Detecting Hallucinations in Scientific Claims by Combining Prompting Strategies and Internal State Classification

Yupeng Cao[1,2], **Chun-Nam Yu**[1], **Koduvayur Subbalakshmi**[2]
[1]Nokia Bell Labs
[2]Stevens Institute of Technology
https://github.com/InfintyLab/SciHal-Challenge

Abstract

Large Language Model (LLM) based research assistant tools demonstrate impressive capabilities, yet their outputs may contain hallucinations that compromise their reliability. Therefore, detecting hallucinations in automatically generated scientific content is essential. SciHal2025: Hallucination Detection for Scientific Content challenge @ ACL 2025 provides a valuable platform for advancing this goal. This paper presents our solution to the SciHal2025 challenge. Our approach combines several prompting strategies to prompt LLMs and leverages their hidden states as features to build the classifier. We first benchmark multiple LLMs on the SciHal dataset under the zero-shot prompting. Next, we developed a detection pipeline that integrates few-shot and chain-of-thought prompting. Then, the hidden representations extracted from the LLMs serve as features for an auxiliary classifier, further improving detection performance. In this paper, we present comprehensive experimental results and discuss the implications of our findings for future research on hallucination detection in scientific content.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in generating scientific content across various domains (Le Scao et al., 2023; Thulke et al., 2024; Zhang et al., 2024; Zheng et al., 2025). LLM-powered research assistant tools further streamline scholarly workflows by answering research-related questions and output structured, concise responses. However, hallucinations may be introduced by LLMs pose a significant challenge to fully trusting these automatically generated scientific outputs (Alkaissi and McFarlane, 2023). Consequently, detecting hallucination content from the LLM-powered system is essential for their safe deployment.

Hallucination Detection for Scientific Content challenge (SciHal 2025) @ ACL 2025 provides a rigorous test platform for this problem (Li et al., 2025). The dataset contains real-user questions, retrieved scientific abstracts, LLM-generated responses, and extracted claims from responses with human annotation. The goal is to classify each claim based on the provided reference abstracts into different hallucination types. This paper describes our technical solution for the SciHal Challenge.

Our solution integrates prompting techniques with LLMs and leverages the models' internal representations for classification. We begin by benchmarking multiple LLMs on the SciHal dataset under zero-shot prompting to gauge their out-of-the-box performance. Subsequently, we develop a detection pipeline by combining domain-specific few-shot examples with Chain-of-Thought (CoT) prompting (Wei et al., 2022). Specifically, we first classify each data point into its respective domain, then pair it with corresponding domain-aware few-shot examples to construct refined CoT prompts. Then, the hidden states produced by the LLM serve as features for training a classifier, enhancing predictive accuracy. On the evaluation set, our approach achieves F1 scores of 0.59 on subtask 1 and 0.51 on subtask 2, as reported on the leaderboard. A detailed performance analysis is provided in Section 4.

2 SciHal Task Description

2.1 Problem Definition

The challenge aims to develop advanced LLMs that can identify hallucinations in scientific claims. Given a claim c to be verified, the model M will take the input query q, which includes claim c, reference r, and prompt instructions p. The model M then makes a classification $y = M(q[c; r; p])$:

- For subtask1, $y \in$ ['Unverifiable', 'Contradiction', or 'Entailment']
- For subtask2, $y \in$ ['Unrelated and unverifiable', 'Related but verifiable', 'Misrepresenta-

Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025), pages 316–327

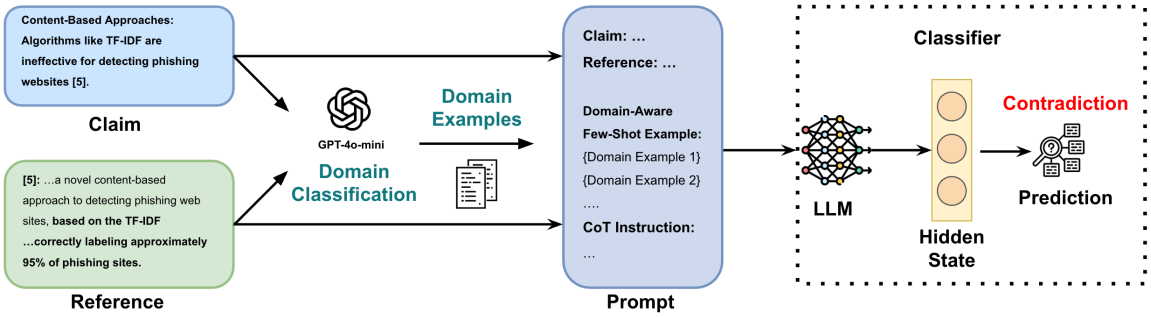

Figure 1: Overview of our method that detects the hallucination in scientific claims.

tion', 'Missing information', 'Numeric error', 'Entity error', 'Negation', 'Entailment']

Performance evaluation employs the weighted F1 score as the major evaluation metric.

2.2 Dataset

The dataset curation began with more than 50,000 real user questions spanning five domains: engineering, environmental science, medicine, agriculture & biological sciences, and computer science. After LLM paraphrasing and manual removal of sensitive information, 500 unique questions remained. For each question, the organizer fetched the 20 most relevant scientific abstracts through a retrieval-augmented generation (RAG) system. Answers were generated from these abstracts, broken into individual claims, and linked to their supporting references. Synthetic hallucinations were injected via targeted LLM prompts to balance the label distribution. Expert annotation combined with LLM labeling produced the final dataset $D = \{d_1, d_2, ..., d_n\}$ consisting of n data samples. Each data point is a six-tuple $d_i = (q, a, c, l, r, j)$ comprising the q-question, a-answer, c-claim, l-label, r-reference, and j-justification.

The organizers provided three training batches with identical data points but differing label sets for each subtask. Batch 1 data is a strict subset of batch 2 and was therefore discarded. All experiments in this paper are therefore conducted on batches 2 and 3 with a total of 3,592 data points. Table 1 presents the label distribution in the training set.

(a) Subtask 1

Label	Count	%
contradiction	1 369	38.1
unverifiable	890	24.8
entailment	1 333	37.1

(b) Subtask 2

Label	Count	%
negation	625	17.4
misinterpretation	395	11.0
related but unverifiable	738	20.5
entailment	1 333	37.1
entity error	174	4.8
unrelated and unverifiable	152	4.2
missing information	59	1.6
numeric error	116	3.2

Table 1: Distribution of ground-truth in both subtasks.

3 Methodology

In this section, we outline the proposed pipeline for hallucination detection in scientific claims (See in Figure 1). We first assign claims to their domain and select a few corresponding examples. By leveraging the in-context learning (ICL) capacity of LLMs (Radford et al., 2019; Brown et al., 2020; Dong et al., 2024), we then construct the detection pipeline that utilizes LLM's hidden states as features to train the classifier.

3.1 Domain-Aware Few-Shot Selection

Because claims and their supporting references span distinct fields, the specialized terminology and knowledge scope vary significantly. To exploit in-context learning more effectively, we first classify each data point d_i into its domain $t \in$ {engineering, computer science, environmental science, medicine, agriculture & biological sciences}. For the given data point d_i, we input the claim c and its associated reference r into the proprietary LLM (GPT-4o-mini) to determine the appropriate domain, formally expressed as $t = \text{LLM}(c, q)$.

After completing the domain assignment, each data point is updated to include its domain t, represented as $d_i = (q, a, c, l, r, j, t)$. Subtask 1 and Subtask 2 differ only in their labels, while the claim and reference remain the same. Thus, we did the domain classification once for both subtasks. After classifying all 3,592 data points by domain, we randomly sampled 100 to do a manual check for

quality control. GPT-4o-mini correctly labeled the vast majority. The only notable confusion occurred between 'computer science' and 'engineering' domains, whose content often overlaps. Therefore, the domain classification accuracy is adequate for pairing each claim with the appropriate few-shot examples and is utilized in the following steps. We have listed the domain statistics results in Table 8 of Appendix A.

