iai_MSU at SemEval-2025 Task-3: Mu-SHROOM, the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes in English

Mikhail Pukemo[1,2] **Aleksandr Levykin**[1,2] **Dmitrii Melikhov**[1,2]
Gleb Skiba[1,2] **Roman Ischenko**[1,2] **Konstantin Vorontsov**[1,2]
Institute of AI, Lomonosov Moscow State University[1],
The faculty of Computational Mathematics and Cybernetics,
Lomonosov Moscow State University[2]
M.Pukemo@iai.msu.ru, A.Levykin@iai.msu.ru, D.Melihov@iai.msu.ru,
gleb-skiba@mail.ru, roman.ischenko@gmail.com, Voron@mlsa-iai.ru

Abstract

This paper presents the submissions of the iai_MSU team for SemEval-2025 Task 3 – Mu-SHROOM, where we achieved first place in the English language. The task involves detecting hallucinations in model-generated text, which requires systems to verify claims against reliable sources. In this paper, we present our approach to hallucination detection, which employs a three-stage system. The first stage uses a retrieval-based method (Lewis et al., 2021) to verify claims against external knowledge sources. The second stage applies the Self-Refine Prompting approach (Madaan et al., 2023) to improve detection accuracy by analyzing potential errors of the first stage. The third stage combines predictions from the first and second stages into an ensemble. Our system achieves state-of-the-art performance on the competition dataset, demonstrating the effectiveness of combining retrieval-augmented verification with Self-Refine Prompting. The code for the solutions is available on GitHub[1]

1 Introduction

Large language models (LLMs) have revolutionized natural language processing (NLP), demonstrating strong capabilities in text generation, summarization, and dialogue systems. However, LLMs remain prone to hallucinations, where generated content contains false, misleading, or unverifiable information. Addressing this issue is crucial for real-world applications, especially in domains requiring high factual accuracy, such as journalism, medicine, and law. The ability to detect and mitigate hallucinations is essential to improve the reliability and trustworthiness of LLM-generated content.

SemEval-2025 Task 3 – Mu-SHROOM[2](Vázquez et al., 2025) introduces a multilingual hallucination detection challenge, requiring participants to identify specific spans of hallucinated text within model-generated output. Unlike traditional fact-checking tasks, Mu-SHROOM provides LLM-generated text alongside tokenized representations and logit scores, and participants must compute a probability score for each character, indicating its likelihood of being a hallucination. The task covers 14 languages, including English, Chinese, Arabic and several European languages, presenting unique challenges such as linguistic diversity, cross-lingual hallucination patterns, and variations in model behavior. To tackle these challenges on English language, we propose a three-stage hallucination detection system:

Stage 1: We employ a Retrieval-Augmented Generation (RAG) pipeline, using Wikipedia as an external knowledge source to verify input claims.

Stage 2: We employ an Self-Refine Prompting strategy, where an LLM re-evaluates the first-stage output to identify potential errors and refine hallucination predictions.

Stage 3: We use an Ensemble strategy that merges three predictions from the first stage and three from the second stage to create the final submission.

2 Related Works

Hallucination detection in large language models (LLMs) is a critical area of research, focusing on identifying and mitigating instances where models generate content that is plausible but factually incorrect. Various approaches have been proposed to address this challenge, including methods utilizing LLMs themselves, retrieval-augmented verification techniques, and self-refinement prompting strategies.

LLMs can be utilized to detect hallucinations by analyzing their internal states and output. In "Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language

[1]https://github.com/pansershrek/IAI_MSU
[2]https://helsinki-nlp.github.io/shroom/

Models," (Su et al., 2024) the authors propose MIND, an unsupervised training framework that leverages the internal states of LLMs for real-time hallucination detection without requiring manual annotations. This approach utilizes the model's internal representations during inference to identify incoherent or factually inaccurate responses. But we face a more difficult task as soon as we have only tokens' logits.

Another study, "Hallucination Detection: Robustly Discerning Reliable Answers in Large Language Models" (Chen et al., 2024) introduces a robust discriminator named RelD to effectively detect hallucinations in LLM-generated answers. RelD is trained on a bilingual question-answering dialogue dataset, enabling it to identify unfaithful or inconsistent content generated by diverse LLMs.

