Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models

Weihang Su[*1], **Changyue Wang**[†1], **Qingyao Ai**[‡1], **Yiran Hu**[1], **Zhijing Wu**[2], **Yujia Zhou**[3]
Yiqun Liu[1]

[1]Department of Computer Science and Technology, Tsinghua University
[2]School of Computer Science and Technology, Beijing Institute of Technology
[3]School of Information, Renmin University of China

Abstract

Hallucinations in large language models (LLMs) refer to the phenomenon of LLMs producing responses that are coherent yet factually inaccurate. This issue undermines the effectiveness of LLMs in practical applications, necessitating research into detecting and mitigating hallucinations of LLMs. Previous studies have mainly concentrated on post-processing techniques for hallucination detection, which tend to be computationally intensive and limited in effectiveness due to their separation from the LLM's inference process. To overcome these limitations, we introduce MIND, an unsupervised training framework that leverages the internal states of LLMs for real-time hallucination detection without requiring manual annotations. Additionally, we present HELM, a new benchmark for evaluating hallucination detection across multiple LLMs, featuring diverse LLM outputs and the internal states of LLMs during their inference process. Our experiments demonstrate that MIND outperforms existing state-of-the-art methods in hallucination detection[1].

1 Introduction

In recent years, Large Language Models (LLMs) have demonstrated remarkable performance in a variety of natural language processing (NLP) applications (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023a; Scao et al., 2022; Zhang et al., 2022). However, the widespread adoption of LLMs has highlighted a critical problem, i.e., hallucination. Hallucination refers to the cases where LLMs generate responses that are logically coherent but factually incorrect or misleading (Maynez et al., 2020; Zhou et al., 2020; Liu et al., 2021; Ji et al., 2023). Such flaw hurts the effectiveness and robustness of LLMs in real-world NLP applications, underlining the pressing necessity for research on detecting and mitigating hallucinations in LLMs.

Existing studies on hallucination detection for LLM mainly focus on how to identify possible fact-related errors in LLM's outputs (Lin et al., 2021; Li et al., 2023d; Manakul et al., 2023). For example, WikiBio GPT3 (Manakul et al., 2023), a well-known benchmark for hallucination detection, hired human annotators to annotate a couple of true and false response. Hallucination detection methods are then evaluated based on their accuracy in predicting the truthfulness of the answers. Based on this paradigm, considerable hallucination detection methods have been proposed with the goal of taking a piece of text as input and predicting whether there are hallucinations in the inputs (Manakul et al., 2023; Zhang et al., 2023b; Azaria and Mitchell, 2023). Since they detect hallucinations after LLMs finish the inference process, we refer to these hallucination detection methods as post-processing methods.

Unfortunately, the post-processing methods widely used in existing studies are suboptimal for the applications of hallucination detection models for LLMs in practice. First, existing post-processing methods often suffer from extreme computation costs and high latency. In order to identify hallucinations in input text without ground truth references (otherwise the task would downgrade to a simple fact verification task), hallucination detection models need to be powerful and knowledgeable on their own. SOTA detection methods are often implemented with LLMs (e.g., chatGPT, LLaMA, OPT) directly (Manakul et al., 2023; Azaria and Mitchell, 2023; Zhang et al., 2023b), which makes the cost of hallucination detection on par or even larger than the inference process of many LLMs. Second, post-processing methods

[*]swh22@mails.tsinghua.edu.cn
[†]contributed equally
[‡]Corresponding Author: aiqy@tsinghua.edu.cn
[1]We have open-sourced all the code, data, and models in GitHub: https://github.com/oneal2000/MIND/tree/main.

Findings of the Association for Computational Linguistics ACL 2024, pages 14379–14391
August 11-16, 2024

are intrinsically limited in model capacities. As post-processing methods detect hallucinations independently with the inference process of LLMs, they can't analyze how hallucinations are generated from scratches in each LLM. Some studies tried to bypass this issue through the construction of proxy models (Azaria and Mitchell, 2023). However, such proxy models must be trained with extensive human annotations, otherwise they cannot capture the unique characteristics of each LLM. Such manual annotation data are expensive to collect and not preferable considering the rapid developments of LLM techniques.

To address these limitations, we propose a novel reference-free, unsupervised training framework **MIND**, i.e., unsupervised Modeling of INternal states for hallucination Detection of Large Language Models. We highlight MIND with the following advantages: (1) Unsupervised. In contrast to previous works, MIND is an unsupervised framework that directly extracts pseudo-training data from Wikipedia. It doesn't require any manual annotation for the training of the hallucination detector. (2) Real-time. Compared with the existing post-processing methods, MIND is a real-time hallucination detection framework designed to reduce computational overhead and detection latency. With a simple multi-layer perceptron model built upon the contextual embeddings of each token in LLM's inference process, MIND can conduct the hallucination detection in a real-time process. (3) Compatibility. MIND is a lightweight framework that can be incorporated into any existing Transformer-based LLMs.

Further, to facilitate future research and to improve the reproducibility of this paper, we introduce a new benchmark for LLM hallucination detection named **HELM**: Hallucination detection Evaluation for multiple LLMs. In contrast to previous benchmarks that only provide the text generated based on hand-crafted heuristics (Liu et al., 2021; Azaria and Mitchell, 2023) or a single LLM (Manakul et al., 2023; Li et al., 2023d), HELM provides not only the texts produced by six different LLMs (together with human-annotated hallucination labels) but also the contextualized embeddings, self-attentions and hidden-layer activations recorded in the inference process of each LLM, which could serve as the snapshots of each LLM's internal states during their inference process.

In summary, the contributions of our paper are as follows:

- We propose MIND, an unsupervised training framework for real-time hallucination detection based on the internal states of LLM.
- We introduce HELM, a hallucination detection benchmark featuring text from six LLMs and contains the internal states of each LLM during the text generation process.
- We evaluate MIND and existing hallucintion detection baselines with human annotations. The experimental results show that MIND outperforms existing hallucination detection methods.

