Knowing What LLMs DO NOT Know: A Simple Yet Effective Self-Detection Method

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown great potential in Natural Language Processing (NLP) tasks. However, recent literature reveals that LLMs hallucinate intermittently, which impedes their reliability for further utilization. In this paper, we propose a novel self-detection method to detect which questions an LLM does not know. Our proposal is empirical and applicable for continually upgrading LLMs compared with state-of-the-art methods. Specifically, we examine the divergence of the LLM's behaviors on different verbalizations for a question and examine the atypicality of the verbalized input. We combine the two components to identify whether the model generates a non-factual response to the question. The above components can be accomplished by utilizing the LLM itself without referring to any other external resources. We conduct comprehensive experiments and demonstrate the effectiveness of our method for recently released LLMs involving Llama 2, Vicuna, ChatGPT, and GPT-4 across factoid question-answering, arithmetic reasoning, and commonsense reasoning tasks.

1 Introduction

With the significant improvements in large language models (LLMs) such as PaLM (Chowdhery et al., 2022), ChatGPT (Ouyang et al., 2022), GPT-4 (OpenAI, 2023), LLAMA 2 (Touvron et al., 2023), and Vicuna (Chiang et al., 2023), LLMs have been applied in various natural language tasks. Unfortunately, LLMs still produce unexpected falsehoods (Bang et al., 2023; Li et al., 2023), i.e., they are unaware of what they do not know and generate responses indiscriminately. For example, ChatGPT generates falsehoods for a knowledge quiz and math problem, as shown in Table 1. These intermittent errors can severely hinder the LLMs' reliability in practice, which makes detecting what they do not know an important research problem (Hendrycks et al., 2021; Lin et al., 2022; Kadavath et al., 2022).

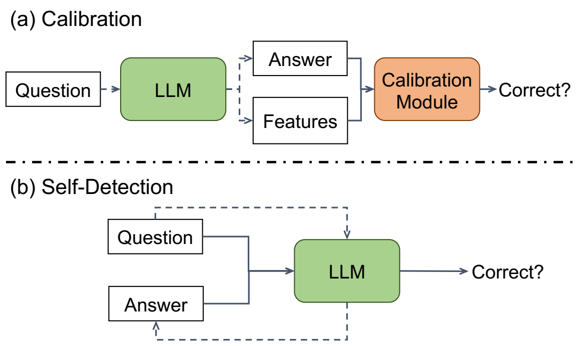


Figure 1: Two paradigms for detecting hallucinations. The dashed lines denote the LLM generation process. The solid lines denote non-factuality detection.

There are two main paradigms to detect non-factuality: the calibration-based methods and the self-detection methods. The first class of methods calibrates the model confidence to better detect falsehoods of the generations (See Figure 1(a)). Among them, Mielke et al. (2022) train auxiliary calibrators, Lin et al. (2022) and Jiang et al. (2021) improve the calibration through fine-tuning the language model. We propose a self-detection method that does not require further fine-tuning.

The self-detection methods directly leverage the LLMs themselves to detect whether they hallucinate (See Figure 1(b)). For example, Kadavath et al. (2022) prompt the LLMs to predict the confidence score on whether their responses are true, and Si et al. (2023) directly utilizes the token probabilities of the generations as the confidence score; Wang et al. (2023) and Manakul et al. (2023) detect the falsehoods by sampling answers with a high temperature and examining self-consistency between them. However, the performance of these works is limited as LLMs tend to be overconfident about their own outputs and these work would be less effective after the models are trained more aligned (OpenAI, 2023; Ouyang et al., 2022; Zhao et al., 2023).

Knowledge Quiz: The lyricist of *Kadam Kadam Badhaye Ja Khushi ke Geet Gaye Ja*	
Q1: Who has written the lyrics to patriotic song kadam kadam badhaye ja khushi ke geet gaye ja? R1: The lyrics of the patriotic song Kadam Kadam Badhaye Ja Khushi Ke Geet Gaye Ja were written by **Shri Ravi Shankar Sharma**.	Q2: Who is the lyricist of the patriotic song Kadam Kadam Badhaye Ja Khushi ke Geet Gaye Ja? R2: The lyricist of the patriotic song Kadam Kadam Badhaye Ja Khushi ke Geet Gaye Ja is **Shri Pyarelal Santoshi**.
Correct Answer: **Vanshidhar Shukla** .	
Math Problem	
Q1: Tom's restaurant gets 6 reservations a night. They normally order 2 meals that cost \$5 each and a \$5 bottle of wine. How much do they make a week if they are open 2 days a week? R1: They make 6*2=12 meals a night. So they make 12*10 =120 a night. **That means they make 120*2=240 a week.**	Q2: Kanan's restaurant gets 6 reservations a night. They normally order 2 meals that cost \$5 each and a \$5 bottle of wine. How much do they make a week if they are open 2 days a week? R2: They make 2*6=12 meals a night. So they make 12*2=24 on meals. They also make 6*5=30 on wine. So they make 24+30=54 a night. **That means they make 54*2=108 a week.**
Correct Answer: **180** .	

Table 1: Two examples of completely different responses for the different verbalized but semantically equivalent questions.

