DUTJBD at SemEval-2025 Task 3: A Range of Approaches for Predicting Hallucination Generation in Models

Shengdi Yin, Zekun Wang, Liang Yang*, Hongfei Lin
Department of Computer Science Dalian University of Technology, LiaoNing, China
{20201071390,zk_wang}@mail.dlut.edu.cn
{liang,hflin}@dlut.edu.cn

Abstract

This paper describes our system designed for SemEval-2025 Task 3 : Mu-SHROOM, the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes. We explored using various methods to train models for predicting the occurrence of large language model hallucinations. The main techniques of our system are: 1) data augmentation, 2) model training, 3) API keys. We also experimented with various prompt engineering techniques and different closed-source large language models to predict the occurrence of hallucinations in a given text.

1 Introduction

In Task 3 (Vázquez et al., 2025), our goal is to predict the occurrence of hallucinations in text generated by various text generation models. In practice, we are provided with LLMs outputs (as strings, lists of tokens, and lists of logits) and we must calculate the probability that each character in the LLM output string is flagged as a hallucination. We can investigate 14 languages: Arabic (Modern Standard), Basque, Catalan, Chinese (Mandarin), Czech, English, Persian, Finnish, French, German, Hindi, Italian, Spanish, and Swedish, and can experiment and make predictions on one or more of the tasks. Task 3, compared to previous tasks, focuses more on the output of LLMs and the location where hallucinations occur. Therefore, it is more conducive to addressing problems caused by hallucinations.

We used the T5 model (Raffel et al., 2020a), a natural language processing model proposed by Google Research in 2019. Compared to other models, T5 can transform all NLP tasks into a text-to-text format. Therefore, T5 can handle various different tasks within the same model architecture, without the need to design separate models for each task.

At the same time, we also tested prompt engineering with several closed-source large language models, including ChatGPT 4o, Gemini 1.5, Qwen Chat, and DeepSeek V3 (Liu et al., 2024). Finally, Gemini 2.0 Flash was used to correct some specific text.

2 Background

2.1 Dataset Description

The training set includes a total of four languages: English, Spanish, French, and Chinese. Each language contains approximately 800 unlabeled data entries.

Each sample in the training set includes a *model_id* key indicating the source of the hallucinated text. The samples also include the *logits* for each generated token, representing the model's confidence in each generated word. Additionally, the output *tokens* represent the model-generated text encoded in subword form. These samples are used to train the model to answer similar questions and generate accurate, natural text replies.

2.2 Related Work

Hallucination detection is an important research direction in the field of natural language generation, especially in the application of large language models (LLMs). Generative models, particularly Transformer-based models such as the GPT series and T5, have achieved significant results on multiple NLP tasks. However, they often generate false or inaccurate content, a phenomenon known as hallucination (Raffel et al., 2020b). Research on hallucination detection aims to improve the practicality and reliability of these models by identifying and correcting such inaccurate generated content (Bender et al., 2021) conducted a detailed analysis of the hallucination problem in generative models, emphasizing that the hallucination phenomenon is closely related to the model's ability to understand the world, and proposed several potential solutions.

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 909–913

Building on this (Liu et al., 2019) proposed a multi-task learning framework to improve the accuracy of hallucination detection through cross-task multi-model integration, particularly when addressing generation tasks in diverse domains. Similarly (Ziegler et al., 2019) adopted a reinforcement learning method, reducing hallucinations in generated text by combining the output of multiple models, thereby enhancing detection performance.

In multilingual environments, hallucination detection becomes more complex due to the significant differences in grammar, semantics, and cultural backgrounds across languages. The MuSHROOM task specifically addresses this challenge by focusing on multilingual hallucination detection, encompassing 14 languages including English, Arabic, and Chinese (Liu et al., 2020) proposed a cross-lingual hallucination detection framework in their research, which utilizes multilingual pre-trained models for hallucination annotation and achieves promising results. Furthermore, unified text-to-text frameworks, such as T5 (Raffel et al., 2020b), provide strong support for multi-task learning. The T5 model's ability to transform various NLP tasks into text generation tasks allows it to handle multiple tasks like translation, summarization, and question answering within a single framework. This unified approach not only simplifies the model training process but also enhances adaptability for hallucination detection, especially in multi-task learning scenarios, ultimately improving detection accuracy through shared model parameters.

3 System Overview

Our system is designed to effectively identify hallucinations in generated text within the MuSHROOM task. To achieve this goal, we employ a confidence-based pseudo-labeling approach, which relies on the following key steps: First, we convert the model's logits into confidence scores, thereby quantifying the model's certainty for each generated token. We then distinguish between hallucinated and non-hallucinated portions of the text by setting a threshold. Subsequently, we generate pseudo-labels based on the threshold, providing the model with learnable targets. Finally, we train the model using cross-entropy loss and backpropagation, validating the T5 model trained in this way on the hallucination task.

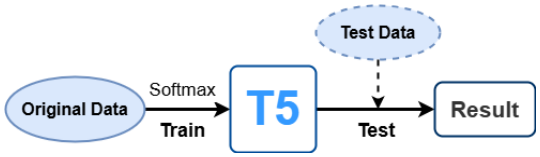

Figure 1: Illustration of the T5 model training process

3.1 Model Training Procedure

To effectively detect hallucinations in the MuSHROOM task, we designed a training procedure consisting of the following three key steps:

3.1.1 Transforming Logits to Probabilities

The logits produced by the model when generating each token represent unnormalized scores, directly reflecting the model's preference for that token. However, the numerical range of logits is often large and difficult to interpret directly. To obtain more interpretable and comparable confidence scores, we employ the softmax function to transform the logits into a probability distribution. The softmax (Bridle, 1990) function not only maps the logits to a range between [0, 1] but also ensures that the probabilities of all tokens sum to 1, forming a valid probability distribution. This transformation allows us to interpret the model's output as the degree of confidence it has in each token. The softmax function is defined as follows:

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z represents the vector of logits, z_i represents the logit value for the i-th token, and K is the total number of tokens. Through the softmax function, the logits are converted into probabilities within the range $[0, 1]$, representing the model's confidence in generating that token. A higher probability corresponds to higher confidence, indicating that the model has a high degree of certainty about the generated content for that token; conversely, a lower probability indicates that the model is less confident about generating that part of the content, potentially suggesting a hallucination.

3.1.2 Hallucination Discrimination

After obtaining the confidence score for each token, we need a method to distinguish between hallucinated and non-hallucinated portions of the text. To achieve this goal, we compare the confidence scores output by the model with a pre-set threshold. This threshold represents the minimum level

of confidence at which we believe the model can reliably generate content.

Specifically, for each token, if its confidence score is below the threshold (e.g., 0.2), we mark it as "hallucination"; conversely, if the confidence score is above the threshold, we mark it as "non-hallucination." The choice of this threshold is crucial, as it directly affects the quality of the pseudo-labels. Selecting a threshold that is too high may lead to the labeling of much factual information as hallucination, while selecting a threshold that is too low may prevent the effective identification of true hallucinations. This method allows us to transform unlabeled data into pseudo-labeled data with hallucination labels, providing a target for subsequent training and enabling the model to learn to distinguish between factual information and hallucinations in the text.

3.1.3 Model Optimization

To train the model to identify and reduce hallucinations in generated text, we explore the use of regression loss. Here we define the regression loss as a Mean Squared Error (MSE) between two values. Specifically, our model aims to minimize the hallucination by optimizating the following objective during training procedure:

$$L_R^i = \text{MSE}(y_1^i, y_2^i)$$

where y_1^i and y_2^i are the target and the predicted values respectively. The backpropagation algorithm updates the model parameters based on the gradient of the loss function, enabling the model to better predict hallucinations in the next iteration. Furthermore, to prevent overfitting, we also employ regularization techniques such as dropout and weight decay.

3.2 Logit-Based Hallucination Detection

In the testing phase, our goal is to detect hallucinations in the generated text using the T5 model trained with pseudo-labels. Similar to the training phase, we use the logits produced by the T5 model to generate pseudo-labels for the test set and identify potential hallucinations.

The process starts by inputting the generated text from the test set into the trained T5 model. The model then generates logits for each token. We apply the softmax function to convert these logits into confidence scores, reflecting the model's certainty about the generated content. Tokens with confidence scores below a pre-defined threshold are flagged as potential hallucinations. This approach allows us to automatically detect and highlight possible hallucinations in the generated text without relying on human annotations.

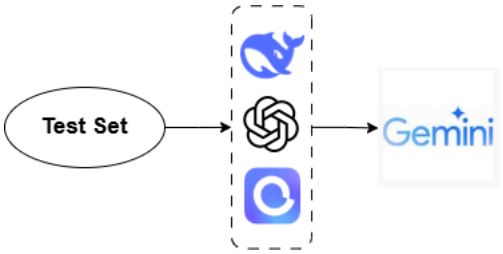

Figure 2: Employing prompt engineering to invoke diverse large language models.

3.3 Experimental Results

The T5 model, trained using our proposed logit-based pseudo-labeling approach, demonstrated a significant improvement in performance on the MuSHROOM hallucination detection task. Specifically, our model achieved an accuracy increase of 2 percentage points over the baseline, showcasing the effectiveness of our training methodology. This improvement demonstrates the ability of our method to learn effective representations for hallucination detection, enabling more accurate identification of spurious content in generated text.

4 Prompt Engineering

This chapter explores the feasibility of addressing the hallucination detection task in a zero-shot setting by leveraging prompt engineering to invoke several cutting-edge APIs, including ChatGPT 4o, Gemini 1.5, Qwen Chat, and DeepSeek V3. These APIs possess robust text generation and understanding capabilities, offering a potential avenue for detecting hallucinations in text without the need for training dedicated models. We will investigate how to harness the inherent abilities of these APIs for direct application to the hallucination detection task.