3.2 Few-Shot Learning with Chain-of-Thought Prompting

We first design baseline few-shot prompts for subtasks 1 and 2 (in Appendix D.2). Specifically, we randomly select two data points from each label as examples and evaluate two prompting variants:

- **Few-Shot Prompt 1:** Provide two data examples for each label, each example consisting of a claim with its corresponding reference, and instruct the LLM to output the prediction directly.
- **Few-Shot Prompt 2:** Provide two data examples, each including the claim, reference, and justification. Instruct the LLM to first generate a justification and subsequently output the corresponding prediction.

Next, we utilize the domain classification results to refine our few-shot strategy. Given a data point d_i, we randomly select two examples per label based on their assigned domain t, and incorporate these domain-specific examples into the two prompt templates described above (whole prompt in Appendix D.3).

Building upon our few-shot prompts, we further incorporate Chain-of-Thought (CoT) prompting (Wei et al., 2022) to enhance model reasoning. For subtask 1, we structure the CoT prompt in four steps: 1) Read the reference abstract(s) carefully; 2) Read the scientific claim carefully; 3) Analyze the relationship between the claim and reference abstract(s); 4) Determine which single category best describes the relationship. Subtask 2 has more complex and fine-grained labels, so we leverage its tree label structure[1] to design the CoT prompt. We require the LLM to provide a detailed justification and respond to a checklist of diagnostic questions before assigning a label. This checklist is illustrated in Figure 5. The combination of justification and checklist-based reasoning exemplifies the application of CoT prompting. All CoT prompt templates can be found in the Appendix D.4.

3.3 Prompting Strategies with Internal State Classification

The few-shot learning approach above only uses a very limited number of labeled examples, and it also doesn't take into account the relative frequencies of each target class. As a refinement of the few-shot prompting approach above, we study the use of the internal states of LLMs for hallucination detection. The internal states of LLMs have been used to detect hallucinations in many studies (Azaria and Mitchell; Marks and Tegmark). Specifically, we take the last layer hidden state vector of the LLM model at the last generated token (the customary choice for finetuning causal LLMs for classification), and train a logistic regression model on top of it. Note that we do not perform any fine-tuning on the LLM parameters. We just take the hidden state vector as a fixed representation and train a classifier on it. We use the "justification+label" template for subtask 1 and "justification+checklist+label" template for subtask 2 from above.

4 Experiments and Results

In this section, we present a detailed analysis of our experimental results and discussion. Specific details regarding experiment setup and configurations are provided in the Appendix B.

4.1 Zero-shot benchmark results

We first evaluate several widely used LLMs on subtask 1 under zero-shot prompting. This initial benchmarking enables us to gain insight into the baseline performance and comparative strengths of different LLMs on the SciHal challenge. For efficiency, we use accuracy as the evaluation metric. The results are presented in Figure 2, with the detailed prompt in the Appendix D.1.

From Figure 2, we can find that the instruct models consistently outperform their base model, and models with larger parameter sizes achieve even better performance. Consequently, for subsequent experiments, we selected Llama3.1-8B-Instruct, Llama3.1-70B-Instruct, and Llama3.3-70B-Instruct as our primary evaluation models, effectively covering a range of parameter sizes.

We also observe that in the zero-shot setting,

[1] https://www.kaggle.com/competitions/hallucination-detection-scientific-content-2025/overview