Integrating external knowledge sources into the generation process can enhance the factual accuracy of LLM output. In "Mitigating Hallucinations in Large Language Models via Self-Refinement-Enhanced Knowledge Graph Retrieval" (Niu et al., 2024) the authors propose Re-KGR, a method that augments the factuality of LLMs' responses by leveraging knowledge graph retrieval. This approach identifies tokens with a high potential for hallucination and refines the associated knowledge triples to reduce verification efforts.

Similarly, "Self-Alignment for Factuality: Mitigating Hallucinations in LLMs via Self-Alignment" (Zhang et al., 2024) introduces SK-Tuning, a strategy that improves an LLM's confidence estimation and calibration, thereby enhancing its self-evaluation ability. This method aligns the model's output with external knowledge to mitigate hallucinations.

Self-refinement prompting strategies involve iterative processes where LLMs generate, evaluate, and refine their output to improve factual accuracy. The "Self-Refine" (Madaan et al., 2023) approach allows LLMs to iteratively refine output and incorporate feedback along multiple dimensions to improve performance on diverse tasks. This method does not require supervised training data or reinforcement learning and works with a single LLM.

Additionally, "Towards Mitigating Hallucination in Large Language Models via Self-Reflection" (Ji et al., 2023) proposes an innovative self-reflection method to mitigate hallucination in LLMs. The iterative feedback loop process generates, scores, and refines responses to reduce hallucinations, particularly in medical question-answering systems.

3 Task solutions

3.1 Dataset and Database for RAG

To enhance the accuracy of hallucination detection, our system utilizes a RAG pipeline that incorporates external knowledge sources. We use the Wikipedia dataset (Foundation), only an English subset with 6.41M articles, as our primary factual reference. We also clean all articles by removing references and repetitive newline characters.

For efficient retrieval, we employ Qdrant[3] as a vector database, which enables fast and scalable similarity searches. We use the Multilingual-E5-Large (Wang et al., 2024) embedding model to generate dense vector representations of the text. To optimize retrieval performance and storage efficiency, we embed only the first 512 characters of each Wikipedia article. Similarity between queries and stored embeddings is computed using Cosine distance and HNSW as a search algorithm.

3.2 Our Solution: Only LLM

In our approach, we evaluate hallucination detection using standalone LLMs without retrieval augmentation only on validation dataset. Specifically, we experiment with Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-72B (Yang et al., 2024) using prompt mentioned in Appendix A.1 and Appendix A.2. These experiments achieve best result of 39.7% IoU and 38.4% Corr.

3.3 Our Solution: RAG Pipeline

To enhance hallucination detection, we implement a RAG pipeline, utilizing Qwen2.5-72B as the LLM and a vector database, as detailed in the "Dataset and Database for RAG" section. We run all our experiments on the validation dataset.

3.3.1 Experiment 1: Initial Prompts with Top-1 Document

We begin by testing prompt mentioned in Appendix A.3, retrieving only the Top-1 related document from the database. This initial setup provides results of 51% IoU and 53% Corr.

3.3.2 Experiment 2: One-Shot

To enhance the model's ability to detect hallucinations, we introduce one-shot prompting strategies, using prompts mentioned in Appendix A.4. These

[3]https://qdrant.tech/

modifications should improve performance by providing clear examples of hallucination detection. The results on the validation is 52.9% IoU and 52.8% Corr.

3.3.3 Experiment 3: Expanding to Top-5 Documents RAG

Finally, we increase the number of retrieved documents to Top-5 related documents, allowing the model to cross-check its output against a broader knowledge base. We experiment with prompt in Appendix A.4 by adding 4 more examples for this setting, leading to further results of 48.1% IoU and 45.6% Corr. Among all tested methods, the One-Shot Prompting with Top-1 Document Retrieval delivers the best performance.