In this paper, we consider detecting non-factuality as that a model does not know which knowledge is related to the question or does not understand the queried question, outputting the non-factual response. A model is expected to provide correct and consistent answers regardless of the ways the questions are verbalized. Therefore, if it responds drastically differently to the different verbalizations, we consider the model does not know the question.

Built on the above hypothesis, we propose a novel self-detection method that includes 1) examining the divergence of the LLM's behaviors on different verbalized questions and 2) examining whether the verbalization of the question is typical in the LLM as shown in Figure 2. Specifically, for the first component, we first diversify the queried question to several semantically equivalent verbalizations. Then, we examine the divergence between the answers corresponding to the questions. For the second component, we use the negative log-likelihood of the verbalized question as the indicator of atypicality in the language model. Concurrent work (Zhang et al., 2023) has also mentioned rephrasing the original question to alternatives and checking the consistency of the answers with the original answer. In contrast, we further propose to examine the representativeness of the input for the model and examine the divergence in the answer distribution. Our self-detection method is applicable for continually upgrading LLMs.

To verify the effectiveness of our method, we conducted extensive experiments on GPT-4, ChatGPT, Vicuna, and Llama 2 across three types of tasks: factoid question answering, commonsense reasoning, and arithmetic reasoning tasks. The experimental results demonstrate the superior performance of our self-detection method.

In summary, our contributions are as follows:

- We show existing LLMs intermittently retain the verbalization-sensitive problem, generating drastically contradicted responses to the questions with the same semantics but verbalized differently.
- We introduce a self-detection suit that relies solely on an LLM itself, enabling a light detection of whether an LLM is unknown for a question.
- We prob what an LLM knows and does not know and show a correlation between the unknown to the popularity, the reasoning steps, and the formulations.

2 Related Work

Model Calibration Calibration is a well-studied topic in traditional neural networks (Hendrycks and Gimpel, 2017; Guo et al., 2017; Pereyra et al., 2017; Qin et al., 2021), aiming to provide a confidence score that aligns well with the true correctness likelihood. Jagannatha and Yu (2020), Jiang et al. (2021) and Kadavath et al. (2022) show BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), T5 (Raffel et al., 2020), BART (Lewis et al., 2020), GPT-2 (Radford et al., 2019), GPT-3.5 (Ouyang et al., 2022) are not well-calibrated on the language tasks.

Post-hoc methods like temperature scaling and feature-based fitting on a development set are widely used (Guo et al., 2017; Desai and Durrett, 2020; Hendrycks et al., 2019; Jiang et al., 2021), which are straightforward to implement. Bootstrapping and ensembling methods (Osband et al., 2016; Lakshminarayanan et al., 2017; Sun et al.; Radford et al., 2019) are explored for the traditional DNN models. Li et al. (2022); Ye and Durrett (2022); Dong et al. (2022); Yuksekgonul et al. (2023) fine-tune and optimize the calibration for BERT, RoBERTa, T5 and Alpaca respectively. Mielke et al. (2022) and Lin et al. (2022) fine-tune the BlenderBot (Roller et al., 2020) and GPT-3 (Brown et al., 2020) separately for calibration and express the models' uncertainty in a verbalized statement. The calibration tuned for specific tasks makes it challenging to generalize on out-of-distribution data (Guo et al., 2017).

Hallucination Detection LLMs such as ChatGPT (Ouyang et al., 2022), GPT-4 (OpenAI, 2023), Vicuna (Chiang et al., 2023), Llama 2 (Touvron et al., 2023) and Claude (Anthropic, 2023) have obtained remarkable performance on various language tasks (Bang et al., 2023; Rangapur and Wang, 2023). However, recent work (Mallen et al., 2023; Bang et al., 2023; Li et al., 2023; Yin et al., 2023) show that LLMs may produce hallucinated contents, i.e., non-factual responses. The importance of the hallucination problem has been highlighted by several work (Lin et al., 2022; Ji et al., 2023) as it hinders the reliability of the LLMs.

Kadavath et al. (2022) and Agrawal et al. (2023) use LLMs to evaluate the sampled answers but can not evaluate their self-generated answers due to overconfidence. Si et al. (2023) and Manakul et al. (2023) utilize their confidence scores like token probability to indicate the confidence of their output. Recent work (Wang et al., 2023; Si et al., 2023; Mündler et al., 2023; Kuhn et al., 2023) examines the self-consistency score among the randomly sampled answers which are generated through a higher temperature. Both the confidence score of the model output and sample-based score highly rely on the current model training, which means the methods would not be that effective after the models are trained to be more aligned.

Xiong et al. (2023) combine the LLMs verbalized statement, self-consistency of the randomly sampled answers, and the consistency between the induced answers. This work proposes to add additional instruction to the prompt for generating induced answers. Concurrent work (Zhang et al., 2023; Cohen et al., 2023) utilizes several verifier LLMs to cross-check whether a language model generates falsehoods. Zhang et al. (2023) also rephrases the original question to alternative inputs and checks the consistency of the answers with the original answer as the confidence score. We propose a unified method that examines the divergence of the LLMs' behaviors across the diversified questions besides the consistency pair and the atypicality of the verbalized input in the LLMs. Our proposal is self-detection without referring to any other LLMs or external resources.

