NCL-UoR at SemEval-2025 Task 3: Detecting Multilingual Hallucination and Related Observable Overgeneration Text Spans with Modified RefChecker and Modified SelfCheckGPT

Jiaying Hong[1], **Thanet Markchom**[2], **Jianfei Xu**[1], **Tong Wu**[3] and **Huizhi Liang**[1]
[1] School of Computing, Newcastle University, Newcastle upon Tyne, UK
[2] Department of Computer Science, University of Reading, Reading, UK
[3] Previously at School of Computing, Newcastle University, Newcastle upon Tyne, UK
hongjalynn@gmail.com, thanet.markchom@reading.ac.uk,
mr.xujianfei@gmail.com, tongwuwhitney@gmail.com, huizhi.liang@newcastle.ac.uk

Abstract

SemEval-2025 Task 3 (Mu-SHROOM) focuses on detecting hallucinations in content generated by various large language models (LLMs) across multiple languages. This task involves not only identifying the presence of hallucinations but also pinpointing their specific occurrences. To tackle this challenge, this study introduces two methods: Modified-RefChecker (MRC) and Modified-SelfCheckGPT-H (MSCGH). MRC integrates prompt-based factual verification into References, structuring them as claim-based tests rather than single external knowledge sources. MSCGH incorporates external knowledge to overcome its reliance on internal knowledge. In addition, both methods' original prompt designs are enhanced to identify hallucinated words within LLM-generated texts. Experimental results demonstrate the effectiveness of the approach, achieving a high ranking on the test dataset in detecting hallucinations across various languages, with an average IoU of 0.5310 and an average COR of 0.5669. The source code used in this paper is available at https://github.com/jianfeixu95/NCL-UoR.

1 Introduction

Large language models (LLMs) have significantly advanced in producing human-like text across various domains (Xiong et al., 2024; Zhao et al., 2024). However, one critical challenge remains: hallucinations—instances where the generated output contains logical inconsistencies, factual inaccuracies, or irrelevant information (Goodrich et al., 2019). These issues are particularly prominent in multilingual settings, where linguistic differences, cultural context, and the availability of external resources introduce additional complexities (Guerreiro et al., 2023). To address this issue, SemEval-2025 Task-3: the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes (Mu-SHROOM) (Vázquez et al., 2025) was introduced. This task involves identifying hallucinated text spans in LLM-generated outputs across multiple languages and LLMs.

To tackle this task, this work modifies two state-of-the-art methods: RefChecker (Hu et al., 2024) and SelfCheckGPT (Manakul et al., 2023). RefChecker detects fine-grained hallucinations by extracting claim triplets (subject, predicate, object) from LLM outputs and comparing them with prebuilt reference data, using text classification and aggregation rules. However, it cannot precisely locate hallucination positions and relies on fixed and incomplete references. The proposed modified RefChecker improves upon this by introducing prompt-based fact verification, structuring references as claim-based tests for greater flexibility, and enhancing hallucination detection by calculating hallucination probabilities and providing soft and hard labels for more precise analysis.

SelfCheckGPT detects hallucinations by prompting the same LLM for multiple responses and identifying inconsistencies. However, reliance on internal knowledge may fail when hallucinations are consistent. To address this, we modify SelfCheckGPT by incorporating external knowledge and enhancing the prompt design to identify specific hallucinated words rather than only their presence.

Overall, unlike the original RefChecker and SelfCheckGPT, which rely on static references and internal prompt-based self-consistency, respectively, our modified methods incorporate external knowledge retrieval and prompt-driven span-level verification to improve hallucination detection accuracy and granularity.

2 Related Work

Most recent approaches to detecting hallucinations in LLM outputs rely on prompting techniques, where the models evaluate the likelihood of hallucinations in their responses. For instance, Ka-

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 271–279

davath et al. (2022) proposed prompting LLMs to generate an answer and then predict the probability of its correctness. Manakul et al. (2023) introduced SelfCheckGPT, which compares an LLM-generated sentence against multiple alternative generations, asking the model to assess whether the original sentence is consistently supported. Friel and Sanyal (2023) presented ChainPoll, using detailed prompts to guide models in identifying hallucinations. Hu et al. (2024) proposed RefChecker, a retrieval-augmented evaluation method that checks the consistency of model outputs against retrieved external references, aiming to identify factual inconsistencies and hallucinations without relying solely on LLM self-judgment. However, most existing methods focus on detecting whether a text contains hallucinations or not. Identifying the specific parts of a text that are hallucinations remains an open research challenge. Therefore, in this work, we modified RefChecker and SelfCheckGPT, two state-of-the-art methods to handle this task.