3.4 Our Solution: Self-Refine Prompting

To further improve hallucination detection, we apply a Self-Refine Prompting strategy, where the model evaluates and refines its own generations. We use Qwen2.5-72B as the LLM to refine output produced by our RAG pipeline with One-Shot Prompting and Top-1 Document Retrieval (see “Our Solution: RAG Pipeline” section). For this refinement step, we experiment with prompt in Appendix A.5 and get results 53.9% IoU and 52.1% Corr. We ran this stage only one time for each sample.

3.5 Our Solution: Final Submission

For our final submission, we adopt a two-stage approach leveraging a RAG pipeline followed by self-refinement. We use GPT-4o as the LLM to generate and refine hallucination predictions.

3.5.1 Stage 1: RAG-Based Hallucination Detection

In the first stage, we apply One-Shot Prompting with Top-1 Document Retrieval from our RAG pipeline (see “Our Solution: RAG Pipeline” stage). This stage utilizes prompt from Appendix A.4 to generate initial hallucination predictions. We also want to create Reranking stage in RAG pipeline to handle cases where the retrieved documents from Wikipedia were ambiguous or conflicting, but we haven’t enough time.

3.5.2 Stage 2: Self-Refinement

In the second stage, we refine the output from Stage 1 using the approach from “Our Solution: Self-Refine Promptingfrom” section with our RAG pipeline, again using prompt from Appendix A.5. This refinement step helps correct potential errors and improves the final hallucination detection.

3.5.3 Stage 3: Ensemble Strategy

To further enhance robustness, we construct an ensemble model by combining multiple runs of the system with different temperature settings:
Stage 1 Only: We generate three independent outputs using the same prompt and temperatures 0.05, 0.1, 0.2 (we didn’t test different temperatures).
Stage 1 + Stage 2: We generate three additional outputs using the full two-stage pipeline, with temperatures 0.05, 0.1, 0.2 (we didn’t test different temperatures).

3.5.4 Ensembling Technique

We receive hard labels from stages 1 and 2 (a list of indices corresponding to hallucinated spans). To ensemble multiple outputs, we convert these hard labels into soft labels, representing hallucination probabilities for each symbol. For each character, its hallucination probability is computed as the fraction of models that marked it as part of a hallucinated fragment. The final answer is represented using length-range encoding of these probabilities. We additionally remove 1-symbol hallucinations and all punctuation marks from hallucinations.

Example: Given the model-generated sentence: "Petra van Stoveren won a silver medal in the 2008 Summer Olympics in Beijing, China." Three models return different hallucination fragments:
Model 1: "silver"
Model 2: "silver medal"
Model 3: "won a silver medal"

The final ensemble prediction assigns probabilities:
"won a" $\rightarrow$ 0.33
"silver" $\rightarrow$ 1.00
"medal" $\rightarrow$ 0.66

For the evaluation system, this output is recorded as a list of indices corresponding to hallucination probabilities.

3.5.5 Result

This three-stage pipeline with ensembling significantly enhances hallucination detection performance, balancing retrieval-based verification, self-refinement, and ensemble robustness to improve overall quality. Our final approach achieves 65.09% IoU and 62.94% Corr, demonstrating the effectiveness of our method in detecting hallucinations in LLM-generated text.

4 Conclusion

In this paper, we presented the submissions of the iai_MSU team for SemEval-2025 Task 3 — MuSHROOM, where we achieved first place in the English language track. Our approach combines Retrieval-Augmented Generation (RAG) with Self-Refine Prompting, demonstrating the effectiveness of integrating external knowledge verification with iterative model refinement. We introduced a three-stage pipeline where the first stage uses a RAG-based method to verify claims, followed by the second stage where Self-Refine Prompting refines hallucination detection output for improved accuracy. Additionally, the use of ensemble techniques further enhanced robustness by aggregating output from multiple runs and varying temperature settings.

Our final submission achieves 65.09% IoU and 62.94% Corr, confirming the strength of our methodology in detecting hallucinations and improving the factual accuracy of model-generated text.

References

Yuyan Chen, Qiang Fu, Yichen Yuan, Zhihao Wen, Ge Fan, Dayiheng Liu, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. 2024. Hallucination detection: Robustly discerning reliable answers in large language models.