Model & Prompt	Batch 2 Data		Batch 3 Data	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
Subtask 1				
Llama3.1-8B-Instruct, ref + label (Few-Shot Prompt 1)	29.76	31.42	20.08	22.61
Llama3.1-8B-Instruct, ref + just + label (Few-Shot Prompt 2)	60.30	64.57	35.82	37.15
Llama3.1-8B-Instruct, ref + just + subj + label (Domain-Aware Few-Shot)	61.43	63.32	36.50	39.41
Llama3.1-70B-Instruct, ref + label (Few-Shot Prompt 1)	62.16	65.43	43.10	45.50
Llama3.1-70B-Instruct, ref + just + label (Few-Shot Prompt 2)	**73.46**	**75.16**	53.50	54.33
Llama3.1-70B-Instruct, ref + just + subj + label (Domain-Aware Few-Shot)	72.36	74.71	54.62	57.13
Llama3.1-70B-Instruct, Domain-Aware Few-Shot + CoT	70.03	71.63	51.02	53.28
Llama3.3-70B-Instruct, ref + just + subj + label (Domain-Aware Few-Shot)	**73.61**	**75.52**	**61.20**	**64.79**
Subtask 2				
Llama3.1-8B-Instruct, ref + label (Few-Shot Prompt 1)	31.23	36.27	-	-
Llama3.1-8B-Instruct, ref + just + label (Few-Shot Prompt 2)	43.98	48.16	-	-
Llama3.1-8B-Instruct, ref + just + checklist + label	30.50	34.15	-	-
Llama-3.1-70B-Instruct, ref + label (Few-Shot Prompt 1)	57.18	68.78	-	-
Llama-3.1-70B-Instruct, ref + just + label (Few-Shot Prompt 2)	**62.97**	**72.88**	38.10	43.25
Llama-3.1-70B-Instruct, ref + just + subj + label (Domain-Aware Few-Shot)	60.28	70.15	36.97	39.15
Llama-3.1-70B-Instruct, ref + just + checklist + label	54.48	59.24	-	-

Table 2: Few-shot and CoT results for Subtasks 1 and 2. 'ref' denotes reference, 'just' denotes justification, and 'subj' signifies domain-matched few-shot examples.

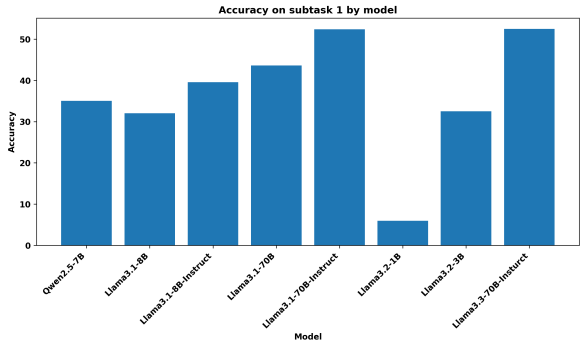

Figure 2: LLMs zero-shot performance on subtask 1.

LLMs exhibit poor performance. Even the 70B-parameter model achieves an accuracy of only around 50%, underscoring the inherent complexity of the task and emphasizing the necessity for continued development of more advanced methods.

4.2 Few-shot with CoT results

Batch 2 Results. From Table 2 'Batch 2 Data' column, we observe that requesting the LLM to provide a justification prior to outputting the label significantly enhances the F1 scores for Subtask 1, particularly for the 70B models. A similar trend is evident for Subtask 2, where prompting for justification also results in improved accuracy. Moreover, performance is further enhanced when employing domain-aware few-shot examples rather than standard few-shot prompts.

However, asking the LLM model to go through a checklist of questions before outputting the label actually degrades performance for both the 8B and 70B models in subtask 2. We examined the results more carefully and found that with the checklist, the LLM models tend to predict the class "missing information" a lot more frequently when it is only a very small class (10 examples out of 2092), leading to a drop in accuracy. We also find it very difficult as humans to distinguish between the two classes "related but unverifiable" and "missing information" in subtask 2. We tried to ask the organizers for clarification of their definitions but could not get an answer. If we merge these two classes and re-run our experiments with the 70B model, we obtain results from Table 3. We can see that there are consistent improvements from adding a checklist on top of justifications.

Model & Prompt	Macro-F1	Micro-F1
Llama3.1-70B-Instruct, ref + label	62.48	70.59
Llama3.1-70B-Instruct, ref + just + label	71.49	76.66
Llama3.1-70B-Instruct, ref + just + checklist + label	72.55	77.05

Table 3: Subtask 2 with 7 classes ('missing information' merged with 'related but unverifiable').

Batch 3 Results. From Table 2 'Batch 3 Data' column, the results are largely consistent with Batch 2, with improvements using logistic regression on the hidden state vectors, except for macro-f1 on Subtask 2 due to the smaller categories. Additionally, due to the timing of data release, constraints imposed by the competition schedule, and limited computational resources, we were unable to complete all planned experiments for subtask 2.

Comparing batch 2 and batch 3 of the training data we notice there is a large drop in the performance. We believe this is due to the differences in how batch 2 and batch 3 are collected. Both batch 2 and batch 3 are labeled by a subject matter

expert (SME) and an LLM. If the SME and the LLM agree, then the data point goes to batch 2. If there is a disagreement, another SME is requested to label the example, and it goes to batch 3 and the test set. So batch 3 and the test set contain more difficult examples compared to batch 2. However, despite the labeling process by multiple SMEs, we still find some labels that we disagree with in batch 3, which we will share in the error analysis section.

4.3 Internal State Classification

We use 80% of the data as the training set, and 20% as evaluation data. Table 4 shows the result of logistic regression on top of the internal state vectors. We can see that with or without merging the two classes "missing information" and "related but unverifiable", the logistic regression improves upon the subtask 2 results based on few-shot prompting only in Tables 2. The corresponding results for subtask 1 are also much improved.

	Macro-F1	Micro-F1
Subtask 1, Llama-3.1-70B-Inst, Batch 2	86.20	87.11
Subtask 1, Llama-3.1-70B-Inst, Batch 3	60.21	62.00
Subtask 2, Llama-3.1-70B-Inst, Batch 2	70.60	82.81
Subtask 2, Llama-3.1-70B-Inst, Batch 2 (merged labels)	79.05	81.14
Subtask 2, Llama-3.1-70B-Inst, Batch 3	36.52	52.33

Table 4: Subtask 1 and subtask 2 with logistic regression on internal state vectors.

We also perform ablation studies on the token location used for extracting hidden states for logistic regression. We compare using the hidden states from the last generated token (our current proposal) with the hidden states from the last token from the prompt (i.e., no generation). Using the hidden states from the last token of the prompt is a common finetuning strategy used for adapting causal language models to classification tasks. From Table 5 we can observe that using the hidden states of the last generated token is better than using the hidden states of the last prompt token, especially for Task 2. This shows the power of combining the generation capabilities of the LLMs together with finetuning in detecting hallucinations, which is better than using few-shot learning generation or finetuning alone.

	Macro-F1	Micro-F1
Task 1, Llama-3.1-70B-Inst, last prompt token	56.84	59.33
Task 1, Llama-3.1-70B-Inst, last generated token	60.21	62.00
Task 2, Llama-3.1-70B-Inst, last prompt token	25.18	45.33
Task 2, Llama-3.1-70B-Inst, last generated token	36.52	52.33

Table 5: Comparison of logistic regression result using last prompt token and last generated token on Batch 3.

4.4 Leaderboard Results

Based on the experimental results presented above, we evaluate the proposed pipeline on the test data. The leaderboard results are shown in Table 6. Our results are at the top-2 of subtask 1 and top-1 of subtask 2 on the leaderboard as of 10 PM EST on June 20. The results obtained from the leaderboard are consistent with the trends observed in the training dataset. These results indicate that our proposed pipeline demonstrates robustness and effectiveness.

Model & Prompt	Score
Subtask 1	
Llama-3.1-70B-Inst, Few-Shot Prompt 2	0.49
Llama-3.3-70B-Inst, Domain-Aware Few-Shot	0.55
Llama-3.3-70B-Inst, Domain-Aware Few-Shot + CoT	0.54
Llama-3.1-70B-Inst, Few-Shot Prompt 2 + Log-Reg on hidd-stat	**0.59**
Llama-3.1-70B-Inst, Domain-Aware Few-Shot + Log-Reg on hidd-stat	**0.59**
Subtask 2	
Llama-3.1-70B-Inst, Few-Shot Prompt 2	0.40
Llama-3.1-70B-Inst, Few-Shot Prompt 2 + checklist	0.47
Llama-3.1-70B-Inst, Few-Shot Prompt 2 + Log-Reg on hidd-stat	**0.51**

Table 6: Leaderboard scores for each subtask.