2 Problem Formulation

Hallucination Detection can be defined as a binary classification problem. The objective is to judge whether the given output from an LLM is a hallucination (false or misleading information) or a non-hallucination (accurate and relevant information). This diverges from traditional fact verification tasks. In contrast to traditional fact verification tasks that mainly assess the factual accuracy of text, hallucination detection in LLMs goes beyond simply evaluating the truthfulness of the information. It focuses more on the comprehensive analysis of the characteristics intrinsic to the generative models. Examples of hallucination detection methods include the consistency of multi-responses to the same question (Manakul et al., 2023), the LLMs' confidence in its generated content (Zhang et al., 2023b), the internal states of LLMs during the text generation process (Azaria and Mitchell, 2023), etc.

3 Methodology

In this section, we introduce the details of our proposed framework **MIND**, i.e., unsupervised Modeling of INternal-states for hallucination Detection of Large Language Models. The MIND framework consist of two steps: automatic training data generation and hallucination classifier training.

3.1 Unsupervised Training Data Generation

Our proposed Unsupervised Training Data Generation involves automatically annotating hallucinations in content produced by a chosen LLM. Let a specific LLM be L_i, then our method aims to create customized training data for training a Hallucination Detection Model for L_i. Figure 1 illustrates our proposed automatic data generation

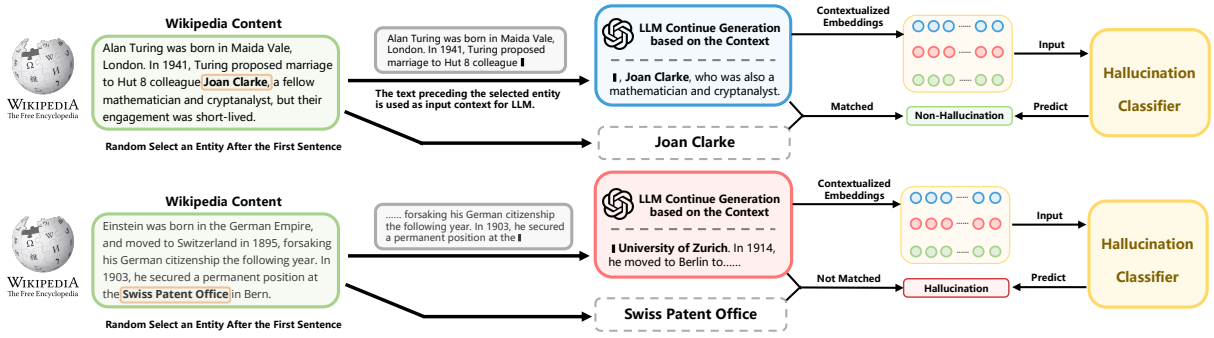

Figure 1: An illustration of the automatic training data generation process of our proposed framework: MIND.

process. The process starts by selecting a subset of high-quality Wikipedia articles, denoted as W (for example, WikiText-103 (Merity et al., 2016)). This subset is represented as $W = \{w_1, w_2, ..., w_n\}$, where each w_i $(1 \leq i \leq n)$ is an individual article. Following that, each article w_i is truncated at a randomly selected entity (except those that appear at the beginning of a sentence) occurring after the first sentence of the article. The selected entity is denoted as e_i. The truncated article is denoted as $w_i^{'} = \text{truncate}(w_i, e_i)$. Subsequently, each truncated article $w_i^{'}$ is inputted into the LLM L_i. L_i is then tasked with a in free-form text generation based on $w_i^{'}$. The continuation text produced by the LLM for $w_i^{'}$ is marked as G_i which is truncated at the end of the first sentence. During the generation of G_i, the internal states of the LLM are recorded, denoted as S_i. Each G_i is then compared to the original article w_i to assess if the LLM can accurately continue writing about the original entity e_i. The key criterion for this comparison is whether the beginning of G_i contains the correct entity e_i as it appears in the original article. If G_i starts with the correct entity e_i, it is labeled as non-hallucination and represented as $H_i = 0$. Conversely, if G_i does not start with the correct entity e_i, and e_i does not appear in G_i, it is labeled as hallucination and represented as $H_i = 1$. For each article, a data tuple is created, represented as $D_i = (L_i, w_i, G_i, S_i, H_i)$, encompassing the selected LLM L_i, the original Wiki article w_i, generated text G_i, the internal states S_i, and the hallucination label H_i.

3.2 Hallucination Classifier Training

3.2.1 Feature Selection

In Transformer-based models, the generation of each token is directly based on its contextualized embedding vectors (also named hidden states), which represents the LLM's comprehension and semantic representation of the previously generated tokens. These contextualized embeddings also manifest the behavior of the LLM in making next token predictions, encompassing elements of uncertainty. Thus, contextualized embedding can serve as an important reference for judging LLMs' Hallucinations. In this section, we investigate a simple research question: Can we detect hallucinations in LLMs using contextualized embeddings?

To verify this, we select the contextualized embeddings of different tokens in various Transformer layers and use a multilayer perceptron (MLP) to classify the embedding during hallucinations from those during non-hallucinations. If it is possible to classify these vectors using a simple MLP, it indicates distinctions between the token embeddings of LLMs during hallucination and non-hallucination states. We train this MLP on 5k samples generated by LLaMA2-13B-Chat via the automatic data generation process described in the previous section, and test the prediction accuracy on 5k samples. The classification accuracy based on the contextualized embedding vectors at various positions of LLM and their combinations are as follows:

Layer	Embedding	Formula	Acc
All	All	$\frac{1}{N}\sum_{j=1}^{N}\frac{1}{n}\sum_{i=1}^{n} H_j^i$	0.7054
First & Last	All	$\frac{1}{2}(\sum_{i=1}^{n} H_1^i + \sum_{i=1}^{n} H_N^i)$	0.6929
Last	All	$\frac{1}{n}\sum_{i=1}^{n} H_N^i$	0.6986
First	All	$\frac{1}{n}\sum_{i=1}^{n} H_1^i$	0.6529
Last	Last	H_N^n	0.7123
All	Last	$\frac{1}{N}\sum_{j=1}^{N} H_j^n$	0.6986
Last	All & Last	$\frac{1}{2}(\frac{1}{n}\sum_{i=1}^{n} H_N^i + H_N^n)$	0.7191