3 Methodology

3.1 Modified-RefChecker (MRC)

MRC is an improved RefChecker, integrating CLAUDE (Anthropic, 2022) for enhanced functionality. Note that any LLM, including open-source ones, can be substituted. However, to ensure consistent and scalable evaluation, we adopt CLAUDE due to its multilingual support, API stability, and superior performance compared to open-source models in the original RefChecker (Hu et al., 2024). MRC consists of two key components: the Extractor for constructing references and the Checker for identifying hallucinated words along with their probabilities. Figure 1 shows the overview of MRC. The details of each component are described below.

Extractor Component This component retrieves external knowledge using keywords or keyphrases through the Google CSE (Custom Search Engine) API (Esraa Q. Naamha, 2023) (summarized search websites) and extracts claims from LLM responses, structured as triplets (subject, predicate, object), to form factual references. The extraction of claims utilizes the prompt design from RefChecker's Extractor (Hu et al., 2024) and is implemented using the Anthropic API (Anthropic, 2022). However, the verification and refinement of claims are also conducted through the CLAUDE API, with the prompt design as in Appendix A (Figure 5).

Checker Component The Checker component evaluates hallucinated words and their probabilities in the model output by validating them against references using prompts. The prompts guide the classification of hallucinations and define their probabilities. The prompt design is as in Appendix A (Figure 6). With the support of CLAUDE API (Anthropic, 2022), the results from Checker are mapped to the LLM output text, highlighting hallucinated words and generating soft labels and hard labels. Soft labels are based on the detected hallucination probabilities, while hard labels are determined by a threshold of 0.5 (probabilities > 0.5 are marked as hallucinations).

3.2 Modified-SelfCheckGPT-H (MSCGH)

MSCGH is based on the method proposed by Markchom et al. (2024). It consists of 4 steps: keywords/keyphrases extraction, context retrieval, prompt construction and hallucination detection. Figure 2 shows an overview of MSCGH. The details of each step are discussed in the following.

Keywords/Keyphrases Extraction To generate a context for each LLM output text, keywords/keyphrases in the input text are first identified. In this work, YAKE (Yet Another Keyword Extractor) (Campos et al., 2020) is adopted to extract keywords across multiple languages, as it is domain- and language-independent. However, some languages are not covered by this method. Therefore, to improve keyword extraction in different languages, Hugging Face models are used for specific languages to identify named entities, while SpaCy facilitates tokenization and stop word removal. A summary of the tools and models used is shown in Appendix B (Table 1). Furthermore, GPT-3.5 was also applied to directly extract keywords/keyphrases from the LLM input text.

Context Retrieval To retrieve a context based on each extracted keyword WikipediaAPI[1] (MediaWiki, 2024) and Google CSE API (Esraa Q. Naamha, 2023) are considered. These resources are chosen for their popularity and capability to provide reliable context (Trokhymovych and Saez-Trumper, 2021). Once contexts for individual keywords are retrieved, they are concatenated to form a complete context for the LLM output text.

Prompt Construction Two prompt designs for identifying hallucinated words are explored.