3 Inconsistency and Atypicality in LLMs

We attribute the non-factuality of an LLM to the generative characteristics which sample the most possible tokens sequentially. It means even if the LLM does not know the exact knowledge related to the question or even does not understand the question, it still generates plausible responses as observed in previous work (Cao et al., 2021; Zhuo et al., 2023).

Consequently, if an LLM returns contradicted responses to the semantically equivalent questions, the LLM does not know the question generating non-factual answers. Besides, if the textual verbalization of a question is not representative for the LLM, i.e., atypical, it would be hard to understand resulting in a lower-quality response (Yuksekgonul et al., 2023). Two examples of ChatGPT are shown in Table 1, where the Q1 and Q2 describe the same question with different verbalizations, but their answers are completely different.

So, we 1) examine the divergence between the responses ($R = \{r_1, ..., r_n\}$) to a question set ($Q = \{q_1, ..., q_n\}$), where any two questions q_i and q_j are semantically equivalent; 2) then examine whether the verbalized question q is representative in the LLM using the atypicality $A(q)$ of the input.

4 Self-Detecting What LLMs Un-Know

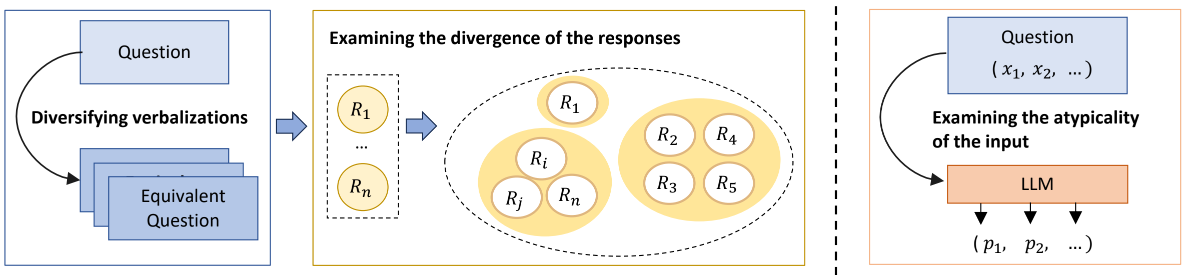
In this section, we introduce our framework including consistency-based detection 4.1 and verbalization-based detection 4.2 as shown in Figure 2.

4.1 Consistence-based Detection

Given a question, we first diversify the original question to several questions (Section 4.1.1). Then, we examine the consistency among the generated responses corresponding to the diversified questions (Section 4.1.2).

4.1.1 Diversifying Question Verbalizations

We diversify question q to several textual verbalizations $Q(q) = \{q_1, ..., q_n\}$ that express the same



(a) Consistency-based Component (b) Verbalization-based Component

Figure 2: The framework of self-detecting what language models un-know.

meaning.

Model-based Generation For those open QA questions, we exploit a LLMs itself (eg., ChatGPT, Vicuna) to generate paraphrased questions through the prompt: `Given the following question [QUESTION], paraphrase it to have different words and expressions but is semantically equivalent.` The unbroken instruction for the task is shown in Table 8 in Appendix A.1.

After obtaining the paraphrased questions, we filter out the unsatisfied ones by prompting the language model to detect whether two questions are semantically equivalent and the instruction is shown in Table 9.

Rule-based Generation For commonsense reasoning and arithmetic reasoning questions, we use expert-defined rules for diversification, as those questions are sensitive to numerical numbers, modifiers, and logical relationships. We exchange the order of choices provided for the question to obtain n paraphrased questions for commonsense reasoning. We substitute the person names of a question with new names to obtain n paraphrased questions for arithmetic reasoning problems, as the second example in Table 1.

4.1.2 Calculating Consistency Score

We examine the consistency among the generated responses $R(q) = \{r_1, ..., r_n\}$ according to the diversified questions $Q(q) = \{q_1, ..., q_n\}$. For generation, we employ the LLM with the greedy decoding strategy to avoid unexpected randomness of the generative model as much as possible.

Consistency Determination Firstly, we examine whether any two answers are consistent $I(r_i, r_j) \in \{0, 1\}$. For these answers with fixed formats like multiple-choice answers, we extract the final answer using regular expressions and check whether the final answer exactly matches (EM) the other one. For these free-form answers, we use the LLM itself to handle the inconsistency detection by asking whether the two answers are the same or contradicted, as shown below. The $I(r_i, r_j)$ is inferred from the generated contents using keywords "Contradicted" or "Same".

```
Determine whether the answer 'A1' is 'Contradicted' or 'Same' with the answer 'A2' for the question 'Q'. You need to check whether the two answers exactly describe the same thing such as the same entity, digit, or arithmetical results. If the two answers are the same, give "Same", otherwise give "Contradicted" as the result.
```

Table 2: The instruction for determining whether two answers are consistent.

This task is a strength of the latest LLMs even in a zero-shot measure as it demands basic logical reasoning abilities (Qin et al., 2023; Liu et al., 2023; Zhong et al., 2023) and we conduct the human evaluation for this component at the experiments.

