important: The number of corrected sentences should be the same as the number of sentences in the generated text.


================================ Human Message =================================

Below are a number of examples of original prompt, sentences from the generated text, and incorrect sentences.
- Original prompt: ```Donald Trump is```
  Generated sentences:
  ```

  1. Donald Trump is the president of Canada.
  2. He was born in 1946 and graduated from the University of Pennsylvania.
  ```
  Incorrect sentences:
  ```

  1. Donald Trump is the president of Canada.
  ```
  Corrected sentences:
  ```

  1. Donald Trump is the president of the United States.
  2. Donald Trump was born in 1946 and graduated from the University of Pennsylvania.
  ```
- Original prompt: ```Kobe Bryant is```
  Generated sentences:
  ```

  1. Kobe Bryant (February 17, 1963 - January 26, 2020) was an American professional basketball player.
  2. Bryant married Vanessa Laine in 2001.
  ```
  Incorrect sentences:
  ```

  1. Kobe Bryant (February 17, 1963 - January 26, 2020) was an American professional basketball player.
  ```
  Corrected sentences:
  ```

  1. Kobe Bryant (August 23, 1978 - January 26, 2020) was an American professional basketball player.
  2. Kobe Bryant married Vanessa Laine in 2001.
  ```

===
For the following original prompt, sentences from the generated text, and incorrect sentences, correct the
     generated text.
Original prompt: ```{original_prompt}```
Generated sentences:
```

{generated_sentences}

```
Incorrect sentences:
```{incorrect_sentences}```
Answer in format:
```

{format}

```
Corrected sentences:
```

Prompt 8: Instructions for the LLM to correct a hallucinated response by pointing out hallucinated sentences. For a description of input fields { }, see Appendix G.
```

```
================================ System Message ================================

You are a top-tier algorithm designed for correcting sentences from the generated text generated by the LLM.
You will be given a original prompt, sentences from the generated text, and list of incorrect facts for each
     sentence.
Empty list means that no incorrect facts were found.
Fact is a triple of the form `(subject, predicate, object)`, just like in the knowledge graph.
You need to correct the sentences to make it more factual and coherent.
Use the provided incorrect facts to correct the sentences, by locating the incorrect facts in the sentences and
     replacing them or removing them.
Put the corrected sentences in the following format, each sentence separated by a new line:
```
1. Corrected sentence 1.
2. Corrected sentence 2.
3. Corrected sentence 3.
```
You must return only the corrected or original sentences, not return any other text or explanation.
Important: The number of corrected sentences should be the same as the number of sentences in the generated text.


================================ Human Message =================================

Below are a number of examples of original prompt, sentences from the generated text, and incorrect sentences
     from the generated text.
- Original prompt: ```Donald Trump is```
  Generated sentences:
  ```
  1. Donald Trump is the president of Canada.
  2. He was born in 1946 and graduated from the University of Pennsylvania.
  ```
  Incorrect facts:
  ```
  1. [(Donald Trump, president, Canada)]
  2. []
  ```
  Corrected sentences:
  ```
  1. Donald Trump is the president of the United States.
  2. Donald Trump was born in 1946 and graduated from the University of Pennsylvania.
  ```
- Original prompt: ```Kobe Bryant is```
  Generated sentences:
  ```
  1. Kobe Bryant (February 17, 1963 - January 26, 2020) was an American professional basketball player.
  2. Bryant married Vanessa Laine in 2001.
  ```
  Incorrect facts:
  ```
  1. [(Kobe Bryant, born, February 17 1963)]
  2. []
  ```
  Corrected sentences:
  ```
  1. Kobe Bryant (August 23, 1978 - January 26, 2020) was an American professional basketball player.
  2. Kobe Bryant married Vanessa Laine in 2001.
  ```

===
For the following original prompt, sentences from the generated text, and incorrect sentences, correct the
     generated sentences.
Original prompt: ```{original_prompt}```
Generated sentences:
```
{generated_sentences}
```
Incorrect facts:
```
{incorrect_facts}
```
Answer in format:
```
{format}
```
Corrected sentences:
```

Prompt 9: Instructions for the LLM to correct a hallucinated response by pointing out hallucinated facts. For a description of input fields { }, see Appendix G.

```
================================ System Message ================================

You are a top-tier algorithm designed for verification whether sentence are supported by the source.
You will be given a sentence, full text, and source.
The full text, as well as the sentence, are generated by the LLM.
Evaluate correctness of the sentence, not the full text.
You have to access the factual consistency of the sentence with the source.
If the sentence is fully supported by the source, you should return `yes`.
If some parts of the sentence are not supported by the source, you should return `no`.
If LLM refused to generate the sentence, e.g. LLM didn't have enough information to generate the sentence, you
     should return `refused`.
You must return a single word, either `yes` or `no` or `refused`.
You must not return any other text or explanation.


================================ Human Message =================================

Below are a number of examples of sentence, full text, and source and their verification results.
- Source: Donald Trump was president of the United States. Donald Trump was born in 1946 and graduated from the
     University of Pennsylvania.
  Full text: Donald Trump was president of the United States. Donald Trump was born in 1946 and graduated from the
       University of Pennsylvania.
  Sentence: Donald Trump was president of the United States.
  Is the sentence supported by the source?: yes
- Source: Michael Jordan is the CEO of PepsiCo. He was born in 1936 and graduated from Yale University.
  Full text: Michael Jordan was born in 1963. He was a basketball player.
  Sentence: He was a basketball player.
  Is the sentence supported by the source?: no
- Source: Kobe Bryant was born in 1978 and played for the Los Angeles Lakers.
  Full text: Kobe Bryant was born in 1978. He played in the NBA.
  Sentence: He played in the NBA.
  note: Los Angeles Lakers is a part of NBA
  Is the sentence supported by the source?: yes

===
For the following sentence, full text, and source, verify if the sentence is supported by the source.
Source: {source}
Full text: {full_text}
Sentence: {input}
Is the sentence supported by the source?
```

Prompt 10: Instructions for the LLM to evaluate factuality of corrected sentences. The prompt takes three inputs: an external Wikipedia biography (*{source}*), a complete corrected response (*{full_text}*), and a specific sentence to evaluate (*{input}*).

```
================================ System Message ================================

You are a top-tier algorithm designed for annotate whether fact is supported by the source.
You will be given a fact and source.
Fact is a triple of the form `(subject, predicate, object)`, just like in the knowledge graph.
You have to access the factual consistency of the fact with the source.
You must return a single word, either `yes` or `no`.
You must not return any other text or explanation.


================================ Human Message =================================

Below are a number of examples of fact and source and their verification results.
- Source: Donald Trump was president of the United States. Donald Trump was born in 1946 and graduated from the
     University of Pennsylvania.
  Fact: ('Donald Trump', 'PRESIDENT', 'Canada')
  Is the fact supported by the source?: no
- Source: Michael Jordan is the CEO of PepsiCo. He was born in 1936 and graduated from Yale University.
  Fact: ('Michael Jordan', 'CEO', 'PepsiCo')
  Is the fact supported by the source?: yes

===
For the following fact and source, verify if the fact is supported by the source.
Source: {source}
Fact: {fact}
Is the fact supported by the source?
```

Prompt 11: Instructions for the LLM to annotate hallucinations at the fact level. The LLM is asked to determine whether a fact (*{fact}*) is supported by an external Wikipedia biography (*{source}*).

```
============================== System Message ==============================

You are a top-tier algorithm designed for verification whether sentence is supported by the provided context.
Your task is to answer whether the given sentence is supported by the context.
Return `yes` if the sentence is supported, `no` otherwise.
You can reason over the context and the sentence to answer.
You can infer the answer from the context, e.g. by inferring new facts from the context.
Do not add new facts other than by inference from the context.
You must return a single word, either `yes` or `no`.
You must not return any other text or explanation.


============================== Human Message ===============================

Below are a number of examples of context and sentences, and their verification results.
- Context: Donald Trump was president of the United States. Donald Trump was born in 1946 and graduated from the
     University of Pennsylvania.
  Sentence: Donald Trump served as president of the United States.
  Is the sentence supported by the context above? yes
- Context: Joe Biden is a politician and president of the United States. He graduated from the University of
     Delaware.
  Sentence: Joe Biden was born in 1942.
  Is the sentence supported by the context above? no
- Context: Michael Jordan is the CEO of PepsiCo. He was born in 1936 and graduated from Yale University.
  Sentence: Michael Jordan played basketball.
  Is the sentence supported by the context above? no
- Context: Kobe Bryant was born in 1978 and played for the Los Angeles Lakers.
  Sentence: Kobe Bryant played in the NBA.
  note: Los Angeles Lakers is a part of NBA
  Is the sentence supported by the context above? yes
- Context: Kobe Bryant was born in 1978 and played for the Los Angeles Lakers.
  Sentence: Bryant was a basketball player.
  note: In this context, Bryant is Kobe Bryant, and Los Angeles Lakers is a basketball team, so Kobe Bryant is a
       basketball player
  Is the sentence supported by the context above? yes

===
For the following context and sentence, verify if the sentence is supported by the context.
Context: {context}
Sentence: {sentence}

Is the sentence supported by the context above?
```

Prompt 12: Enhanced prompt for SelfCheckGPT.