[1] https://pypi.org/project/Wikipedia-API/

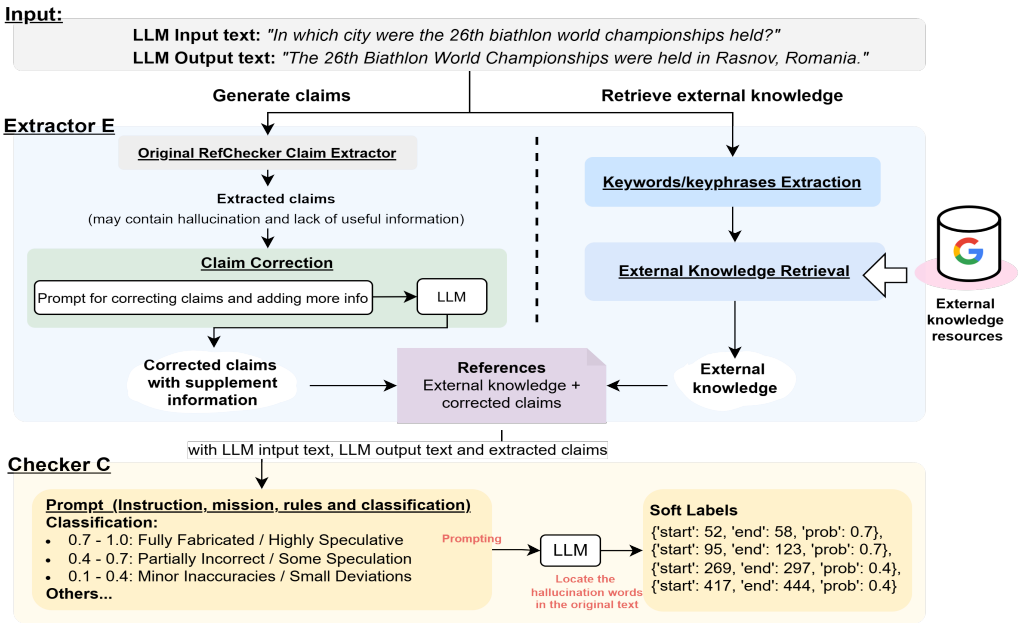


Figure 1: Overview of MRC

Prompt 1, adapted from SelfCheckGPT, is a simple prompt that asks an LLM to identify hallucinated words without specific instructions. Prompt 2, designed to identify hallucinated words, categorizes hallucination types and assigns probabilities while defining detection scope and output conditions. Compared to Prompt 1, it imposes stricter constraints to reduce unnecessary results (Rashkin et al., 2021). By directly classifying hallucinations and modeling probability distributions, it mitigates misalignment issues in LLM-generated text, improving detection accuracy and consistency.

Hallucination Detection To detect hallucination words, an LLM (this work considers GPT-3.5, GPT-4 and GPT-4o following the original methodology of using GPT models in SelfCheckGPT (Manakul et al., 2023).) is used to answer the prompt created in the previous step for each LLM output text. For each response, the hallucination words are identified, and a list of index intervals indicating the positions of these words in the LLM output string, $\mathcal{L} = \{L_1, L_2, \ldots, L_n\}$, is obtained. Then, all overlapping and adjacent intervals across all N responses are merged into a set of distinct, non-overlapping intervals $\mathcal{M} = \{(s_1, e_1), (s_2, e_2), \ldots, (s_m, e_m)\}$.

The soft probabilities for each merged interval are computed differently depending on the prompt used (Prompt 1 or Prompt 2). For Prompt 1, the probability of each merged interval (s_i, e_i) is computed by $p(s_i, e_i) = \frac{1}{n}\sum_{k=1}^{n} \frac{o_{i,k}}{e_i - s_i}$, where $o_{i,k}$ is the total overlap between the merged interval (s_i, e_i) and the intervals in the list L_k and $e_i - s_i$ is the length of the interval.

Prompt 2 detects the probabilities of hallucinated words. However, in the repeated N times process, the probabilities need to be recalculated, leading to the introduction of the following formula: $p(s_i, e_i) = \left(\frac{\sum_{k=1}^{n} o_{i,k} \cdot p_k}{\sum_{k=1}^{n} o_{i,k}}\right)^{1.2}$, where p_k is the probability of hallucination for each interval in the n responses, which is combined with the overlap length $o_{i,k}$ to calculate the weighted average probability. The exponent 1.2 introduces nonlinearity, giving higher importance to intervals with frequent overlaps and improving the accuracy of hallucination detection. All merged intervals in $\mathcal{M}$, along with their probabilities, serve as soft labels. Hard labels are obtained by selecting the intervals in $\mathcal{M}$ with probabilities higher than a predefined threshold, which is set to 0.5 in this work.

4 Datasets and Experimental Setup

Dataset The datasets used in this study are provided by the organizers of Mu-SHROOM. The validation set, which contains annotated labels, was used for model development and tuning. In the final experiments, the test set was employed to comprehensively evaluate the performance of the models. The validation set includes data in 10 languages, along with LLM input texts, LLM-generated texts, LLM tokens, corresponding logit values, and hallucination annotations in the form of soft and hard

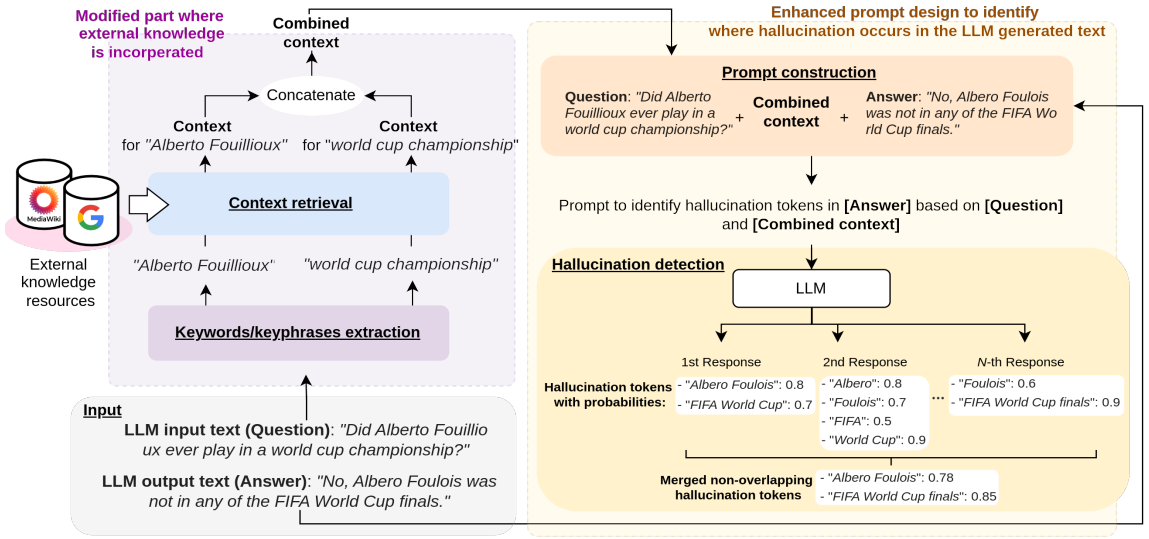


Figure 2: Overview of MSCGH

labels, indicating both the locations and probabilities of hallucinations. The test set contains data in 14 languages. The models were evaluated independently for each language to ensure a comprehensive assessment across multilingual data.

Method Selection Initially, evaluation was conducted on MRC and five variations of MSCGH , each utilizing different keyword extractors, context retrieval tools, prompt designs, and LLMs for hallucination detection (as discussed in Section 3.2). This resulted in six different models, each applied to 14 languages, for a total of 84 experiments. The details of these methods and their performance results can be found in Appendix C. Each model is assigned a Submitted Identifier, which corresponds to the Identifier submitted on the official website[2]. Based on the performance, the three best methods were selected for discussion: (1) **MRC_CLAUDE_CSE_A**: MRC using GPT-3.5 for keyword extraction, Google CSE API (abstract only) for context retrieval, and CLAUDE for hallucination detection. (2) **MSCGH_GPT_CSE_F**: MSCGH using GPT-3.5 for keyword extraction, full Google CSE API results, and GPT-4o for hallucination detection ($N = 5$). (3) **MSCGH_GPT_WIKI_A**: MSCGH using custom rules for keyword extraction, first 200 characters Wikipedia API results, and GPT-4o for hallucination detection ($N = 5$).

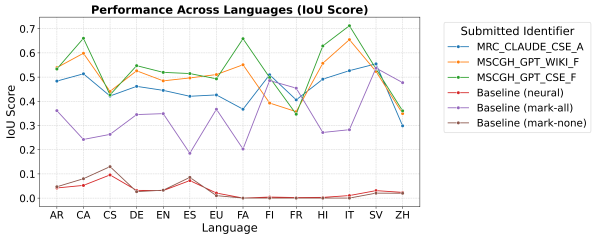
Evaluation Metrics The metrics provided by the organizers were used: **Intersection-over-Union (IoU) of Characters**: Measures the overlap between hallucinated characters marked in the gold reference and those predicted by the system, and **Probability Correlation**: Assesses how well the probability assigned by the system for a character being part of a hallucination correlates with the probabilities observed in human annotations.

Baseline Three baselines were provided in the task (Vázquez et al., 2025): (1) Baseline (neural): Fine-tuning of the neural network classifier based on XLM-R, outputting binary (0/1) probability predictions for each token, (2) Baseline (mark-all): Predicting all characters as hallucinations ($probability = 1$), and (3) Baseline (mark-none): Predicting all characters as non-hallucinations ($probability = 0$).

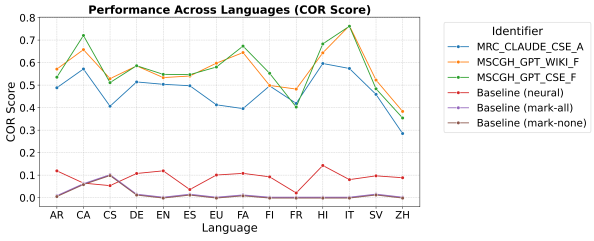
5 Results and Discussions

Overall comparison of the proposed methods The comparative results of our methods and the baselines are shown in Figures 3a and 3b. From these figures, the neural and mark-none baselines performed the worst across all languages, while the mark-all baseline achieved slightly higher IoU but nearly zero COR scores. In contrast, our method outperformed these baselines in all languages, with average improvements of approximately 0.30 in IoU and 0.45 in COR. More importantly, according to the 100,000 bootstrap resamplings mentioned in (Vázquez et al., 2025), our submitted methods achieved a $Pr(rank)$ above 0.5 in every language. This indicates a higher probability of outperforming the next-best team in the majority of samples,

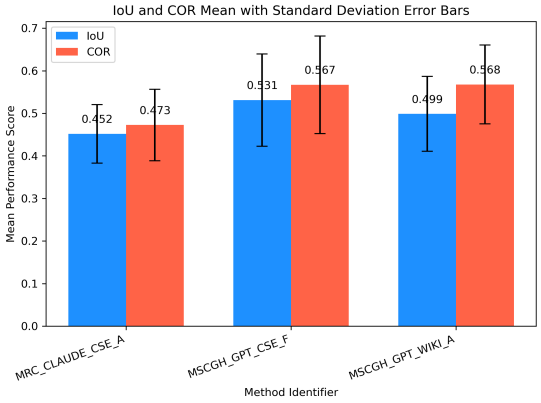
[2]https://helsinki-nlp.github.io/shroom/



(a) Performance Across Languages (IoU Score)



(b) Performance Across Languages (COR Score)



(c) Methods Performance and Stability Comparison

Figure 3: Performance Across Languages of Methods

and thus demonstrates robust and consistent cross-lingual performance.

Across multiple languages, our models exhibited distinct performance differences, as shown in Figure 3. MSCGH_GPT_CSE_F consistently led, benefiting from more effective keyword extraction, comprehensive external knowledge retrieval, and stronger hallucination detection. Its broader retrieval strategy provided an advantage in handling ambiguous or multi-step queries. MSCGH_GPT_WIKI_A followed closely, particularly excelling in Chinese results, where complex segmentation and word relationships were better handled through its customized keyword extraction. MRC_CLAUDE_CSE_A, while still effective, showed greater variance across languages, likely due to less optimized retrieval strategies or weaker hallucination detection.

Figure 3c presents the average IoU and COR scores across all languages, illustrating the overall performance and stability of the three methods. MSCGH_GPT_CSE_F achieved the highest IoU and COR scores, while MSCGH_GPT_WIKI_A performed similarly to MRC_CLAUDE_CSE_A. However, MSCGH methods exhibited larger error bars, indicating greater variability and less stability. The fluctuations in MSCGH may stem from differences in knowledge retrieval and prompt design. In contrast, MRC demonstrated more consistent performance, suggesting its higher stability.

Comparison of knowledge retrieval methods In Figure 3c, MSCGH_GPT_CSE_F outperformed MSCGH_GPT_WIKI_A. This could be attributed to differences in external knowledge resources and the precision of keyword extraction. GPT-3.5, as a keyword extraction tool, likely understood the context of questions better and extracted more precise and relevant keywords for retrieval. In contrast, custom rules had limitations in generalization and contextual understanding. They relied on specific language resources, which were limited in scope. This could affect the accuracy of keyword extraction and subsequently reduce the relevance and coverage of retrieved information. Besides keyword extraction, knowledge resources also played a vital role. The Google CSE API encompassed the Wikipedia API and extended beyond it, providing broader search coverage through a customizable search engine (Esraa Q. Naamha, 2023). Additionally, retrieving full-page content via the Google CSE API could yield better results than retrieving only abstract content, as suggested by MSCGH_GPT_CSE_F's superior performance over MRC_CLAUDE_CSE_A. Overall, both keyword extraction accuracy and knowledge coverage influenced model performance. This highlights the importance of optimizing external knowledge extraction methods to improve detection outcomes.

Comparison of prompted LLMs for hallucination detection Figure 4 compares different LLMs. In this figure, each model is represented with bars showing the IoU and COR scores for individual languages. The grey bar behind each model's score bars indicates the average IoU and COR scores for that model. From this figure, CLAUDE performed the worst, while GPT-4o showed significant improvements. However, not all GPT-4o-based methods outperformed CLAUDE, indicating that LLM upgrades alone do not guaran-

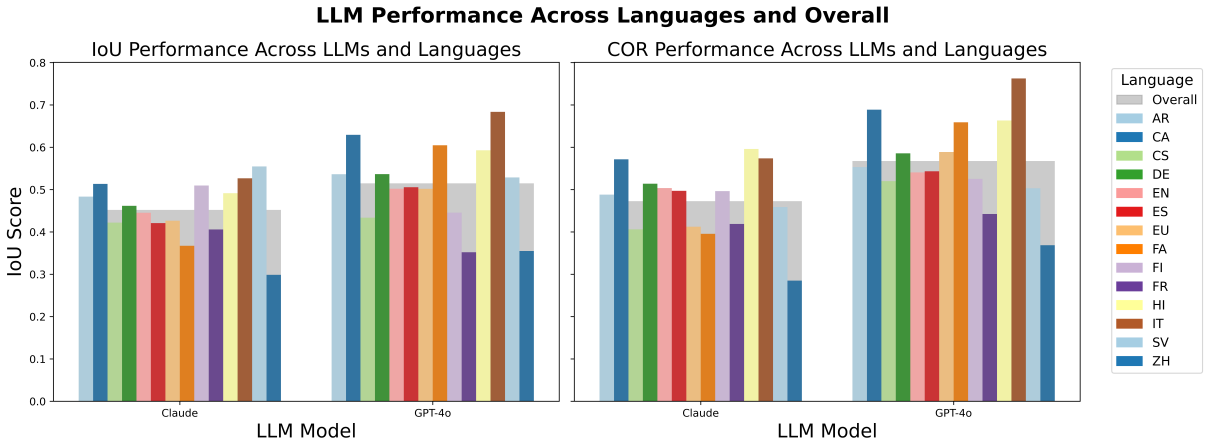


Figure 4: LLM Performance Across Languages and Overall

tee better results—effective knowledge retrieval remained essential. Performance gaps between LLMs were more pronounced in high-resource languages (e.g., Italian), where GPT-4o significantly outperformed CLAUDE. In contrast, for low-resource languages (e.g., Arabic), GPT-4o's benefits were inconsistent—some methods showed only marginal gains, while those using the Google CSE API achieved substantial improvements. This underscored the critical role of external knowledge integration in maximizing LLM performance.

6 Conclusion

SemEval-2025 Mu-SHROOM introduced the task of detecting hallucination spans in multilingual LLM outputs. To tackle this task, this work proposed two methods: Modified-RefChecker (MRC) and Modified-SelfCheckGPT-H (MSCGH). These methods incorporated external knowledge integration and an improved prompt design, enabling the detection of text-span hallucinations in LLM-generated texts. MRC and variations of MSCGH (with different keyword extraction techniques, external knowledge sources, and prompt strategies) were evaluated across datasets in 14 languages. Three top-performing methods were chosen for discussion in this paper. Among the evaluated methods, MSCGH using GPT-3.5 for keyword extraction, full Google CSE API results, and GPT-4o for hallucination detection achieved the best overall performance. Although MSCGH demonstrated higher performance, it lacked stability when applied across different languages. Meanwhile, MRC was more stable but less optimized. One limitation of the proposed approaches is the assumption that external knowledge is accurate. However, the retrieved information may not always be fully factual due to the ever-growing volume of online content. Such inaccuracies could reduce the effectiveness of the proposed approaches. Future research could focus on refining the prompt design and enhancing external knowledge integration and faulty correction strategies. Additionally, adaptive learning for low-resource languages and broader language task expansion could be considered.

References

AI4Bharat. *Indic NLP Library: Tokenization Module*.

Mohamed Taher Alrefaie. 2019. Arabic stop words.

Anthropic. 2022. Anthropic api documentation.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Matheel E. Abdulmunim Esraa Q. Naamha. 2023. Metadata Scraping Using Programmable Customized Search Engine. *Iraqi Journal of Computer, Communication, Control and System Engineering*, pages 10–25.

Nasim Fani. Hazm: Python library for persian nlp.

Robert Friel and Atindriyo Sanyal. 2023. Chainpoll: A high efficacy method for llm hallucination detection. *arXiv preprint arXiv:2310.18344*.

Ben Goodrich, Vinay Rao, Mohammad Saleh, and Peter J Liu. 2019. Assessing the factual accuracy of generated text.

Nuno M. Guerreiro, Duarte M. Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André F. T. Martins. 2023. Hallucinations in

large multilingual translation models. *Transactions of the Association for Computational Linguistics*, 11:1500–1517.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Refchecker: Reference-based fine-grained hallucination checker and benchmark for large language models.

Stopwords ISO. 2016. Stopwords iso - a collection of stopwords for various languages. https://github.com/stopwords-iso/stopwords-eu/tree/master.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know.

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

Thanet Markchom, Subin Jung, and Huizhi Liang. 2024. NU-RU at SemEval-2024 task 6: Hallucination and related observable overgeneration mistake detection using hypothesis-target similarity and SelfCheckGPT. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 253–260, Mexico City, Mexico. Association for Computational Linguistics.

MediaWiki. 2024. API:Main page.

Open-Source Toolkit. Gitcode repository: 63e0e.

Hannah Rashkin, Tal Linzen, and Tom McCoy. 2021. Increasing faithfulness in knowledge-grounded dialogue with controllable features. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 704–718.

Stanford NLP Group. Stanza: A python nlp library for many human languages.

Stopwords ISO Contributors. Stopwords iso: Multilingual stopwords collection.

Mykola Trokhymovych and Diego Saez-Trumper. 2021. Wikicheck: An end-to-end open source automatic fact-checking api based on wikipedia. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4155–4164, New York, NY, USA. Association for Computing Machinery.

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

Raúl Vázquez, Timothée 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. Part of SemEval-2025 Task 3; to appear in the Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025).

Feng Xiong, Thanet Markchom, Ziwei Zheng, Subin Jung, Varun Ojha, and Huizhi Liang. 2024. NCL-UoR at SemEval-2024 task 8: Fine-tuning large language models for multigenerator, multidomain, and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 163–169, Mexico City, Mexico. Association for Computational Linguistics.

Junzhe Zhao, Yingxi Wang, Huizhi Liang, and Nicolay Rusnachenko. 2024. NCL_NLP at SemEval-2024 task 7: CoT-NumHG: A CoT-based SFT training strategy with large language models for number-focused headline generation. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 261–269, Mexico City, Mexico. Association for Computational Linguistics.