Consistency Calculation A common way of calculating the consistency score is:

$$Consistency(R(q)) = \frac{1}{n-1} \sum_{r_i, r_i \neq r} I(r_i, r) \tag{1}$$

where r is the response for the original question q.

We further compute the divergency of the response distribution to characterize the uncertainty about the question. Based on consistency, we group the responses into several clusters and obtain a cluster distribution $\Omega = \{\omega_1, ..., \omega_k\}$ for the n responses. Specifically, we perform the following clustering algorithm 1:

After clustering, we calculate the entropy of the answer distribution as another consistency score:

$$Entropy(R(q)) = \sum_{l} \frac{N(\omega_l)}{n} \log \frac{N(\omega_l)}{n} \tag{2}$$

where $N(\omega_l)$ is the number of responses in the cluster ω_l. The entropy measures the degree of divergence between the responses to the same question.

Algorithm 1 Clustering Answers

Input: $R(q), \{I(r_i, r_j)\}$
Output: $\Omega = \{\omega_1, ..., \omega_k\}$
Initialization: $\omega_1 = \{r_o\}$, where r_o is randomly sampled from $R(q)$
for all $r_j \in R(q), r_j \neq r_o$ **do**
  $Clustered = False$
  for all $\omega_l \in \Omega$ **do**
    Randomly draw a response r_i from ω_l
    if $I(r_j, r_i) == 1$ **then**
      $\omega_l \leftarrow \omega_l + r_i, Clustered = True$
      Break
    end if
  end for
  if $Clustered == False$ **then**
    $\omega_{new} = \{r_j\}, \Omega \leftarrow \Omega + \omega_{new}$
  end if
end for

A higher entropy indicates greater randomness in the generations. It corresponds to a lower probability of providing correct answers for the question, which suggests the LLM is less likely to know the question.

4.2 Verbalization-based Detection

We then compute the atypicality of the input. Inspired by (Yuksekgonul et al., 2023), current LLMs are autoregressive models that compute a marginal distribution $P(x)$ as its confidence score. We compute the negative log-likelihood of the verbalized input as the indicator of the atypicality:

$$A(q) = -\log P(q) = -\sum_{t}^{T} \log P(x_t | X_{<t}) \quad (3)$$

where x_t and $X_{<t}$ indicate a token and a token set in the question q. We add a normalized score $A(q)/N(q)$ in this component, where $N(q)$ is the number of tokens in question q. We use $A(q)$ along with its normalized version as the atypicality of the input to quantify whether the verbalized input is representative in the language model. A higher value of $A(q)$ would indicate that the verbalization is more atypical for the language model.

Finally, we combine the two components to predict the final confidence score that the language model does not know the question.

5 Experiments

5.1 Experimental Settings

Datasets We evaluate the effectiveness of our self-detection on factoid question answering, arithmetic reasoning, and commonsense reasoning tasks. For factoid question answering, we use FaVIQ (Park et al., 2022) and ComQA (Abujabal et al., 2019) as our benchmark dataset. For arithmetic reasoning, we use GSM-8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021). For commonsense reasoning, we use ARC-Challenge (Clark et al., 2018) and CommonsenseQA (Talmor et al., 2019). For FaVIQ, we randomly split the a-set into train, dev and test sets, and samples 500, 500, and 200 instances respectively. For other datasets, we use the built-in splits and sample the same number of instances for training, validating and testing.

Models We self-detect the SOTA LLMs including ChatGPT (gpt-3.5-turbo), GPT-4, Vicuna-13B and Llama2-13B. For GPT-series models, we request the openAI APIs[1] to obtain the responses. For Vicuna and Llama 2, we deployed the model ourselves using 2 A100 40G GPUs.

Evaluation Metrics We report PR AUC to measure whether our uncertainty score correlates well with a nonfactual response. For each question in the datasets, we have a golden answer. For factoid question answering tasks, we prompt GPT-4 to verify the correctness of the response by comparing it with the golden answer similar to what we described before. For arithmetic and commonsense reasoning questions, we check whether the final answer exactly matches the golden answer, while the final answer is extracted using regular expressions. If the extraction fails, we prompt GPT-4 to assess whether the answer is correct as we did in the factoid question answering tasks.

Baselines We compare our self-detection with recent SOTA methods including: 1). Token-level probability (TokenProbs for short), proposed in (Manakul et al., 2023), measures the response's likelihood and the average of the token probabilities is used as the confidence score; 2). Perplexity, the reciprocal of the (normalized) language model probability, is used to indicate the uncertainty (Si