In the table above, N represents the total number of Transformer layers in this LLM, and n is the length of the input token sequence, "H_j^i" represents the contextualized embedding vector of the i^{th} token in the j^{th} Transformer layer of the LLM. The experimental results demonstrate that we can indeed classify these vectors using a simple MLP. This finding

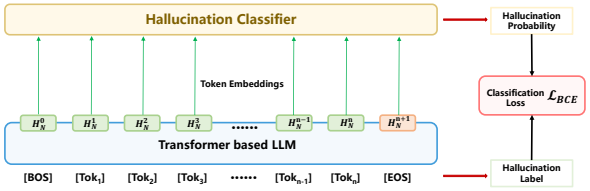

Figure 2: An illustration of the hallucination classifier training process. "H_j^i" represents the token embedding of the i^{th} token in the K^{th} Transformer layer.

indicates that the contextualized embedding vectors of LLMs exhibit discernible differences during hallucination and non-hallucination states. Furthermore, it is noteworthy that an effective distinction can be achieved by merely utilizing the last token's contextualized embedding of the final layer. In Section 6, we extensively explored the generality and robustness of this method in validating existing open-source large models, and conducted more detailed experiments.

3.2.2 Training Process

The MIND classifier is architecturally structured as a Multilayer Perceptron (MLP) network. For the activation functions of the MIND classifier, we select the Rectified Linear Unit (ReLU). For the training of the classifier, the input is a $1 \times n$ matrix, where n represents the dimension of the LLM's contextualized embedding. We choose the contextualized embeding of the last token of last Transformer layer as the input of the MIND classifier. The output of the hallucination classifier is a binary label, indicating whether the LLM is experiencing a hallucination while generating a specific segment of text. The process can be mathematically represented as follows:

$$P = MLP(ReLU(\mathbf{W} \cdot \mathbf{H} + \mathbf{b})), \quad (1)$$

where P is the hallucination label predicted by the classifier, $\mathbf{H}$ is a $1 \times n$ matrix representing the selected hidden states, $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector of the MLP respectively, and MLP represents the function implemented by the multilayer perceptron. For the loss function, we use the Binary Cross-Entropy (BCE) Loss (De Boer et al., 2005) to optimize the hallucination classifier, which is defined as:

$$\mathcal{L}_{BCE}(y_i, p_i) = y_i \log(p_i) + (1 - y_i) \log(1 - p_i), \quad (2)$$

where y_i represents the actual label of the i^{th} example, and p_i indicates the predicted probability that the i^{th} example belongs to the positive class.

3.3 Real-time Hallucination Detection

Upon successfully training the Hallucination Classifier as outlined in Section 3.2, the MIND framework enable real-time hallucination detection in the chosen LLM. Specifically, the Hallucination Classifier receives the contextualized embeddings of each token during the LLM's inference process. It then outputs a probability indicating the probability of hallucination in the LLM's output.

4 The HELM Benchmark

In this section, we introduce the details of our proposed new benchmark HELM: Hallucination detection Evaluation for multiple LLMs.

4.1 Data Generation

We select six widely-used, open-source LLMs for annotation. These models range in complexity, encompassing both base models and chat models, with sizes varying from 6 billion to 40 billion parameters, including Falcon-40B (Almazrouei et al., 2023), GPT-J-6B (Wang and Komatsuzaki, 2021), LLaMA-2-Base-7B (Touvron et al., 2023b), LLaMA-2-Chat-7B (Touvron et al., 2023b), LLaMA-2-Chat-13B (Touvron et al., 2023b), and OPT-6.7B (Zhang et al., 2022). For the generation process, we start with randomly sampling 50,000 articles from a high-quality Wikipedia corpus: WikiText-103 (Merity et al., 2016)[2]. Following that, the selected LLMs were tasked with prompt-based continuation writing. For base LLMs and chat LLMs, we have designed different prompt templates respectively which is detailed in Appendix B.

4.2 Human Annotation

Annotators are tasked with evaluating the truthfulness of the content generated by the LLM. We only take factual errors into account, without considering grammatical mistakes or subjective opinions[3]. Furthermore, annotators are instructed not to use generative models (such as ChatGPT) for assistance. Wikipedia is the primary source for fact-checking, followed by Google's search engine when Wikipedia lacks relevant information.

The annotation process begins with annotators manually dividing the passage into discrete

[2]The selected articles have no overlap with the training set of MIND.

[3]For example, stating that "Ronald Reagan was a President of the United States" is a fact, whereas commenting on "Ronald Reagan's attractiveness" is a matter of opinion.

Table 1: The statistics of the HELM dataset, where H=1 indicates that the sentence is annotated as hallucination, and H=0 indicates non-hallucination. LLB stands for "LLaMA2-Base", and LLC stands for "LLaMA2-Chat".

	#Sentence	#H=1	#H=0	#passage
Falcon-40B	521	261	260	196
GPTj-6B	572	172	400	208
LLB-7B	565	243	322	207
LLC-7B	617	308	309	204
LLC-13B	596	329	267	203
OPT-7b	566	181	385	201
Total	3582	1494	2088	1224

sentences. Annotators are then required to use Wikipedia and Google search to verify the truthfulness of every sentence. If neither Wikipedia nor the top 20 search engine results confirm the accuracy of a sentence, it is marked as 'Unverifiable.' Additionally, in instances of hallucinations, annotators are required to indicate the location of the hallucination within the sentence. Every sentence and passage in the dataset was annotated by two distinct annotators. Only the data with matching annotations from both annotators was adopted.

4.3 Benchmark Analysis and Usage

The statistics of our proposed HELM dataset are shown in Table 1. Our dataset ultimately comprises a total of 3342 sentences, each of which corresponds to a label indicating whether it is a hallucination. Additionally, it includes the complete contextualized embeddings and hidden layer activations of the LLM during the text generation process. These 3342 sentences are derived from 1224 distinct passages. HELM provides two levels of hallucination detection: Sentence Level and Passage Level. Since every annotated sentence originates from a passage, HELM dataset also encompasses a task for Passage Level Hallucination Detection. The hallucination at the passage level depends on its sentences. If any sentence within a passage is identified as hallucinated, the entire passage is classified as hallucinated.

To use our benchmark, users can either directly use the code provided in our open-source GitHub [4] repository or download our data and implement it themselves.

[4]We have publicly shared the HELM dataset and code in this anonymous GitHub repository: https://github.com/oneal2000/MIND/tree/main

5 Experimental Settings

5.1 Dataset and Metrics

We evaluate MIND and other baselines on our proposed dataset HELM which is detailed in section 4. We use the AUC (Area Under the Curve) and the Pearson correlation coefficient (corr) with human-annotated relevance as evaluation metrics. We conducted experiments at both the sentence-level and the passage-level hallucination detection.

5.2 Baselines

We choose the following reference-free hallucination detection methods as baselines:

- **Predictive Probability (PP)** (Manakul et al., 2023). A method for detecting LLMs' hallucinations based on the probability of tokens generated by LLMs. PP_{max}, PP_{min}, and LN-PP (Length Normalised PP) indicates max pooling, min pooling and mean pooling methods to combine multiple tokens generated by LLM respectively.

- **Predictive Entropy (PE)** (Kadavath et al., 2022) *PE* is widely used to evaluate the uncertainty inherent in the model's output distribution (Kadavath et al., 2022). For each token t_i in the LLM output, the *PE* is defined as:

$$PE = -\sum_{\tilde{w} \in \mathcal{W}} p_i(\tilde{w}) \log p_i(\tilde{w}), \tag{3}$$

  where $p_i(\tilde{w})$ represents the likelihood of generating word $\tilde{w}$, and $\mathcal{W}$ is the vocabulary of the LLM. PE_{max}, PE_{min}, LNPE (Malinin and Gales, 2020) (Length Normalised PE) indicates max pooling, min pooling and mean pooling strategies to combine multiple tokens generated by LLM respectively.

- **SelfCheckGPT** (Manakul et al., 2023) (SCG) is a strong hallucination detection baseline that is designed based on the principle that if an LLM has knowledge of a given concept, sampled responses are likely to be similar and contain consistent facts. SCG employs four distinct techniques: SCG_BERTScore, SCG_QA, SCG_NLI, and SCG_n-gram, each of which evaluates the consistency of responses from the LLM.

- **SAPLMA** (Azaria and Mitchell, 2023) is a novel hallucination detection training method, wherein a classifier is trained using the activation values derived from the hidden layers of LLMs.

Table 2: The overall experimental results of MIND and other baselines on the HELM benchmark. The best results are in bold. LLB stands for "LLaMA2-Base", SCG stands for "SelfCheckGPT", and LLC stands for "LLaMA2-Chat". Falcon and GPT-J are Falcon-40B and GPT-J-6B, respectively.

	HELM Sentence Level AUC						HELM Passage Level AUC					
Baselines	**Falcon**	**GPT-J**	**LLB-7B**	**LLC-13B**	**LLC-7B**	**OPT-7B**	**Falcon**	**GPT-J**	**LLB-7B**	**LLC-13B**	**LLC-7B**	**OPT-7B**
PE-max	0.6479	0.7497	0.6851	0.4439	0.4931	0.7263	0.8347	0.8875	0.8400	0.5933	0.6988	0.8851
PE-min	0.5757	0.7044	0.5878	0.3164	0.4411	0.7228	0.7115	0.7595	0.7587	0.5918	0.6409	0.8075
PP-max	0.5749	0.7074	0.5872	0.3013	0.4166	0.7302	0.7063	0.7585	0.7543	0.5344	0.6614	0.8100
PP-min	0.5546	0.7413	0.6025	0.3725	0.4479	0.7121	0.7649	0.8526	0.7969	0.5301	0.6870	0.8384
LNPP	0.5327	0.6927	0.6098	0.2673	0.3536	0.7206	0.7008	0.8039	0.7755	0.5522	0.5952	0.8374
LNPE	0.5442	0.6980	0.6175	0.3352	0.4114	0.7019	0.7145	0.8111	0.7776	0.5878	0.6704	0.8349
SCG-MQAG	0.5409	0.7873	0.6401	0.4040	0.4613	0.7593	0.7275	0.8827	0.8196	0.6672	0.7284	0.8594
SCG-NG	0.5218	0.7549	0.5365	0.2650	0.3105	0.7490	0.7124	0.8579	0.7155	0.4983	0.5771	0.8340
SCG-BS	0.6418	0.7424	0.6178	0.3026	0.3760	0.6594	0.7428	0.8165	0.7631	0.4760	0.5938	0.7597
SCG-NLI	0.6846	0.8680	0.7644	0.5834	0.6565	0.8103	0.8121	0.9384	0.8897	**0.7559**	0.7951	0.9096
SAPLMA	0.5128	0.6987	0.5777	0.3047	0.4066	0.6212	0.7236	0.8294	0.7823	0.5179	0.6265	0.7476
EUBHD	0.7509	0.7593	0.6479	0.4658	0.4805	0.7563	0.8659	0.8560	0.7859	0.6685	0.7126	0.8545
GPT4-HDM	0.6329	0.7843	0.6583	0.4108	0.5127	0.7972	0.8150	0.9183	0.8625	0.5900	0.7768	0.9196
MIND (ours)	**0.7895**	**0.8774**	**0.7876**	**0.6043**	**0.6755**	**0.8835**	**0.8886**	**0.9599**	**0.9048**	0.7175	**0.8547**	**0.9449**