A Prompts

This appendix presents the prompts used in RefChecker (MRC) and modified SelfCheckGPT (MSCGH). Figure 5 shows the Claims Correction Prompt, used in MRC. Figure 6 shows the Checker Component Prompt for MRC. Figure 7 shows Prompt 1 for MSCGH. Figure 8 shows Prompt 2 for MSCGH.

B Custom Rules of Keywords/keyphrases Extraction in MSCGH

Table 1 shows custom rules of keywords/keyphrases extraction across various languages in MSCGH.

C All Results

Table 2 shows the results of all the methods on the 14-language test set.

Table 1: Tools and Models Utilized for Keyword Extraction

Language	Stop Word Removal Tool	NER Model	Additional Model/Approach
Chinese (zh)	jieba and HIT_stopwords (Open-Source Toolkit)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (jieba.analyse)
Arabic (ar)	'stopwords-ar.txt' (Alrefaie, 2019)	Hugging Face ('asafaya/bert-base-arabic')	Tokenization (Hugging Face, TF-IDF)
Hindi (hi)	Indic NLP Library ('indic_tokenize') (AI4Bharat)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	Tokenization (Indic Tokenizer)
Basque (eu)	Stopwords-iso ('stopwords-eu.txt') (ISO, 2016)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'xx_ent_wiki_sm')
Czech (cs)	StopwordsISO (Stopwords ISO Contributors)	Stanza (Stanford NLP Group)	Tokenization (Stanza, TF-IDF)
Farsi (fa)	Hazm (Fani)	Hugging Face ('bert-fa-base-uncased-ner-arman')	Tokenization (Stanza, TF-IDF)
Catalan (ca)	spaCy (ca_core_news_sm)	Hugging Face ('projecte-aina/roberta-base-ca-v2-cased-ner')	TF-IDF (spaCy 'ca_core_news_sm')
English (en)	spaCy (en_core_web_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'en_core_web_sm')
Spanish (es)	spaCy (es_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'es_core_news_sm')
French (fr)	spaCy (fr_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'fr_core_news_sm')
German (de)	spaCy (de_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'de_core_news_sm')
Italian (it)	spaCy (it_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'it_core_news_sm')
Finnish (fi)	spaCy (fi_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'fi_core_news_sm')
Swedish (sv)	spaCy (sv_core_news_sm)	Hugging Face ('xlm-roberta-large-finetuned-conll03-english')	TF-IDF (spaCy 'sv_core_news_sm')

Prompt

System Task: Please expand, provide additional relevant factual information and verify about the following claim
Claims: {claims}
- If the claim is accurate, not hallucination and complete, return the original claim.
- If the claim is inaccurate, partial, or lacking detail, return a corrected, more detailed, and comprehensive factual statement.

Figure 5: Claims Correction Prompt for the Extractor Component of MRC

Prompt

System Task: Evaluate the model output text for hallucinations by comparing it to the provided references, existing fact, claims, and question (model input). Identify any hallucinated or potentially inaccurate parts in the entire model output text. Highlight the hallucinated word and assign a probability of the hallucination word in the 'model_output_text'.
LLM input text: {LLM input text}
Claims: {claims}
References: {references}
LLM output text: {LLM output text}
Instructions
1. Compare each claim with the provided references, question and existing fact (internal knowledge).
2. If a claim cannot be fully supported by the references, identify the hallucinated words and mark it to 'model output text'.
3. Return character-level offsets and assign hallucination probabilities.
4. If the claim is fully supported, hallucination should not to be labeled.
5. Assign hallucination probabilities based on the following criteria:
- 0.7 - 1.0: Fully fabricated or highly speculative content with no supporting evidence.
- 0.4 - 0.7: Partially incorrect or speculative content, but some evidence supports parts of the claim.
- 0.1 - 0.4: Minor inaccuracies, such as spelling errors, wrong formatting, or small factual deviations.
6. Ensure that the hallucinated words do not overlap or repeat. If overlapping occurs, merge them or separate them appropriately.
7. Ensure the words are shown in the 'model output text'.
8. Highlight text in 'model output text' that could potentially be a hallucination even if not explicitly listed in the claims.
9. Return all the hallucinated words or phrases and assign each a hallucination probability (between 0 and 1).
10. Do not filter out hallucinations based on low probability. Return results for any potential hallucination.
11. Do not include any explanations, summaries, or additional text. Return the JSON list directly.
12. Ensure all potential hallucinations are listed, even those with probabilities as low as 0.1.

Figure 6: Prompt for the Checker Component in MRC

Prompt 1

Context: {combined context}
Sentence: {LLM output text}
Which tokens in the sentence are not supported by the context above? Provide the answer in the form of a list of hallucination tokens separated by 'l' without accompanying texts.