[1] `https://platform.openai.com/docs/api-reference`

	ARC	CommonsenseQA	GSM-8K	SVAMP	FaVIQ	ComQA
ChatGPT						
Random	10.78	22.49	11.77	17.94	45.96	27.05
ConsistAnswers	14.24	25.96	52.71	**30.50**	57.09	31.76
SelfCheckGPT	23.60	39.38	21.14	25.68	52.26	39.56
SelfDetection (w/o Atypicality)	**40.86**	**40.23**	**56.29**	28.18	**59.65**	**42.86**
GPT-4						
Random	6.29	9.71	6.91	7.13	37.67	23.02
ConsistAnswers	27.44	35.47	22.39	**25.99**	51.30	37.34
SelfCheckGPT	21.15	39.26	12.99	22.87	46.66	46.31
SelfDetection (w/o Atypicality)	**36.45**	**42.71**	**36.83**	24.78	**56.26**	**58.95**
Vicuna-13B						
Random	35.45	51.15	35.94	54.92	31.56	35.32
TokenProbs	40.66	52.39	39.03	60.00	34.39	59.18
Perplexity	41.27	52.01	37.63	61.60	36.43	59.58
ConsistAnswers	42.69	54.13	43.97	63.28	24.44	50.84
SelfCheckGPT	40.43	54.52	36.49	60.35	18.81	26.52
SelfDetection	**54.55**	**62.93**	**53.31**	**71.19**	**39.45**	**66.97**
SelfDetection (w/o Atypicality)	48.23	59.76	43.24	67.85	30.45	60.93
SelfDetection (w/o Consistency)	48.76	55.37	42.83	60.73	31.95	50.29
Llama2-13B						
Random	64.27	58.93	34.25	57.43	31.44	37.27
TokenProbs	64.10	62.92	35.12	55.73	33.21	43.84
Perplexity	64.08	62.88	35.18	55.87	33.53	44.70
ConsistAnswers	71.17	61.79	47.43	63.84	**59.16**	**65.34**
SelfCheckGPT	69.59	60.95	33.77	59.79	40.69	41.23
SelfDetection	**77.73**	**71.95**	**50.38**	**70.33**	39.83	52.36
SelfDetection (w/o Atypicality)	65.88	65.13	40.80	61.34	41.42	52.42
SelfDetection (w/o Consistency)	70.90	64.00	38.19	62.08	34.19	40.26

Table 3: The PR-AUC of different methods for ChatGPT (gpt3.5-turbo), GPT-4, Vicuna-13B and Llama2-13B on 6 representative datasets of commonsense reasoning, arithmetic reasoning, and question answering tasks. The best results are shown in bold.

et al., 2023); 3). Self-consistency of answers (ConsistAnswers for short) is calculated as the consistency of the sampled answers while the answers are sampled using a high-temperature value (0.7) leading to 10 different predictions (Si et al., 2023); 4). SelfCheckGPT (Manakul et al., 2023) combines the averages of the main response's BERTScore with the most similar sentence of each drawn sample and the token-level probability.

Implementation Details For paraphrasing, we set a high temperature 1.0 to obtain 10 re-phrasings for each question. We incorporate the 10 rephrasings for each question and expand the original training sets and validation sets to 10 times larger. To generate the corresponding answers, we use the default template of each model and employ greedy decoding setting temperature 0.0 to avoid unexpected randomness. This decoding strategy still fits for filtering wrong paraphrases and determining consistency. We employ an XGBoost to fit the four features in the expanded training sets and choose hyperparameters from the expanded dev sets. We report the performance on the six original test sets.

5.2 Overall Performance

In Table 3, we report the overall performance of six methods on ChatGPT, GPT-4, Vicuna-13B, and Llama2-13B across six datasets. Since we cannot obtain the token probabilities for ChatGPT and GPT4, we omit perplexity and token probability methods and only report the performance of Self-Detection without atypicality. The random method randomly assigns a score between 0 and 1 denoting whether the generation is nonfactual serving as the lowest baseline for comparison. The PR-AUC values across different models are not comparable. This is because the ground-truth labels of the four models, whether the models know the answer to a question, are not the same as we report the unknown ratios of each model in Appendix A.2. We compare different methods within the same model.

We see that compared with recent methods, our self-detection method mostly achieves the best performance on the six data sets, validating the effectiveness of our method on different LLMs. Specifically, self-detection shows significant improvements for the commonsense reasoning task on ARC and CommonsenseQA, compared to the previous baselines. In math problems, GSM8k and SVAMP, the self-detection method demonstrates mostly optimal performance, and the consistAnswers serve as a strong baseline. For the two QA datasets FAVIQ and ComQA, the self-detection method performs the best except on Llama 2, and the consistAnswers method serves as a strong baseline.

Overall, our self-detection achieves the best performance because we capture the essence of identifying what a language model knows. If a question is atypical or the answers for a question are unstable, the probability of its response being coincidentally correct aligns with the consistency level of its generated responses and its atypicality.

5.3 Ablation Study

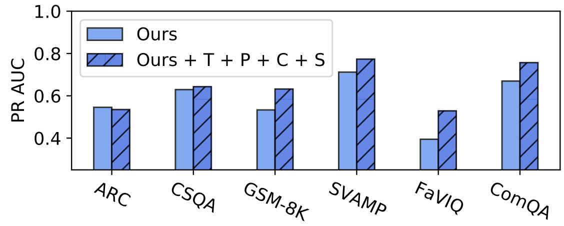
We report the performance of our SelfDetection without atypicality and entropy in Table 3. For Vicuna-13B and Llama2-13B, we see the performance drops when we remove atypicality or entropy indicating the effectiveness of each component. We also see the performance drops greater when we remove entropy compared with atypicality in most experiments, which reveals that the divergence between the answers for diversified questions is more crucial for the SelfDetection method.

Besides, we conduct experiments on combining our method with the previously proposed tokenProbs, perplexity, consistAnswers, and SelfCheckGPT and report the PR AUC on Vicuna-13B and Llama2-13B in Figure 3. We see the performance is continuously improved when combining more signals and our method is comparable in most experiments. We do not report the performance of other combinations of these methods as this is not the focus of this paper.

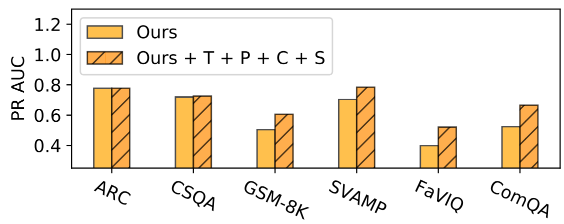
5.4 Unknown Questions Study

Then, we investigate what types of questions the LLMs do not know. We analyze the unknown and known questions of ChatGPT on question answering, arithmetic reasoning, and commonsense reasoning tasks across the six datasets. The known and unknown questions are determined based on the golden correctness label.

Knowledge Popularity We find that the LLM is prone to be ignorant of the atypical knowledge for openQA tasks. For example, when asked about the lyric writer of a less popular song, the model may produce different answers for differently rephrased questions shown in Table 1. Additionally, the most frequent answer is not always the correct one. To further explore the difference between unknown and known questions, we consult search engines including Google and Bing. We use the number of returned search results as an indicator of the popularity of the knowledge for the question. In Table 4, we reveal that the number of search results for unknown questions is significantly lower than for known questions. This suggests that the LLM has relatively poorer memorization of unpopular knowledge.



(a) Comparison on Vicuna-13B



(b) Comparison on Llama2-13B

Figure 3: The PR AUC when combining our method and previous proposed TokenProbs (T), Perplexity (P), ConsistAnswers (C), and SelfCheckGPT (S).

Question Type	Google	Bing
Unknown	7,497k	1,255k
Known	10,929k	2,647k

Table 4: The number of search results for unknown and known questions.

`Tom's restaurant gets 6 reservations a night. They normally order 2 meals that cost $5 each and a $5 bottle of wine. How much do they make a week if they are open 2 days a week?`
`A family wants to adopt for enviro-ethical reasons, what did they abhor?" (A) abandon; (B) foster child; (C) orphan; (D) biological child; (E) give away`

Table 5: Two failed questions for ChatGPT that require longer reasoning steps.

Reasoning Steps For arithmetic reasoning questions, if the solution requires 4 or more reasoning steps, and contains different arithmetic operations simultaneously, the model tends to confuse the order of operations. This leads to incorrect answers. As shown in the first example in Table 5, the model

Question Type	Vicuna-13B	Llama2-13B
Unknown	228.4	202.4
Known	204.0	185.1

Table 6: The negative log-likelihoods for unknown and known questions.

needs to calculate the cost of a reservation first, which includes 2 meals with $5 and a bottle of wine with $5. Then calculate the cost of a night and a week.

For commonsense reasoning tasks, if the solution requires two or more reasoning steps, the model is more likely to make mistakes. As shown in the second example in Table 5, the model needs to reason the subject being concentrated on "adoption" first, and then "enviro-ethical reasons".

Distracted Formulations When distracted formulations appear in a question, the model is prone to generate unexpected errors. We use "distracted" instead of "adversarial" to illustrate that the formulations are not crafted but are built-in, which requires the model to carefully focus on the chain of thought not to be distracted.

`nell collects cards. she had 239 baseball cards and 38 10 cards. she gave some of her cards to jeff and now has 376 10 cards and 111 baseball cards left. how many more 10 cards than baseball cards does nell have?`
`The performer was ready to put on a show and stepped onto the launch platform, what was his job? (A) ocean; (B) battleship; (C) cape canaveral florida; (D) trapeze; (E) nasa`

Table 7: Two questions with distracted formulations.

As shown in Table 7, for the first example, the model needs to be aware that it is unnecessary to calculate how many cards Jeff has but only calculate the number of baseball cards that Neil has more than 10 cards in one reasoning step. For the second commonsense reasoning example, the presence of "Cape Canaveral Florida" is a strong distractor compared to "trapeze" as the question mentions "launch platform".

Besides, we report the negative log-likelihoods averaged across the six datasets of the known question and unknown questions in Vicuna and Llama 2 as the indicator of the atypical input in Table 6. We show that the unknown questions correlate with a higher score, i.e., higher atypicality.

5.5 Impact of Diversified Questions

We examine whether the number of paraphrased questions affects self-detection performance. Due to time and cost constraints, we only report the performance for ChatGPT on three representative datasets (FaVIQ, CommonsenseQA, and GSM8K) corresponding to the three tasks. We report the performance when the number of paraphrased questions is set to 10, 20, and 30. We observe that as the number of paraphrased questions increases, there is a slight improvement, as shown in Figure 4. Our analysis reveals that some unknown questions may be answered coincidentally correctly when the number of questions is small. This inconsistency can be detected as the number of paraphrased questions increases. Additionally, for questions where the model is confident, the model continues to answer consistently, even with more questions. The two phenomena explain the improvement with more questions.

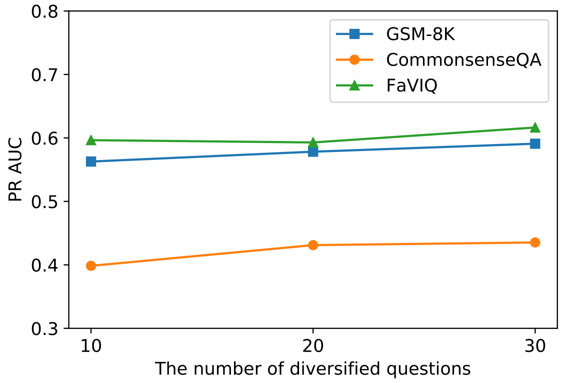


Figure 4: The performance of different numbers of diversified questions for the self-detection.

Finally, we conduct human evaluations on each sub-step of our self-detection in Appendix A.3 and report the costs when we call the OpenAI APIs in Appendix A.4.

6 Conclusion

In this paper, we propose a simple yet effective method to self-detect whether an LLM generates non-factual responses for certain questions, without referring to any other external resources. We conducted extensive experiments on four recent LLMs– ChatGPT, GPT-4, Vicuna, and Llama 2 on three different types of tasks. The experimental results demonstrate the effectiveness of our method. Our method captures the essentials of detecting the LLMs' non-factuality and is applicable for continually upgrading language models. Furthermore, we also explore the question types that LLMs tend to struggle with, like low popularity and distracted formulations. Our method can assist the models to detect and improve their specific weaknesses, improving their reliability in the future.

Limitations

While our method is effective, it still has several limitations. Our self-detection method utilizes a model itself to diversify the verbalizations and thus the diversity is constrained by the LLM's abilities. In the future, we plan to collect more end-user questions from conversational agents or search engines to diversify the original questions to capture the built-in ambiguity of the questions. The ambiguity helps to further detect certain vulnerabilities of the model. Besides, we detect the model's non-factuality through the divergence of the generated answers. It is unable to detect the cases where the model generates consistently but incorrectly, resulting the false negatives. Utilizing additional verifier LLMs or incorporating external knowledge for cross-checking is prevalent and we believe these would help to improve the detection performance. As this is not the focus of our paper, we omit the combinations with them.

Ethics Statement

We ensure that this work does not have explicit ethical considerations such as anonymity and privacy as all the models and datasets we use are public. We are unclear whether the publicly available LLMs may encode problematic bias as it is not the focus of this paper. Our technique is used to detect what LLMs do not know and should not be used in other applications. At least for now, there is no risk of ethics for this method.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2019. Comqa: A community-sourced dataset for complex factoid question answering with paraphrase clusters. In *ACL*, pages 307–317.

Ayush Agrawal, Lester Mackey, and Adam Tauman Kalai. 2023. Do language models know when they're hallucinating references? *arXiv preprint arXiv:2305.18248*.

Anthropic. 2023. Model card and evaluations for claude models.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

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. In *NeurIPS*, pages 1877–1901.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *ACL*, pages 1860–1874.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. 2023. Lm vs lm: Detecting factual errors via cross examination. *arXiv preprint arXiv:2305.13281*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *EMNLP*, pages 295–302.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of EMNLP*, pages 5937–5947.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *ICML*, pages 1321–1330.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. 2019. Using pre-training can improve model robustness and uncertainty. In *ICML*, pages 2712–2721.

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *ACL*, pages 2078–2092.

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, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *TACL*, 9:962–977.

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*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *ICLR*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Dongfang Li, Baotian Hu, and Qingcai Chen. 2022. Calibration meets explanation: A simple and effective approach for model confidence estimates. In *EMNLP*, pages 2775–2784.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *EMNLP*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *TMLR*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2023. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In *ACL*, pages 9802–9822.

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

Sabrina J Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. 2022. Reducing conversational agents' overconfidence through linguistic calibration. *TACL*, 10:857–872.

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

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *NeurIPS*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*, pages 27730–27744.

Jungsoo Park, Sewon Min, Jaewoo Kang, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. Faviq: Fact verification from information-seeking questions. In *ACL*, pages 5154–5166.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *NAACL*, pages 2080–2094.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *ICLR*.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Yao Qin, Xuezhi Wang, Alex Beutel, and Ed Chi. 2021. Improving calibration through the relationship with adversarial robustness. In *NIPS*, pages 14358–14369.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67.

Aman Rangapur and Haoran Wang. 2023. Chatgpt-crawler: Find out if chatgpt really knows what it's talking about. *arXiv preprint arXiv:2304.03325*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. Prompting gpt-3 to be reliable. In *ICLR*.

Meiqi Sun, Wilson Yan, Pieter Abbeel, and Igor Mordatch. Quantifying uncertainty in foundation models via ensembles. In *NeurIPS 2022 Workshop on Robustness in Sequence Modeling*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL*, pages 4149–4158.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hoo. 2023. Can llms express their uncertainty? an empirical evaluation of confidence elici. *arXiv preprint arXiv:2306.13063*.