Wikimedia Foundation. Wikimedia downloads.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, and Aiesha Letman. 2024. The llama 3 herd of models.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating hallucination in large language models via self-reflection.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. 2024. Mitigating hallucinations in large language models via self-refinement-enhanced knowledge retrieval.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran HU, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised real-time hallucination detection based on the internal states of large language models.

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.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024. Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation.

A Appendix

This appendix shows the training and generation parameters for the models described above in the text.

A.1 First Zero-shot prompt without RAG

You are a fact - checking assistant.Your task is to identify fragments of the response that are hallucinations–parts of the text that are factually incorrect or made up by model.Pay attention to facts, dates, numbers, places. Detect only hallucination words, without neighbour words. Give me only a list of fragments - hallucinations you found in model output. Write answer in JSON with the next structure:
{ ”hallucinations”: [”h1”, ”h2”] },
where h1 and h2 are hallucination fragments from

model output.
Model output:

A.2 Second Zero-shot prompt without RAG

You are assistant for analysing model hallucinations. Your task is to extract fragments from model output, containing factually incorrect answers. You need to extract factually incorrect or inconsistent with input fragments from model output. Write answer in JSON with the next structure:
{ ”hallucinations”: [”h1”, ”h2”] },
where h1 and h2 are hallucination fragments from model output. Write in answer only JSON structure without other comments.
Model output:

A.3 Zero-shot prompt with RAG

You are a fact-checking assistant for analysing model hallucinations. Your task is to identify fragments in model output that are hallucinations - parts of the text that are factually incorrect or made up by model or inconsistent with model input. You get a user query in model input and hallucinated answer in model output. You also get a reliable relevant document from Wikipedia, pay attention to it while checking facts in hallucinated model output. Detect only hallucination words, without neighbour common, linking words. Write answer in JSON with the next structure:
{ ’hallucinations’: [’h1’, ’h2’]},
where h1 and h2 are hallucinations from model output. Write your answer exactly in JSON structure without other symbols.
Relevant document: {doc 1}
Model input: {model input}
model output: {model output text}
Your answer:

A.4 One-shot prompt with RAG

You are a fact-checking assistant for analysing model hallucinations. Your task is to identify fragments in model output that are hallucinations - parts of the text that are factually incorrect or made up by model or inconsistent with model input. You get a user query in model input and hallucinated answer in model output. You also get a reliable relevant document from Wikipedia, pay attention to this document while checking facts in hallucinated model output. Detect only hallucination fragments, without neighbour common, linking words. Write answer in JSON with the next structure:
{’hallucinations’: [’h1’, ’h2’]},
where h1 and h2 are hallucination fragments from model output. Write in answer only JSON structure without other comments. Here is an example of correct dialogue:
Relevant document example:
Model input example: {model input}
model output example: {model output text}
Your answer example:
{’hallucinations’: [...]}
Input:
Relevant document: {doc 1}
Model input: {model input}
model output: {model output text}
Your answer:

A.5 Self-refine prompt

You are an assistant to check the correctness of detected hallucinations - hallucinations that were detected in model output, model output was generated by model input (question, given by user). Hallucinations are parts of the model input that are factually incorrect or made up by model or inconsistent with model input. detected hallucinations were detected by other model by given model input, model output and reliable relevant document from Wikipedia. You get the model input, model output, relevant document (pay attention to it while fact checking) and detected hallucinations (a Python list of strings that are hallucinations from model output). Your task is to fix errors in detected hallucinations, improve it by adding all missed hallucinations and removing all detections that are not hallucinations. Detect only hallucination fragments, without neighbour common, linking words. Write the answer in the same JSON format. Write in answer only JSON structure without any other comments. Here is an example of correct dialogue:
Relevant document №1 example :
Model input example: {model input}
model output example: {model output text}
Detected hallucinations: {detected hallucinations }
Your answer example:
{’hallucinations’: [...]}
Input:
Relevant document №1: {doc 1}
...
Relevant document №5: {doc 5}
Model input: {model input}
model output: {model output text}
Detected hallucinations: {detected hallucinations }
Your answer: