# Light-Weight Hallucination Detection using Contrastive Learning for Conditional Text Generation

**Miyu Yamada and Yuki Arase**
Institute of Science Tokyo
yamada.m.ee1b@m.isct.ac.jp, arase@c.titech.ac.jp

## Abstract

We propose a simple and light-weight, yet effective hallucination detection method for conditional text generation. Hallucinated outputs include information that is either absent from and/or difficult to infer from the input context. Leveraging this feature, we add contrastive learning to the hallucination detection classifier to pull faithful outputs and input contexts together while pushing hallucinated outputs apart. Experimental results confirm that our method on top of RoBERTa improves binary hallucination detection performance, outperforming much larger GPT-4o prompting. Remarkably, our method shows higher performance for outputs where hallucinated spans are sparse.

## 1 Introduction

Large Language Models (LLMs) are currently used in a wide range of text generation tasks. However, their outputs often include information that deviates from the facts described in the input or information that cannot be easily verified based on the input (Kaddour et al., 2023), which we define as *hallucination* in this study. Users unintentionally accept hallucinated content as factual, leading to the potential spread of misinformation. To enable safer use of LLMs, it is essential to develop accurate hallucination detection methods. In addition, such detection methods are desired to be computationally efficient given the sheer volume of texts being generated by LLMs.

Various methods have been proposed for hallucination detection. A popular approach employs the hidden states of LLMs to identify irregular internal states due to hallucinated content (Jiang et al., 2024). While promising, this approach only applies to the scenario where we can access the LLMs which have generated the outputs.

Another series of studies targets the scenario where we cannot access nor know the LLM that has generated the outputs. SelfCheckGPT (Manakul et al., 2023) compares multiple outputs from the same LLM to identify inconsistencies among the outputs as clues of hallucination. Due to the design, SelfCheckGPT requires multiple outputs for the same input to detect hallucination. Mishra et al. (2024) uses the Retrieval-Augmented Generation (RAG) to retrieve relevant documents and provide them to the model for verification. FActScore (Min et al., 2023) decomposes generated outputs into a sequence of atomic facts and calculates the percentage of these facts that are supported by an external knowledge base. However, such an external knowledge base is not always available, particularly for individual or less common topics. Furthermore, these methods can be costly because of the use of LLMs as base models. The decoder-based architecture also makes the detection process slower.

There have also been methods specialized for conditional text generation. For example, in the summarization task, QAFactEval (Fabbri et al., 2022) evaluates factual consistency by first generating questions from the summary, then comparing the answers obtained from the summary with those obtained from the original input document. If their answers are different, the output is judged as hallucinated. DAE (Goyal and Durrett, 2020) conducts dependency parsing and then uses natural language inference to determine whether each of these relations is entailed by the input. These approaches can capture more fine-grained inconsistencies by reasoning over intermediate representations like questions or dependency arcs. However, they require additional preprocessing steps such as question generation and dependency parsing.

To address these challenges, we propose a light-weight hallucination detection method for conditional text generation. Hallucinated outputs often contain information that either clearly contradicts the input, lacks support from the input, or consists of unverifiable or subjective statements. Based on

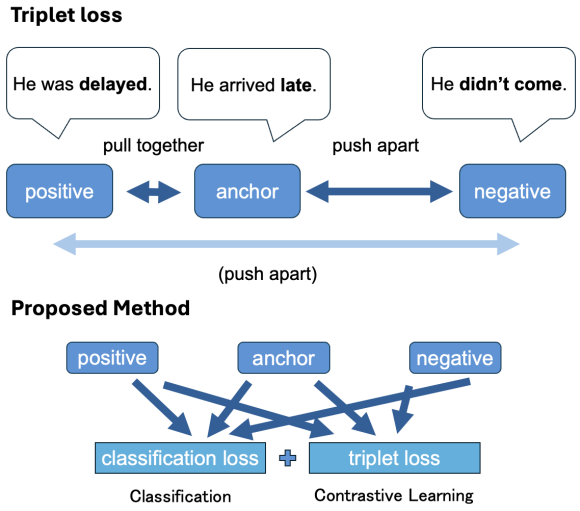


Figure 1: Overview of the proposed method

this feature, we employ contrastive learning (Gao et al., 2021) to a binary classification model using an encoder-based pre-trained model. We train the detector using a triplet loss that pulls faithful generation and the input together while pushes hallucinated generation and the input apart. This should make faithful and hallucinated outputs more distinctive, which may ease the classification.

Experimental results demonstrate that our method outperforms GPT-4o prompting on hallucination detection, achieving 67 times faster computation. Remarkably, our method performs well even when the number and/or proportion of hallucinations in the generation are small. Our code is available at https://github.com/miyu-y/LightHalluDetecter.