Xi Ye and Greg Durrett. 2022. Can explanations be useful for calibrating black box models? In *ACL*, pages 6199–6212.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*.

Mert Yuksekgonul, Linjun Zhang, James Zou, and Carlos Guestrin. 2023. Beyond confidence: Reliable models should also consider atypicality. In *NeurIPS*.

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley A Malin, and Sricharan Kumar. 2023. Sac3: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency. *arXiv preprint arXiv:2311.01740*.

Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J Liu. 2023. Calibrating sequence likelihood improves conditional language generation. In *ICLR*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*.

Terry Yue Zhuo, Yujin Huang, Chunyang Chen, and Zhenchang Xing. 2023. Red teaming ChatGPT via Jailbreaking: Bias, Robustness, Reliability and Toxicity.

A Appendix

A.1 Prompts

We show the instruction to diversify the question verbalizations in Table 8. The instruction for detecting wrong paraphrases of the question in Table 9.

```
Given a question, paraphrase it to have
different words and expressions but have the
same meaning as the original question. Please
note that you should not answer the question,
but rather provide a re-phrased question.
```

Table 8: The instruction for the paraphrasing task.

```
Determine whether the paraphrased question
describes the same thing as the original
question, and give "Contradicted" if they are
not the same otherwise give "Same" as the
result.
```

Table 9: The instruction for detecting wrong paraphrases.

A.2 Evolution of LLMs

We report the ratios of unknown questions for the continually upgrading models across the openQA, commonsense reasoning, and arithmetic reasoning tasks, where the unknown and known questions are determined by the golden correctness labels. As shown in Table 10. We see that GPT-4 performs the best and ChtGPT is weaker. Vicuna-13B and Lllam2-13B perform closely and both of them are weaker than the GPT series in terms of all tasks.

A.3 Component Evaluation

We analyze the precision of each component in our framework. For the first paraphrase module, we randomly sampled 100 paraphrases generated from the four LLMs. Then we manually label whether the rephrased versions describe the same thing as the original questions. We report the human-labeled agreement ratio upon the 100 instances as the paraphrase precision.

The precision for the commonsense reasoning tasks is 100% as we only exchange the options as the paraphrased version. In arithmetic reasoning tasks, the precision is 99% as we only exchange the subjects of the question for a paraphrased version, with the remaining 1% errors due to the conflicts between animal names and human names. For openQA questions, the precisions for ChatGPT, GPT-4, Vicuna-13B, and Llama2-13B are 95%, 95%, 93%, and 93% respectively.

Then, we evaluate the answer clustering performance directly and omit evaluating the consistency

Dataset	ChatGPT	GPT4	Vicuna	Llama 2
ARC	0.10	0.05	0.57	0.36
CSQA	0.19	0.13	0.47	0.34
GSM8k	0.11	0.05	0.64	0.65
SVAMP	0.15	0.07	0.44	0.43
FaVIQ	0.43	0.32	0.67	0.67
ComQA	0.30	0.27	0.44	0.42

Table 10: Comparison of the ratios of unknown questions for different LLMs. CSQA is commonsenseQA for short.

Methods	QA	CSQA	Arith.
ChatGPT (gpt-3.5-turbo)			
TP & PRL	0.00008	0.0002	0.00006
SCGPT & CA	0.002	0.004	0.0006
SelfDetect	0.004	0.004	0.0006
GPT-4			
TP & PRL	0.0024	0.0068	0.0014
SCGPT & CA	0.046	0.105	0.014
SelfDetect	0.092	0.106	0.014

Table 11: The costs per question for the TokenProbs (TP), Perplexity(PRL), ConsistAnswers (CA), SelfCheckGPT (SCGPT) and SelfDetection methods on OpenQA (QA), CommonsenseQA (CSQA) and arithmetical reasoning (Arith.) tasks.

detection performance, as we group the answers solely based on whether the two answers are consistent. The precision is measured by calculating the proportion of answer-pairs in the intersection correctly assigned between the output cluster $\Omega = \{\omega_1, \ldots, \omega_k\}$ and the ground-truth cluster $\mathcal{C} = \{c_1, \ldots, c_p\}$. We report the clustering precision in our manually labeled 400 clusters.

$$\text{Precision}(\mathcal{C}, \Omega) = \frac{1}{k} \sum_{i=1}^{k} \frac{\binom{\max_j |\omega_j \cap c_i|}{2}}{\binom{|c_i|}{2}},$$

We achieved 100% precision for the commonsense reasoning task for the four LLMs. For openQA questions, we achieve precisions of 89%, 90%, 83%, and 81% for ChatGPT, GPT-4, Vicuna-13B and Llama2-13B respectively. For arithmetic reasoning tasks, the precision scores are 92%, 93%, 89%, and 88% for ChatGPT, GPT-4, Vicuna-13B and Llama2-13B respectively.

A.4 Costs

We report the costs for our self-detection and the compared methods. For open-source models like Vicuna, we deploy them ourselves for inference. For those close-sourced like ChatGPT, we request APIs. The costs per question in U.S. dollars across different tasks are shown in Table 11.