	HELM Sentence Level Corr						HELM Passage Level Corr					
Baselines	**Falcon**	**GPT-J**	**LLB-7B**	**LLC-13B**	**LLC-7B**	**OPT-7B**	**Falcon**	**GPT-J**	**LLB-7B**	**LLC-13B**	**LLC-7B**	**OPT-7B**
PE-max	0.2405	0.0839	0.2032	0.2375	0.2029	0.0573	0.3106	0.2660	0.2561	0.2258	0.1956	0.2180
PE-min	0.1204	-0.0316	0.0152	-0.0166	0.0438	0.0601	0.0855	-0.0645	0.0928	0.2382	0.0832	0.0823
PP-max	0.1193	-0.0467	0.0154	-0.0404	-0.0337	0.0732	0.0843	-0.0672	0.0837	-0.0426	0.1426	0.0943
PP-min	0.1504	0.2057	0.1842	0.1092	0.1287	0.1808	0.1815	0.2454	0.2231	0.0940	0.1668	0.1191
LNPP	0.0667	0.0773	0.1752	-0.0971	-0.0339	0.2075	0.0706	0.1601	0.2297	0.2188	0.1169	0.2417
LNPE	0.0936	0.0376	0.1461	0.0826	0.0858	0.1216	0.1020	0.1261	0.2018	0.2867	0.2258	0.1879
SCG-MQAG	0.0369	0.2306	0.1822	0.1820	0.1856	0.2151	-0.0851	0.3145	0.2278	0.2902	0.2581	0.2993
SCG-NG	0.0995	0.1770	0.0590	-0.0483	-0.1016	0.1167	0.0575	0.2222	0.0785	-0.1065	-0.1348	0.1021
SCG-BS	0.2293	0.0745	0.1268	-0.0136	-0.0378	-0.0563	0.2471	0.1288	0.1447	-0.0138	-0.0504	-0.0732
SCG-NLI	0.3789	0.4087	0.3809	0.3092	0.3835	0.3312	0.4062	0.4217	0.4389	**0.4145**	0.4005	0.4091
SAPLMA	0.0208	0.0456	-0.0015	-0.0003	0.0130	-0.1410	0.0900	0.1298	0.0271	-0.0714	0.0519	-0.1433
EUBHD	0.3161	0.1057	0.1170	0.2115	0.1043	0.1423	0.3447	0.1546	0.1693	0.3079	0.0512	0.1342
GPT4-HDM	0.1346	0.1096	0.0086	0.0769	0.1199	0.1678	0.0812	0.2539	0.1876	0.0530	0.1883	0.2372
MIND (ours)	**0.5032**	**0.5244**	**0.4857**	**0.4273**	**0.4938**	**0.4760**	**0.5251**	**0.5296**	**0.4911**	0.3823	**0.4778**	**0.5636**

- **EUBHD** (Zhang et al., 2023b). Enhanced Uncertainty-Based Hallucination Detection is a SOTA hallucination detection method based on the uncertainty of LLMs' outputs.

- **GPT4-HDM** (Li et al., 2023d). Through a simple prompt template, this method directly use GPT-4 as hallucination detection model to judge whether the text generated by other LLMs contains hallucination. The prompt templates is detailed in Appendix C.

5.3 Implementation Details

- **NER**: For the Named Entity Recognition (NER) component of MIND, we follow the methodologies in prior studies (Liu et al., 2021; Tarcar et al., 2019). Specifically, we utilized the Spacy library, a tool recognized for its effectiveness and efficiency in NER as evidenced by previous research (Shelar et al., 2020).

- **MIND**: The MIND classifiers employs a 4-layer Multilayer Perceptron (MLP) network, featuring a 20% dropout rate applied at the initial layer. The architecture of this network is characterized by a progressively decreasing hidden layer size, with dimensions set at 256, 128, 64, and 2 for each consecutive layer. In terms of activation functions, the Rectified Linear Unit (ReLU) is selected. The learning rate is set to 5e-4, the weight decay is set to 1e-5, and the training batch size is set to 32. For the ablation experiments of important hyperparameters, we have discussed in detail in Section 6.3.

- **LLM Configuration**: For the selected LLMs, we directly download model parameters from the official Hugging Face repositories for each model, and use the code provided by Hugging Face to conduct text generation. For the generation configuration, we use the official default configurations provided by each model. We introduce our selected models Appendix D.

6 Experimental Results

We conduct experiments to verify our our proposed Hallucination Detection method: MIND. Specifically, this section studies the following research questions (RQ):

- **RQ1**: Is hallucination detection based on the internal states of Large Language Models (LLMs)

broadly effective across existing LLMs?

- **RQ2**: How effective and efficient is MIND on hallucination detection tasks?
- **RQ3**: For the training of hallucination detection models, is it necessary to use training data generated by the model itself?

6.1 Overall Results of MIND and Baselines

In this subsection, we present a comprehensive evaluation of our MIND framework against baselines. The objective is to answer Research Questions RQ1 and RQ2. Our experimental results are shown in Table 2. The key findings are summarized as follows: (1) MIND, our proposed unsupervised internal states-based hallucination detection method, demonstrates broad effectiveness in all six LLMs. This result validates the hypothesis that hallucination detection based on the contextualized embeddings of LLMs is effective across various models. (2) MIND outperforms existing reference-free hallucination detection methods at both sentence and passage levels. This superiority not only highlights the robustness of MIND, but also underscores the effectiveness of the unsupervised training framework proposed in this study. (3) SCG-NLI emerged as the second-best hallucination detection method after MIND. Remarkably, it even surpasses MIND in performance on certain models. However, a critical drawback of SelfCheckGPT is its latency. Specifically, the time taken to detect hallucination in a response is ten times longer than generating the response itself. This makes MIND a more viable option for scenarios requiring real-time hallucination detection. (4) SAPLMA, though effective on its own training dataset, exhibits suboptimal performance when applied to the real-time hallucination detection during the text generation process. This could be attributed to the model's overfitting to the specific training data, which is not generated by the LLM itself.

6.2 Efficiency

This section compares the efficiency of the MIND classifier with other hallucination detection methods, focusing on LLaMA-7B-Chat. We examine the average generation time for each question, and the average time of these hallucination detection methods take to detect hallucinations which are demonstrated in Table 3. MIND is shown to be highly efficient, taking only 3% of the LLM's response generation time for hallucination detection.

