Hallucination Detection in Large Language Models

Introduction

Language models (LMs) have demonstrated remarkable capabilities in generating human-like text. However, they are prone to generating hallucinations - misleading or false outputs that appear factual but lack grounding in reliable sources. This issue undermines the reliability and trustworthiness of LMs, making hallucination detection an essential research area. This report explores three prominent methods for hallucination detection: **Vectara**, **SelfCheckGPT**, and **SAC3**, comparing their effectiveness and discussing their implementation details.

Hallucination Detection Challenges

- **Low Accuracy**: Current hallucination detection models struggle with precision and recall.
- **LLM-as-a-Judge vs. Dedicated Models**: Dedicated models are cheaper and faster but still not fully reliable.

Groundedness Models Explored

Vectara HHEM v2 : A New and Improved Factual Consistency Scoring Model Github

Hallucinations in LLMs occur when generated text includes information absent from the source data. In RAG applications, this can result in summaries containing content not found in the retrieved documents.

- **Efficiency**: As a dedicated model, HHEM v2 operates faster and more cost-effectively than using large LLMs as judges.
- **Accuracy**: Benchmarks indicate that HHEM v2 outperforms models like GPT-3.5 in detecting hallucinations, achieving higher balanced accuracy.
- **Calibrated Scores**: A raw model score lacks meaningful interpretation. Calibration translates this into a probability, making it actionable (e.g., 0.8 means an 80% likelihood of factual consistency). This ensures better trust and usability in real-world applications.
- **Scalability**: The model's ability to handle unlimited context windows allows it to process longer texts, enhancing its utility in various applications.
- **Multilinguality**: Supports English, German, and French, with more languages planned.
- **Unlimited Context Window**: Allows processing longer texts, beneficial for RAG applications.

- **Balanced Accuracy**: On the AggreFact SOTA subset, HHEM v2 achieved a balanced accuracy of 73%, surpassing GPT-3.5's performance.
- **Precision and Recall**: While maintaining high precision, HHEM v2 also focuses on improving recall, ensuring a balanced approach to detecting hallucinations.

Evaluation on Universal Creative Data:

- **Dataset Composition:** The dataset consists of structured data entries containing:
    - **Source/Context** (e.g., technical/legal text)
    - **Question** related to the source
    - **Grounded Answer** (factually correct based on the source)
    - **Ungrounded Answer** (plausible but incorrect response)
    - **Explanation** of why the ungrounded response is incorrect
- **Documents Used:**
    - GPS 23.0 1.md: 26 samples
    - Subsystem Controller Specification (SSC).md: 36 samples
    - OHS 3.2.md: 40 samples
- **ChatGPT-Assisted Data Generation:** A structured dataset was created using the following prompts:

  "I am working on a hallucination detection task. I need to evaluate my model on the attached document. Can you provide me with a structured dataset that includes: A source/context (e.g., legal text). A question based on the source/context. Two AI-generated answers: One that is faithful to the source (grounded). One that includes hallucinated information which can be factual or non factual (ungrounded). A brief explanation of why the ungrounded response is incorrect. Please make sure the ungrounded response is plausible-sounding but factually incorrect."

  "Great, can you give me additional entries?"

  This method helped generate additional entries and ensured a diverse dataset.

- **Results**:

Dataset	**Precision**	**Recall**
GPS 23.0 1.md	0.917	0.846
Subsystem Controller Specification (SSC).md	1.000	0.944
OHS 3.2.md	1.000	0.900

- **Implementation Challenges and Retrieval-Based Improvements:** Despite the model's claim of an "unlimited context window," full document inputs exceeded practical processing limits. A **retrieval mechanism** was implemented to extract the most relevant text before sending it to the model, significantly reducing memory usage while preserving key context.
- **Future Improvements:**
    - Experimenting with different **retrieval strategies** to optimize relevance.
    - Matching **document titles and sections** if applicable, instead of relying solely on a context window.
- **Ablation Study on OHS Dataset:** Experiments were conducted with different **chunk sizes**, **overlap values**, and **top-k selections** to determine optimal parameters for retrieval. The best configuration for the **OHS dataset** was **Chunk Size: 400, Overlap: 50**, but this may vary across different datasets.

Chunk Size	Overlap	Top-k	Precision	Recall	Execution Time (ms)
400	50	3	1.000	0.692	31.36
400	50	2	1.000	0.692	20.44
300	50	2	0.857	0.462	14.27
450	50	2	1.000	0.462	25.79
350	50	2	1.000	0.462	17.57
400	50	1	1.000	0.692	10.16
350	50	1	1.000	0.308	8.34
450	50	1	1.000	0.308	11.82

- **Implementation Guide:** Below is an example of how hallucination detection is implemented using Vectara HHEM v2.

```
pairs = [ ("The capital of France is Berlin.", "The capital of France is Paris."),
          ('I am in California', 'I am in United States.') ]

from transformers import AutoModelForSequenceClassification

# Load the model
model = AutoModelForSequenceClassification.from_pretrained(
 'vectara/hallucination_evaluation_model', trust_remote_code=True)

# Get predictions
predictions = model.predict(pairs) # predictions stores the score for each sample
```

GitHub: https://github.com/ilkinisler/LM-Hallucinations/blob/main/Vectara/demo.ipynb

- **Conclusion:**
  - **Vectara HHEM v2** is a highly effective factual consistency model with superior efficiency and accuracy over LLM-based evaluators.
  - **Retrieval-based methods** significantly improve real-world usability by reducing memory use while maintaining relevance.
  - **Optimal parameters** vary across datasets, requiring dataset-specific tuning.
  - **Further improvements** could include refining retrieval methods and leveraging structured metadata (titles/sections) for more precise context extraction.
  - Future work should focus on improving retrieval mechanisms and exploring cross-domain applications for factual consistency evaluation.

License: Open-source under Apache 2.0.

Factuality Models Explored

1. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models (EMNLP 2023) GitHub

SelfCheckGPT employs **self-consistency and model agreement** to detect hallucinations. Instead of external verification, it generates multiple responses to the same prompt and analyzes the consistency across outputs. A zero-resource hallucination detection method. Does not require access to the model's probability distributions nor external databases. If the model is knowledgeable, responses should be consistent; inconsistent responses indicate hallucinations. Compares the original response with multiple stochastically sampled responses using methods like BERTScore, question-answering, and natural language inference (NLI).

- Strenghts: Works without external databases, making it suitable for APIs like ChatGPT.
- Metrics:
  - Sentence-Level Hallucination Detection: Measures how effectively SelfCheckGPT detects non-factual sentences.
  - Passage-Level Factuality Assessment: Ranks passages based on their overall factuality.
- License: MIT.

2. SAC3: Reliable Hallucination Detection in Black-Box Language Models (EMNLP 2023) GitHub

Builds on the concept of self-consistency checks by adding semantic and cross-model evaluations. The approach addresses two types of hallucinations: question-level (misunderstandings of the input question) and model-level hallucinations (inaccuracies due to the model's internal limitations). Uses semantically equivalent rephrasing of questions and compares responses across different models to detect inconsistencies.

- Key Findings:
    - Limitations of Existing Methods: Standard self-consistency checks (e.g., sampling-based methods) often fail to detect certain types of hallucinations, particularly those stemming from: (1) Ambiguous or rephrased questions or (2) Model-specific biases or limitations.
- Performance: SAC3 outperforms prior methods in detecting hallucinations, particularly in challenging cases involving question rephrasings or different model architectures. Makes QA systems more robust to question phrasing and contextual ambiguities. Extends beyond QA to any generative task requiring factual reliability, such as summarization or information retrieval.
    - Factuality Assessment: Measures consistency with external knowledge or human evaluations.
    - Response Consistency: Evaluates agreement across different models or question perturbations.
- License: Apache-2.0.

Implementation Details:

- **Dataset(wiki_bio_gpt3_hallucination)**
    - **gpt3_text** – Text generated by GPT-3.
    - **wiki_bio_text** – Reference text from the WikiBio dataset.
    - **gpt3_sentences** – Sentence-level breakdown of GPT-3 generated text.
    - **annotation** – Labeled annotations indicating hallucination presence.
    - **wiki_bio_test_idx** – Index mapping to the original WikiBio dataset.
    - **gpt3_text_samples** – A subset of 20 generated text samples for each data point.
- The dataset consists of **238 rows**, where each row corresponds to a single instance of generated and reference text.
- **GPT-3 generated text** is evaluated against **WikiBio ground truth** to detect hallucinations.
- Each **gpt3_text** has **20 sampled variations** stored in 'gpt3_text_samples'.
- **Annotations** provide human-labeled feedback on hallucination presence.

Code: https://github.com/ilkinisler/LM-Hallucinations/blob/main/SelfCheckGPTvsSAC.ipynb

Comparison:

Method	Detecting False (AUC)	Detecting False (harder/ less trivial) (AUC)	Detecting True (AUC)
SelfCheckGPT	92.50	45.17	66.08

SAC3	60.23	21.79	49.69

Comparison of All Methods

- **SelfCheckGPT** is useful when hallucinations stem from internal inconsistencies but does not validate against external facts.
- **SAC3** is a hybrid approach that adds robustness via statistical methods and adversarial attacks.
- **Vectara** is effective when access to an external knowledge base is available but fails when verifying novel or opinion-based content.

Future Improvements

Despite their strengths, all these methods have limitations. Future research should focus on:

- **Hybrid Approaches**: Combining retrieval-based verification (Vectara) with self-consistency checks (SelfCheckGPT) for a more comprehensive solution.
- **Improved Knowledge Bases**: Improving the reliability of external sources to minimize false positives in Vectara. Make them simpler to help with retrieval or relevant parts.
- **Scalability**: Optimizing computational efficiency for SAC3, particularly in large-scale real-time applications.