Figure 7: Prompt 1 for the Hallucinations Detection in MSCGH

Prompt 2

Language: {language}
Question: {LLM input text}
Sentence: {LLM output text}
Context (if available): {context}
Task
You are an AI model output evaluation expert, responsible for detecting hallucinated words in model output and assigning accurate probability scores to each hallucination.
1. Identify hallucinated words or phrases in the model output based on the question and background knowledge.
- A word or phrase is considered a hallucination if it:
- Contradicts the background knowledge.
- Is unverifiable or fabricated.
- Contains logical inconsistencies.
2. Assign a probability score to each hallucinated word or phrase according to the following criteria:
- Probability > 0.7: Severe factual errors or contradictions.
- Probability 0.5 - 0.7: Unverifiable or speculative content.
- Probability 0.3 - 0.5: Minor inconsistencies or unverifiable details.
- Probability 0.1 - 0.3: Minor inaccuracies or vague ambiguities.
- Do not label words with probability ≤ 0.1 (i.e., verifiable facts).
Additional Instructions
- Do not mark redundant or overly generic words (e.g., "the", "a", "and") as hallucinations unless they introduce factual errors.
- Pay special attention to:
- Numerical data (e.g., dates, quantities, percentages).
- Named entities (e.g., people, organizations, locations).
- Logical contradictions (e.g., self-contradictions within the text).
- If background knowledge is absent, base your judgment solely on internal consistency.

Figure 8: Prompt 2 for the Hallucinations Detection in MSCGH

Table 2: All Methods Test Results

Language	Framework	Submitted Identifier	Keywords Extraction	External Knowledge	LLM	N	IoU	COR
AR	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.2485	0.2154
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4834	0.4881
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.3752	0.3707
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5389	0.5710
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.5334	0.5350
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4353	0.4539
CA	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.0.3650	0.3778
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.5135	0.5714
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.4849	0.5423
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5984	0.6573
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.6602	0.7202
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4621	0.6072
CS	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.2121	0.2364
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4218	0.4061
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.2513	0.3189
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.4409	0.5285
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.4264	0.5110
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.3935	0.4816
DE	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.3295	0.3713
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4617	0.5139
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.4173	0.4601
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5259	0.5852
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.5472	0.5860
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4467	0.5001
EN	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.4245	0.4544
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4451	0.5035
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.3690	0.3905
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.4844	0.5333
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.5195	0.5476
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4469	0.4690
ES	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.3129	0.3122
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4206	0.4970
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.3843	0.4104
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.4964	0.5402
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.5146	0.5464
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4240	0.4790
EU	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.3111	0.2833
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4263	0.4123
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.4340	0.4907
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5104	0.5974
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.4928	0.5802
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.3922	0.4932
FA	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.3254	0.3421
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.3672	0.3955
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.5027	0.5653
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5509	0.6444
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.6585	0.6732
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4034	0.5500
FI	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.2983	0.3114
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.5095	0.4964
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.3187	0.3656
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.3928	0.4982
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.4982	0.5523
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.3866	0.4906
FR	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.2094	0.2065
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4058	0.4187
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.3202	0.3685
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.3571	0.4822
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.3466	0.4024
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.3386	0.4712
HI	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.2251	0.1705
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.4914	0.5958
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.5606	0.6078
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5570	0.6433
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.6286	0.6830
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.5886	0.6664
IT	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.4153	0.4123
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.5265	0.5737
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.6563	0.6941
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.6547	0.7637
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.7122	0.7613
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.5950	0.7313
SV	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.3763	0.2863
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.5546	0.4587
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.4047	0.4335
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.5233	0.5224
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.5340	0.4836
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.4918	0.4907
ZH	MSCGH	NCL-UoR_Self_GPT3.5_YAKE_Wiki	YAKE	Wikipedia API	gpt-3.5-turbo	5.0	0.1683	0.2840
	MRC	NCL-UoR_CLAUDE-Modifier	gpt-3.5-turbo	Google CSE API (abstract)	CLAUDE-3-5-haiku-20241022	-	0.2986	0.2849
	MSCGH	NCL-UoR_SelfModify-H	custom rules (Table 1)	Wikipedia API	gpt-3.5-turbo	5.0	0.1849	0.2271
	MSCGH	NCL-UoR_SelfModify-H-plus	custom rules (Table 1)	Wikipedia API	gpt-4o	5.0	0.3492	0.3830
	MSCGH	NCL-UoR_Self_GPT4o_Google_CSE	gpt-3.5-turbo	Google CSE API	gpt-4o	5.0	0.3606	0.3539
	MSCGH	NCL-UoR_Self_GPT4_GPT3.5_Google_CSE	gpt-3.5-turbo	Google CSE API (abstract)	gpt-4-turbo	5.0	0.2842	0.3073