## 2 Proposed Method

We formulate hallucination detection for conditional text generation as a binary classification: determining whether a given text contains hallucinations referring to the input context. The proposed method incorporates contrastive learning (the upper part of Figure 1) using the triplet loss computed with an anchor $a$ as input context, a positive sample $g_p$ as faithful generation, and a negative sample $g_n$ as hallucinated generation.

$$\begin{aligned}&\text{triplet}(\boldsymbol{e}_a, \boldsymbol{e}_{g_p}, \boldsymbol{e}_{g_n}) \\ &= \max\left(0, \alpha + d(\boldsymbol{e}_a, \boldsymbol{e}_{g_p}) - d(\boldsymbol{e}_a, \boldsymbol{e}_{g_n})\right), \quad (1)\end{aligned}$$

where $\boldsymbol{e}_a, \boldsymbol{e}_{g_p}, \boldsymbol{e}_{g_n}$ are embeddings of $a$, $g_p$, and $g_n$, respectively, and the hyperparameter $\alpha$ is the margin. The distance function $d(\boldsymbol{x}, \boldsymbol{y})$ we used is the cosine distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = 1 - \text{cossim}(\boldsymbol{x}, \boldsymbol{y}), \quad (2)$$

where $\text{cossim}(\boldsymbol{x}, \boldsymbol{y})$ computes cosine similarity.

We combine the triplet loss with a classification objective (the bottom part of Figure 1). While the triplet loss guides the model to learn embedding that make hallucinated and faithful outputs distinctive, a classification head is simultaneously trained to predict whether a given output contains hallucination. The total loss is defined as:

$$\mathcal{L}_\theta = \text{triplet}(\boldsymbol{e}_a, \boldsymbol{e}_{g_p}, \boldsymbol{e}_{g_n}) + \text{CE}(\boldsymbol{e}_a \oplus \boldsymbol{e}_g). \quad (3)$$

The function $\text{CE}(\boldsymbol{e}_a \oplus \boldsymbol{e}_g)$ is the cross-entropy loss for the binary classification, where the embedding of input context $\boldsymbol{e}_a$ is concatenated with that of generated output, i.e., either $\boldsymbol{e}_{g_p}$ or $\boldsymbol{e}_{g_n}$. For the triplet loss, both positive and negative outputs are used. In contrast, for the classification loss, only one of them is passed to the classifier,[1] concatenated with the input context $a$.

At inference time, only the binary classification is conducted. The input text and the LLM-generated output are concatenated and passed to the classifier to determine whether the output contains hallucination.

## 3 Experiment Settings

We evaluate whether contrastive learning could improve hallucination detection performance.

### 3.1 Dataset

We used the RAGTruth dataset (Niu et al., 2024) for our experiments. This dataset provides outputs generated by six different LLMs: GPT-3.5-turbo-0613, GPT-4-0613 (Achiam et al., 2023), Mistral-7b-Instruct (Jiang et al., 2023), Llama-2-7B-chat, Llama-2-13B-chat, Llama-2-70B-chat (Touvron et al., 2023). I.e., for each input, RAGTruth provides six outputs by these LLMs, with different levels of hallucinations. Each output is annotated with the hallucinated spans and their hallucination types. In accordance with the RAGTruth annotation protocol, hallucination is defined as content that is clearly different from the input, content not be supported by the input, or unverifiable or subjective statements.

[1]This setting was chosen to make our method directly comparable with other baselines. We can train the model by conducting classification with positive and negative samples simultaneously, which slightly improves the detection performance.

| | Train | Valid | Test |
|---|---|---|---|
| QA | 4,614 (3,756) | 420 (330) | 900 (564) |
| D2T | 4,878 (4,506) | 420 (390) | 900 (864) |
| SUM | 4,338 (4,074) | 420 (396) | 900 (780) |
| Total | 13,830 (12,336) | 1,260 (1,116) | 2,700 (2,208) |

Table 1: Dataset statistics (Parentheses indicate the number of triples.)

The original datasets of RAGTruth come from question answering (QA), data-to-text generation (D2T), and news summarization (SUM), with each task having varying hallucination rates across the LLM outputs. For the QA task, the input consists of a passage and a question from MS MARCO (Nguyen et al., 2016), and the output is the corresponding answer. For the D2T task, the input is JSON-formatted structured data (restaurant metadata and user reviews) from the Yelp Open Dataset (Yelp, 2017), and the output is a natural language description of that data. For the News Summarization task, the input is a news article (primarily from the CNN/Daily Mail dataset (See et al., 2017)), and the output is a summary.

We constructed triplets of (input text, faithful output, hallucinated output) using the outputs of the six LLMs. The original dataset contained 17,790 generated outputs, from which we extracted 15,660 triplets after discarding cases where all outputs are faithful or hallucinated. For evaluation, we used the 2,208 triplets in the test split across all settings. Since the RAGTruth does not provide a validation set, we randomly sampled a subset from the training data for validation. The number of samples for each split is summarized in Table 1.

### 3.2 Implementation

We used the light-weight, encoder-based model of RoBERTa-base (Liu et al., 2019) with 125M parameters as the base model for the classifier. As the text embedding, we employ the hidden outputs of the final layer corresponding to the start-of-sequence token, i.e., "<s>", attached to the input text.

We also experimented with a light-weight decoder-based LLM of Phi-3.5-mini-instruct (Abdin et al., 2024), that has 3.8B parameters. As the text embedding encoded by this model, we used the hidden output of the final layer corresponding to the last token of the input.

Fine-tuning was conducted for 10 epochs with a learning rate of $5.0e-6$ for RoBERTa-base and $1.0e-6$ for Phi-3.5-mini-instruct. The margin value $\alpha$ in our method was set to 1.0 for RoBERTa-base and 0.5 for Phi-3.5-mini-instruct based on the performance on the validation set. Yet the preliminary experiments showed that the detection performance is not sensitive to the $\alpha$ setting. All the experiments were conducted on a NVIDIA H100 GPU with 94GB memory.

### 3.3 Baselines

We compared our method against the following three baselines.

**LLM-Prompting** This method prompts LLMs to detect hallucinations. Given an input text and its corresponding output, an LLM was prompted to judge whether the output contained hallucination. We used both Phi-3.5-mini-instruct and GPT-4o as LLMs. The prompts can be found in the Appendix.

**FActScore** As a strong hallucination detection method applicable to the scenario where LLMs that generated outputs are unknown, we compare to FActScore. FActScore requires a knowledge base to identify hallucinations. To make it compatible with RAGTruth dataset, we used the input texts as the knowledge source, i.e., regarding outputs that are not supported by the input contexts as hallucinations. Following the original setting of Min et al. (2023), GPT-3.5-turbo was used as the base model to decompose output texts into a sequence of atomic facts and to calculate the percentage of the facts supported by the input text. If the computed score was exactly 1.0, a generated output was labeled as faithful; otherwise, it was considered hallucinated.

**Classifier** As an ablation study, we compared our method to its variation that trains the binary classifier using only the cross-entropy loss, without the triplet loss. Our method and this Classifier baseline were trained using all samples in the training split across tasks.

## 4 Results and Discussion

### 4.1 Overall Performance

Table 2 shows the precision, recall, and F1 scores for hallucination detection on different tasks. The "ALL" column shows these scores measured on all samples across tasks. The proposed method achieved the best F1 scores on QA, D2T, and ALL tasks when combined with RoBERTa, largely outperforming a much larger-scale model of GPT-4o and FActScore. The proposed method with RoBERTa showed higher recall. GPT-4o

| Model | Method | QA | | | D2T | | | SUM | | | ALL | | | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 | |
| GPT-4o | Prompt | 60.7 | 46.3 | 52.5 | **94.0** | 63.4 | 75.7 | **89.1** | 49.5 | **63.6** | **86.3** | 57.3 | 68.8 | 2.01 |
| GPT-3.5 | FactScore | 35.3 | 88.1 | 50.4 | 66.9 | **94.3** | 78.3 | 33.2 | **66.7** | 44.3 | 50.3 | **87.1** | 63.7 | 2.29 |
| RoBERTa | Classifier | 45.8 | 60.0 | 57.0 | 80.9 | 90.2 | 85.3 | 34.2 | 27.3 | 30.3 | 78.3 | 58.2 | 66.8 | **0.01** |
| | Proposed | 62.7 | **88.7** | **60.4** | 79.9 | 91.9 | **85.5** | 33.5 | 54.0 | 41.4 | 59.8 | 83.1 | **69.5** | 0.03 |
| Phi-3.5 | Prompt | 27.3 | 1.9 | 3.5 | 50.0 | 4.6 | 8.4 | 30.8 | 20.2 | 24.3 | 35.6 | 7.5 | 12.5 | 0.45 |
| | Classifier | 59.5 | 56.9 | 58.1 | 82.4 | 86.0 | 84.1 | 35.2 | 32.3 | 33.7 | 74.0 | 63.8 | 68.5 | 0.29 |
| | Proposed | **71.0** | 44.1 | 54.4 | 83.4 | 83.8 | 83.6 | 38.7 | 35.8 | 37.2 | 67.1 | 70.1 | 68.6 | 0.34 |

Table 2: Precision (P), Recall (R), and F1 scores (%) for hallucination detection across tasks. "Time" indicates average time per case.

demonstrated higher precision, whereas FActScore showed higher recall. GPT-4o and FActScore performed strongly on the summarization task, but the performance was limited on other settings.

Hallucination detection on summarization task requires detailed comparisons of a long input document and a shorter output summary. We conjecture GPT-4o and GPT-3.5 are capable of such comparison, but it may be difficult for much smaller RoBERTa-base. Our method on Phi-3.5-mini-instruct was consistently inferior to that on RoBERTa. This may be due to the differences in embeddings from the encoder or decoder; a detailed investigation is our future work.

The far right column shows the computational time: the average second to process a sample. Our method on RoBERTa is much faster than other decoder-based LLMs, thanks to the efficient encoder model and its small number of parameters. Prompting GPT-4o and FActScore took 67.0 to 76.3 times longer than our method.

## 4.2 Analysis

This section investigates features of hallucinations that can affect the detection performance by comparing our method on RoBERTa and GPT-4o.

**Effect of Hallucinating Models** Table 3 presents F1 score for hallucination detection, grouped by the LLM that generated the outputs. Overall, the detection rate tends to be higher for generations containing more hallucinations. Although we hypothesized that GPT-4o may have a higher success rate on GPT-3.5 and GPT-4, this did not hold. Rather, the task differences are more dominant than the model differences.

**Number of Hallucinations** Figures 2 and 3 show the success rate of hallucination detection as a function of the proportions of the number of hallucinated tokens and the number of hallucinated spans, respectively. The bar charts in the background indicate the numbers of samples within each bin. Hallucinations with smaller proportions are more challenging to detect, yet such cases are more prevalent in the dataset. Nevertheless, our method achieved significantly higher detection rates than GPT-4o in these cases.

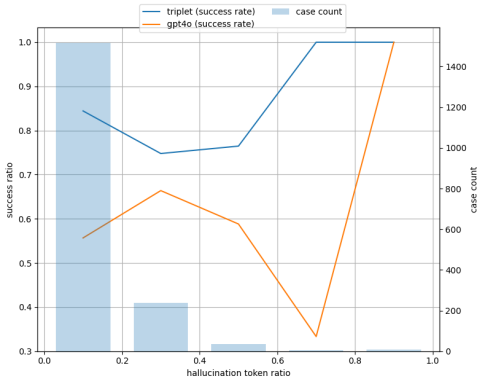


Figure 2: Detection success ratio and the number of cases by hallucinating token ratio in an output

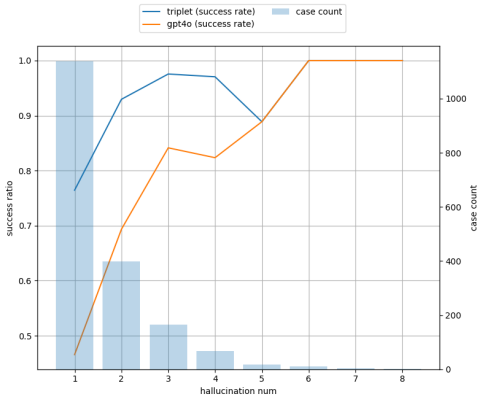


Figure 3: Detection success ratio and the number of cases by the number of hallucinations in an output

**Embedding Space** Figures 4 and 5 visualizes the distributions of cosine distances between the input and faithful/hallucinated outputs before and after contrastive learning. In the original embeddings, the distributions for faithful and hallucinated

| | | GPT3.5 | GPT4 | Llama2-7B | Llama2-13B | Llama2-70B | Mistral |
|---|---|---|---|---|---|---|---|
| QA | GPT4o | 14.3 | 0.0 | 68.7 | 43.6 | 40.0 | 55.7 |
| | Proposed | **21.4** | 0.0 | **74.6** | **65.4** | **57.7** | **65.2** |
| | Num | 5 | 1 | 52 | 36 | 35 | 31 |
| D2T | GPT4o | 21.1 | 6.5 | 74.2 | 93.0 | 67.5 | 82.0 |
| | Proposed | **31.3** | **21.3** | **89.7** | **95.7** | **84.8** | **94.1** |
| | Num | 31 | 29 | 117 | 132 | 106 | 128 |
| SUM | GPT4o | 0.0 | **50.0** | **65.8** | **46.8** | **54.5** | **72.5** |
| | Triplet | 0.0 | 16.7 | 49.1 | 34.3 | 35.7 | 63.4 |
| | Num | 3 | 5 | 50 | 32 | 23 | 85 |
| ALL | GPT4o | **18.2** | 14.3 | 71.0 | **79.4** | 60.2 | 75.1 |
| | Proposed | 17.1 | **16.3** | **77.0** | 79.1 | **69.1** | **79.7** |
| | Num | 39 | 35 | 219 | 200 | 164 | 244 |

Table 3: F1 for hallucination detection per model ("Num" rows show the number of samples with hallucination.)

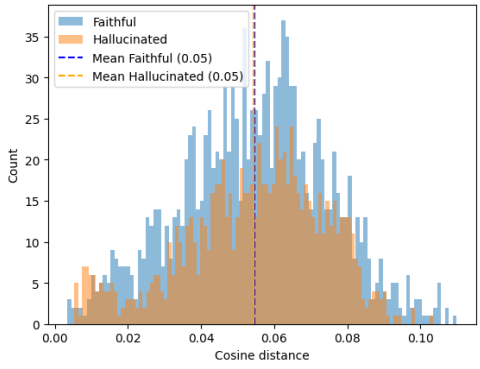


Figure 4: Distribution of cosine distances between original embeddings (before contrastive learning)

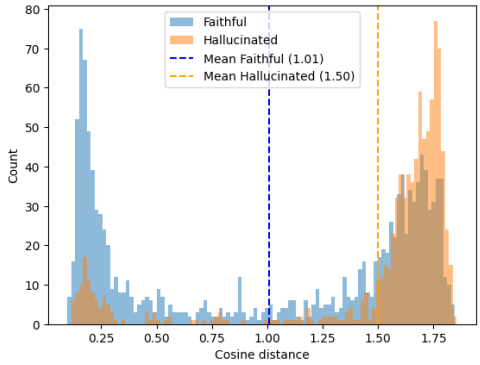


Figure 5: Distribution of cosine distances after contrastive learning

outputs are highly similar, with both distributions tightly concentrated in a narrow range. This indicates that inputs, faithful and hallucinated outputs are entangled in the embeddings space. After contrastive learning using triplet loss, these are well disentangled. The cosine distance distributions of faithful and hallucinated outputs differ significantly, with their respective peaks clearly shifted from each other in opposite directions.

## 5 Conclusion

We proposed a method for training a hallucination detector using contrastive learning. Experimental results demonstrated that our method is particularly effective for detecting cases where proportions and/or numbers of hallucinated spans are smaller, which are typically more challenging to identify. In future, we will explore methods for locating and identifying hallucinated spans in generation, which remains an open problem despite its practical importance.

## Limitations

Our method requires an input context to identify hallucination in generated output; hence, it does not apply to scenarios where only generated outputs are available, such as fake news detection.

Our method requires triples of (input context, hallucinated output, faithful output), which requires extra efforts in construction rather than simpler pairs of (input context, hallucinated or faithful output). Nonetheless, such triples can be collected using sampling in generation or using multiple LLMs.

## Acknowledgments

This work was supported by JST K Program Grant Number JPMJKP24C3, Japan. This study was carried out using the TSUBAME4.0 supercomputer at Institute of Science Tokyo.

## References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, and Harkirat et al. Behl. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and Shyamal et al. Anadkat. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Tanya Goyal and Greg Durrett. 2020. Evaluating factuality in generation with dependency-level entailment. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, and Lucile et al. Saulnier. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. 2024. On large language models' hallucination with regard to known facts. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1041–1053, Mexico City, Mexico. Association for Computational Linguistics.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *EMNLP*.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. 2024. Fine-grained hallucination detection and editing for language models. In *First Conference on Language Modeling*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. 2024. RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10862–10878, Bangkok, Thailand. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and

Shruti et al. Bhosale. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yelp. 2017. Yelp open dataset. http://www.pluto.ai.kyutech.ac.jp/NLP/.

# A Appendix

Table 4 shows prompts used in this study.

| | |
|---|---|
| Classifier, Triplet | [input text] Please judge the following statement whether it includes hallucination or not based on the references above: [output text] |
| Prompt (Phi) | Input_Document: [input text] Please judge the following Text whether it includes hallucination or not based on the Input_Document above and output 1 if it includes hallucination and 0 if not. Output should be only an number (1 or 0). You mustn't output any description other than a number. Text: [output text] Output: |
| Prompt (GPT4o) | [input text] Please judge the following statement whether it includes hallucination or not based on the references above and output 1 if it includes hallucination and 0 if not. Output should be only an number (1 or 0): [output text] Output: |

Table 4: Used prompt in the experiments