4.1 Prompting Strategy

We designed a series of prompts and directly submitted them to the aforementioned APIs. Each API received the same task: to detect hallucinations in the generated text and return the corresponding results. By comparing the responses from different APIs, we were able to evaluate their performance on the hallucination detection task. The specific content of the APIs will be provided in the appendix.

Model	Result
GPT 4o	**0.0571**
Gemini 1.5	0.0389
DeepSeek V3	0.024
Qwen Chat	0.0187

Table 1: Results of different large language models on the experimental data.

4.2 API Evaluation

To evaluate the effectiveness of these APIs in hallucination detection, we utilized a diverse set of generated texts and tested them individually with ChatGPT 4o, Gemini 1.5, Qwen Chat, and DeepSeek V3. Each API generated hallucination annotations based on the defined prompts, assisting in the identification of potential hallucinated segments.

4.3 Experimental Results

Based on our experimental results, the approach of invoking APIs using prompt engineering did not achieve ideal outcomes. We hypothesize that this may be due to the fact that these large models themselves are prone to generating hallucinations, thus limiting their ability to effectively identify them. As the quality of the generated text is constrained by the inherent limitations of the models, they struggle to differentiate between plausible content and that which is fictitious or erroneous. While large models may produce errors during generation, they typically lack the ability to proactively identify and flag these inaccuracies, particularly when dealing with complex hallucination detection tasks.

To further investigate this phenomenon, we attempted to elicit the reasoning processes of these large models, hoping to reveal their logic when judging hallucinations. However, the results indicated that the models did not genuinely comprehend the concept of hallucination. In our reasoning, a hallucination equates to fabricated content—i.e., generated text that contradicts facts in the real world. Yet, for these large models, a hallucination is merely a simple error; they tend to focus more on spelling errors, grammatical errors, or irregularities in sentence structure, rather than correctly identifying fictional content. This suggests that the core issue in large models' handling of hallucination detection tasks may be a discrepancy between their focus and our definition of hallucination.

5 Conclusion

This paper explores two approaches for tackling the hallucination detection task: the first involves training a T5 model with pseudo-labels, and the second leverages prompt engineering to invoke multiple large language model APIs. Initially, by training the T5 model and applying pseudo-label generation, we successfully improved hallucination detection performance. Experimental results indicate that the T5 model achieved approximately a 2 percentage points increase in detection accuracy compared to the baseline.

In addition, we attempted to directly invoke large language models such as ChatGPT 4o, Gemini 1.5, Qwen Chat, and DeepSeek V3 via prompt engineering for hallucination detection. However, experimental results revealed that this approach did not yield the anticipated results. The primary reason for this is likely that these large models themselves are prone to generating hallucinations, consequently limiting their ability to effectively identify them. While these models excel in generating text, they tend to focus more on spelling and grammatical errors when handling hallucination detection tasks, struggling to effectively identify hallucinated content.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and

Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

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

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

Raúl Vázquez, Timothee Mickus, Elaine Zosa, Teemu Vahtola, Jörg Tiedemann, Aman Sinha, Vincent Segonne, Fernando Sánchez-Vega, Alessandro Raganato, Jindřich Libovický, Jussi Karlgren, Shaoxiong Ji, Jindřich Helcl, Liane Guillou, Ona de Gibert, Jaione Bengoetxea, Joseph Attieh, and Marianna Apidianaki. 2025. SemEval-2025 Task 3: Mu-SHROOM, the multilingual shared-task on hallucinations and related observable overgeneration mistakes.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Appendix

The following is the pseudocode for the prompts we used.

"You are an expert in identifying hallucinations in large language models. Please help me identify which tokens in the sentence are hallucinations.

I will be providing you with input similar to the following:

<Examples from the training set>

"model_input" will be the input to another large language model, "model_output_text" its text output, and "model_output_tokens" its output tokens. You can refer to that model's "model_output_logits" as a reference. Your output should be both hard and soft predictions for the output tokens.

```
{
 "soft_labels":[
   {"start":10,"prob":0.2,"end":12},
   {"start":12,"prob":0.3,"end":13},
   {"start":13,"prob":0.2,"end":18},
   {"start":25,"prob":0.9,"end":31},
   {"start":31,"prob":0.1,"end":37},
   {"start":45,"prob":1.0,"end":49},
   {"start":49,"prob":0.3,"end":65},
   {"start":65,"prob":0.2,"end":69},
   {"start":69,"prob":0.9,"end":83}
 ],
 "hard_labels":[[5,31],[45,49]]
}
```

The "prob" for soft predictions represents the hallucination probability between tokens, and the hard prediction is the starting position of tokens that you identify as hallucinations. Please provide an example output based on the instructions above.

<text>

Important: Do not output your reasoning. Provide the answer directly in the requested format."