Table 3: Efficiency of MIND and Other Hallucination Detection Baselines. This table presents the average time taken to detect hallucinations in an LLM response and the percentage of time spent on hallucination detection in the complete LLM response time.

	LLaMA2-7B-Chat	
	Inference Time	**Percentage**
LLM's Response	1.52 s	100.0 %
PP & PE & EUBHD	< 0.01 s	0.000 %
SCG-MQAG	27.06 s	1785 %
SCG-Ngram	15.66 s	1033 %
SCG-BertScore	27.29 s	1800 %
SCG-NLI	15.31 s	1010 %
GPT4	4.29 s	282.2 %
MIND	0.05 s	3.289 %

Table 4: Effectiveness of Training OPT-7B and LLaMA2-Chat-7B with Training Data Generated by Different Models. We utilized the HELM dataset and used correlation (Corr) as evaluation metric. The best results are in bold.

		LLM Used for Generating Training Data				
		GPTJ	**MPT-7B**	**OPT-7B**	**LLB-7B**	**LLC-7B**
Sentence-level	**OPT**	0.4117	0.2236	**0.4760**	0.4136	0.3312
Passage-level	**OPT**	0.5001	0.2849	**0.5636**	0.4981	0.4349
Sentence-level	**LLC**	0.2577	0.2696	0.2357	0.3811	**0.4938**
Passage-level	**LLC**	0.3073	0.2048	0.2159	0.3429	**0.4778**

This makes MIND ideal for real-time applications, unlike SelfCheckGPT, which is less efficient due to requiring multiple LLM responses. Since the LLM records the probability of generating each token during response generation, methods like Predictive Probability (PP) and Predictive Entropy (PE) are easy to compute and require negligible time, making them the least time-consuming. Nonetheless, when considering both efficiency and effectiveness, MIND emerges as the superior reference-free hallucination detection method.

6.3 Ablation Studies

6.3.1 Impact of Customized Training Data

This section focuses on the importance of incorporating model-specific training data to answer RQ3. To be specific, we validate the impact of customized training data by comparing the performance of hallucination classifiers in the following two settings. In the first setting, training data is customized for the hallucination classifier of each LLM. **In this setting, the input to the classifier is the internal state of the LLM during its text generation process.** In the second setting, we use

Table 5: The performance of MIND at different sizes of training data. We report the accuracy of the dev set.

#Num	1024	2048	3072	4096	5120
Accuracy	0.6929	0.7066	0.7158	0.7237	0.7192

Table 6: The performance of MIND at different depths of the classifier. We report the accuracy of the dev set.

#Num	2	3	4	5	6
Accuracy	0.7157	0.7226	0.7291	0.7260	0.7248

existing training data that was generated by another LLM. Then, this pre-existing data is input into the target LLM. **In this setting, the input to the classifier is the internal state of the LLM when encoding the pre-existing texts generated by other LLMs.** Table 4 presents the performance of hallucination detection models for OPT-7B and LLaMA-7B-Chat.

The experimental results indicate a clear trend: customized training data significantly enhances the performance of hallucination classifiers for both OPT-7B and LLaMA2-Chat-7B models. This observation supports the hypothesis that training data tailored to the specific language model can improve its ability to identify hallucinations in generated text. This implies that for a new model, it would be advantageous to customize the hallucination detection model using the MIND method tailored to that specific model.

6.3.2 Impact of the Training Data Size

The training data for our proposed MIND model is generated automatically, making it crucial to determine the optimal amount of data to produce. We conduct a systematic investigation into the impact of training dataset size on model performance which is detailed in Table 5. The experimental result shows that increasing the training dataset size from 1,024 to 4,096 labels improves accuracy, suggesting that larger datasets enable the model to learn more complex patterns. However, the benefit plateaus and even slightly decreases after exceeding 4,096 data points, indicating a threshold beyond which additional data no longer improves performance significantly.

6.3.3 Classifier Layer Depth

In this section, we explore the impact of varying the depth of the classifier layer on the performance of the MIND model. The depth, measured in terms of the number of layers in the classifier, ranged from 2 to 6 layers. The accuracy of the MIND classifier on the development set was used as the metric for assessing performance. The results of this experiment are shown in Table 6. Initially, as the number of layers increased from 2 to 4, there was a slight improvement in accuracy. Specifically, the model's accuracy went from 0.7157 with two layers to a peak of 0.7291 with four layers. However, the small difference between different numbers of layers indicates that the MIND classifier is not sensitive to this hyperparameter of the number of layers.

7 Related Work

7.1 Hallucination Detection

To tackle the issue of hallucinations in LLMs, researchers have devised various methods for hallucination detection. SelfCheckGPT (Manakul et al., 2023) (SCG) is designed based on the principle that if an LLM has knowledge of a given concept, sampled responses are likely to be similar and contain consistent facts. HaluEval (Li et al., 2023d) represents a direct approach, where strong LLMs like GPT4 are directly used to evaluate the output of other LLMs. SAPLMA (Azaria and Mitchell, 2023) introduces human annotations to label hallucinations ChatGPT outputs, then training a classifier that detects hallucinations in LLMs by analyzing their internal states. EU-HD (Zhang et al., 2023b). Enhanced Uncertainty-Based Hallucination Detection is a SOTA hallucination detection method based on the predictive probability and LLM's attention for each generated tokens.

7.2 Evaluation of Hallucination Detection

The purpose of the Hallucination Detection Evaluation (HDE) is to evaluate the effectiveness of various Hallucination Detection Methods (HDMs). The SelfCheckGPT dataset (Manakul et al., 2023) utilizes GPT-3 to generate passages about individuals from the WikiBio dataset, with manual annotations assessing the factuality of these passages, classifying them into major inaccurate, minor inaccurate, and accurate categories. The True-False dataset (Azaria and Mitchell, 2023) represents another innovative approach, constructed based on a database of instances with multiple factual attributes. HaluEval (Li et al., 2023d) takes a different approach by prompting GPT models to generate hallucinatory texts, using prompts like, "I want you

to act as a hallucination answer generator," coupled with human annotation. HADES (Liu et al., 2021) employs a rule-based method to modify tokens in Wikipedia articles to generate hallucination texts.

8 Conclusions and Future Works

In this paper, we introduce MIND, a novel unsupervised approach leveraging the internal states of Large Language Models (LLMs) for real-time hallucination detection. Moreover, we propose HELM, a comprehensive benchmark for hallucination detection, incorporating outputs from six diverse LLMs along with their internal states during text generation.

9 Limitations

We acknowledge the limitations of this paper, particularly in the aspect of detecting hallucinations using only the internal states of LLMs. While effective, this method has the potential for enhanced accuracy. To address this, our future work will focus on integrating the internal states of LLMs with their generated text. This combined approach aims to improve the precision and reliability in identifying and mitigating hallucinations in LLM outputs, leading to more robust and accurate hallucination detection methodologies.

10 Ethics Statement

In conducting this research, we have prioritized ethical considerations at every stage to ensure the responsible development and application of AI technologies. In the development of MIND, our approach has been to create a reference-free, unsupervised training framework that primarily utilizes publicly available data sources, such as Wikipedia. This methodological choice ensures that our research does not rely on personally identifiable information. We firmly believe in the principles of open research and the scientific value of reproducibility. To this end, we have made all models, data, and code associated with our paper publicly available on GitHub. This transparency not only facilitates the verification of our findings by the community but also encourages the application of our methods in other contexts.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jia Chen, Haitao Li, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. Thuir at wsdm cup 2023 task 1: Unbiased learning to rank. *arXiv preprint arXiv:2304.12650*.

Xuesong Chen, Ziyi Ye, Xiaohui Xie, Yiqun Liu, Xiaorong Gao, Weihang Su, Shuqi Zhu, Yike Sun, Min Zhang, and Shaoping Ma. 2022. Web search via an efficient and effective brain-machine interface. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1569–1572.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67.

Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. 2024. Scaling laws for dense retrieval. *arXiv preprint arXiv:2403.18684*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Zhengbao Jiang, Luyu Gao, Jun Araki, Haibo Ding, Zhiruo Wang, Jamie Callan, and Graham Neubig. 2022. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. *arXiv preprint arXiv:2212.02027*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Haitao Li, Jia Chen, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023a. Towards better web search performance: Pre-training, fine-tuning and learning to rank. *arXiv preprint arXiv:2303.04710*.

Haitao Li, Weihang Su, Changyue Wang, Yueyue Wu, Qingyao Ai, and Yiqun Liu. 2023b. Thuir@ coliee 2023: Incorporating structural knowledge into pre-trained language models for legal case retrieval. *arXiv preprint arXiv:2305.06812*.

Haitao Li, Changyue Wang, Weihang Su, Yueyue Wu, Qingyao Ai, and Yiqun Liu. 2023c. Thuir@ coliee 2023: More parameters and legal knowledge for legal case entailment. *arXiv preprint arXiv:2305.06817*.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023d. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. 2021. A token-level reference-free hallucination detection benchmark for free-form text generation. *arXiv preprint arXiv:2104.08704*.

Yixiao Ma, Yueyue Wu, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. Caseencoder: A knowledge-enhanced pre-trained model for legal case encoding. *arXiv preprint arXiv:2305.05393*.

Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.

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

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Hemlata Shelar, Gagandeep Kaur, Neha Heda, and Poorva Agrawal. 2020. Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3):324–337.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Weihang Su, Qingyao Ai, Xiangsheng Li, Jia Chen, Yiqun Liu, Xiaolong Wu, and Shengluan Hou. 2023a. Wikiformer: Pre-training with structured information of wikipedia for ad-hoc retrieval. *arXiv preprint arXiv:2312.10661*.

Weihang Su, Qingyao Ai, Yueyue Wu, Yixiao Ma, Haitao Li, and Yiqun Liu. 2023b. Caseformer: Pre-training for legal case retrieval. *arXiv preprint arXiv:2311.00333*.

Weihang Su, Xiangsheng Li, Yiqun Liu, Min Zhang, and Shaoping Ma. 2023c. Thuir2 at ntcir-16 session search (ss) task. *arXiv preprint arXiv:2307.00250*.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. *arXiv preprint arXiv:2403.10081*.

Amogh Kamat Tarcar, Aashis Tiwari, Vineet Naique Dhaimodker, Penjo Rebelo, Rahul Desai, and Dattaraj Rao. 2019. Healthcare ner models using language model pretraining. *arXiv preprint arXiv:1910.11241*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Ziyi Ye, Xiaohui Xie, Qingyao Ai, Yiqun Liu, Zhihong Wang, Weihang Su, and Min Zhang. 2024. Relevance feedback with brain signals. *ACM Transactions on Information Systems*, 42(4):1–37.

Hengran Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2023a. From relevance to utility: Evidence retrieval with feedback for fact verification. *arXiv preprint arXiv:2310.11675*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. 2023b. Enhancing uncertainty-based hallucination detection with stronger focus. *arXiv preprint arXiv:2311.13230*.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Paco Guzman, Luke Zettlemoyer, and Marjan Ghazvininejad. 2020. Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593*.

A Pseudocode Description of MIND

The pseudocode description of our proposed unsupervised training data generation process is shown in Algorithm 1.

B Prompt Template of the HELM Dataset

For the data generation process of our proposed HELM dataset, the selected LLMs were tasked with free-form generation. Specifically, the task involved prompt-based continuation writing. For base LLMs and chat LLMs[5], we have designed different prompt templates respectively.

For the base LLM, The prompt template is as follows:

Prompt 1

This is a Wikipedia passage about **[title]**. **[First sentence of that article]**.

For the chat LLM, The prompt template is as follows:

Prompt 2

The following sentence is the first sentence of a Wikipedia article titled **[Title]**. Please continue writing the sentence below. **[First sentence of that article]**

C Prompt Template of GPT4

Following the settings of Li et al. (Li et al., 2023d), we directly use GPT-4 as hallucination detection model to judge whether the text generated by other LLMs contains hallucination through the following prompt template:

Prompt 3

Given the following text span, your objective is to determine if the provided text contains non-factual or hallucinated information. You SHOULD give your judgment based on the world knowledge.
Text span: **[Provided Text]**
Now, determine if the above text span contains non-factual or hallucinated information. The answer you give MUST be "Yes" or "No".

D Details of Our Selected LLMs

To validate the effectiveness of the approach utilizing the internal states of Large Language Models

[5]Base LLMs are language model that has been pre-trained on a large corpus. On the other hand, the Chat LLM, besides pre-training, also undergoes additional processes such as instruction tuning to better align with conversational tasks.

Algorithm 1 Unsupervised Training Data Generation for Hallucination Detection

Input: Language Model L_i, Wikipedia Articles $W = \{w_1, w_2, ..., w_n\}$
Output: Data Tuples $D = \{D_1, D_2, ..., D_n\}$
for each article $w_i \in W$ **do**
    Select random entity e_i not at the beginning of any sentence
    $w_i^{'} \leftarrow \text{truncate}(w_i, e_i)$ ▷ Truncate w_i at e_i
    $G_i \leftarrow L_i(w_i^{'})$ ▷ Generate continuation with L_i
    Truncate G_i at the end of its first sentence
    Record internal states S_i during generation
    if the beginning of G_i contains e_i correctly **then**
        $H_i \leftarrow 0$ ▷ Non-hallucination
    else
        $H_i \leftarrow 1$ ▷ Hallucination
    end if
    $D_i \leftarrow (L_i, w_i, G_i, S_i, H_i)$ ▷ Form data tuple
end for

(LLMs) for hallucination detection across various existing LLMs, we conducted experiments with as many open-source LLMs as possible. Specifically, this included the following 14 LLMs:

- **GPT-J-6B** (Wang and Komatsuzaki, 2021) is a 6 billion parameter, autoregressive text generation model trained on The Pile corpus (Gao et al., 2020).
- **OPT-6.7B** (Zhang et al., 2022). OPT is a collection of decoder-only pre-trained Transformers, with models ranging from 125 million to 175 billion parameters. We choose **OPT-6.7B** from the OPT series.
- **LLaMA-2** (Touvron et al., 2023b) is a collection of pre-trained and fine-tuned LLMs ranging in scale from 7 billion to 70 billion parameters. This series includes fine-tuned LLMs, known as Llama 2-Chat, specifically designed for optimal performance in dialogue-based applications. We choose **LLaMA-2-Chat-7B**, **LLaMA-2-Base-7B**, **LLaMA-2-Base-13B**, and **LLaMA-2-Chat-13**.
- **Falcon** (Almazrouei et al., 2023) comprises a set of causal decoder-only models that have been trained on a dataset of 1,000 billion tokens, which includes data from RefinedWeb (Penedo et al., 2023). Among the models in Falcon, we have chosen **Falcon-7B** and **Falcon-40B**.

E Implementation Details

- **NER**: For the Named Entity Recognition (NER) component of MIND, we follow the methodologies in prior studies (Liu et al., 2021; Tarcar et al., 2019). Specifically, we utilized the Spacy library, a tool recognized for its effectiveness and efficiency in NER as evidenced by previous research (Shelar et al., 2020).

- **MIND**: The MIND classifiers employs a 4-layer Multilayer Perceptron (MLP) network, featuring a 20% dropout rate applied at the initial layer. The architecture of this network is characterized by a progressively decreasing hidden layer size, with dimensions set at 256, 128, 64, and 2 for each consecutive layer. In terms of activation functions, the Rectified Linear Unit (ReLU) was selected. The learning rate is set to 5e-4, the weight decay is set to 1e-5, and the training batch size is set to 32. For the ablation experiments of important hyperparameters, we have discussed in detail in Section 6.3.

- **SCG**: For the implementation of SelfCheckGPT (SCG), we directly use the code and follow all the hyperparameters from their official GitHub[6].

- **LLM Configuration**: For the selected LLMs, we directly download model parameters from the official Hugging Face repositories for each model, and use the code provided by Hugging Face to conduct text generation. For the generation configuration, we use the official default configurations provided by each model.

[6]https://github.com/potsawee/selfcheckgpt/tree/main

F Discussion on Mitigating Hallucination

This paper focuses on hallucination detection and the evaluation of hallucination detection methods. However, in practical applications, it is indeed feasible to use our proposed MIND Framework to mitigate hallucinations. Therefore, we briefly discuss two potential methods to mitigate hallucinations based on the MIND Framework, offering some guidance for those in need.

Firstly, the MIND Classifier can be employed as a re-ranker. The MIND Classifier outputs a score indicating the likelihood of a hallucination occurring in LLMs' response. Thus, by utilizing a sampling decoding method, an LLM can generate multiple outputs for the same query. Subsequently, the MIND Classifier can assess these outputs for hallucinations, allowing the selection of the output with the lowest probability of hallucination as the final response.

Additionally, we can employ the dynamic retrieval augmented generation (RAG) framework (Khandelwal et al., 2019; Borgeaud et al., 2022; Lewis et al., 2020; Guu et al., 2020; Izacard and Grave, 2020; Jiang et al., 2022; Li et al., 2023a; Shi et al., 2023; Su et al., 2023b; Chen et al., 2023, 2022; Li et al., 2023c; Su et al., 2024; Fang et al., 2024; Zhang et al., 2023a) to mitigate hallucinations. The dynamic RAG method triggers the retrieval module (Li et al., 2023b; Su et al., 2023a; Ma et al., 2023; Ye et al., 2024; Su et al., 2023c) during the inference process of an LLM, facilitating the retrieval of external knowledge. The MIND Framework can provide guidance on when to trigger the RAG. Specifically, when MIND indicates a high probability of hallucination, the retrieval module can be triggered and then add the retrieved passages into the context of the LLM, allowing the LLM to continue generating based on the retrieved external knowledge.