4.5 Error Analysis

We first analyzed the experiment results by using subtask 1, Batch 3 Data with Domain-Aware Few-Shot setting, and show the result in Table 7. The analysis indicates that the *entailment* class achieved the highest recall and overall F1-score, demonstrating that it was the easiest category for the model to identify accurately. Conversely, the class *unverifiable* exhibited the lowest recall and F1-score, highlighting its difficulty for classification.

Class	Precision	Recall	F1-score
Contradiction	0.657	0.462	0.542
Entailment	0.547	0.861	0.669
Unverifiable	0.550	0.238	0.332

Table 7: Classification metrics (precision, recall, and F1-score) for each class.

Following these findings, we conducted a detailed analysis of the data and labels to assess dataset quality. The complete results of this error analysis are provided in the Appendix C.

5 Conclusion

In this paper, we present our solution to the SciHal 2025 challenge. By integrating domain-aware few-shot and CoT prompt, and the model's hidden state as the feature, our method achieved promising results. Due to time constraints, additional experiments are ongoing and will be reported later.

Limitations

First, our current experiments were conducted exclusively using open-source models; proprietary models have not yet been evaluated on this dataset. Second, due to time constraints, several fine-tuning experiments remain ongoing. We plan to continue these experiments beyond the current submission and will provide additional results and in-depth analyses later. Finally in our preliminary evaluations with training data batch 3, the macro and micro f1 scores are close to the numbers on the leaderboard but much lower than those from batch 2 reported above. This suggests our results can be sensitive to shifts in distribution and composition of different classes.

Ethics Statement

The authors take full responsibility for the proposed method. The proposed method is intended for academic and educational purposes only and is not a substitute for a professional system. The data accessed from this challenge is solely for academic purposes and will not be shared or disseminated.

References

Hussam Alkaissi and Samy I McFarlane. 2023. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2).

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, page 1, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Dan Li, Bogdan Palfi, and Colin Kehang Zhang. 2025. Hallucination detection for scientific content. https://kaggle.com/competitions/hallucination-detection-scientific-content-2025. Kaggle competition.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*.

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

David Thulke, Yingbo Gao, Petrus Pelser, Rein Brune, Rricha Jalota, Floris Fok, Michael Ramos, Ian van Wyk, Abdallah Nasir, Hayden Goldstein, et al. 2024. Climategpt: Towards ai synthesizing interdisciplinary research on climate change. *arXiv preprint arXiv:2401.09646*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. 2024. A comprehensive survey of scientific large language models and their applications in scientific discovery. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8783–8817.

Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Anh TN Nguyen, Lauren T May, Geoffrey I Webb, and Shirui Pan. 2025. Large language models for scientific discovery in molecular property prediction. *Nature Machine Intelligence*, pages 1–11.

A Domain-Aware classification results

We performed domain classification on the train set batch 2 and batch 3, and the results are presented in Table 8. The distribution of data points across the five scientific domains is relatively balanced, with no substantial differences in data representation observed.

Domain	Count	% of total
Computer Science	713	19.8%
Medicine	801	22.3%
Engineering	756	21.0%
Environmental Science	780	21.7%
Agricultural&Biological Science	542	15.1%

Table 8: Distribution of data across scientific domains.

B Experiment Setup

At the outset, we selected eight widely-used LLMs for our zero-shot experiments: Qwen2.5-7B, LLaMA3.1-8B, LLaMA3.1-8B-Instruct, LLaMA3.1-70B, LLaMA3.1-70B-Instruct, LLaMA3.2-1B, LLaMA3.2-3B, and LLaMA3.3-70B-Instruct. Based on their performance, we subsequently select LLaMA3.1-8B-Instruct, LLaMA3.1-70B-Instruct, and LLaMA3.3-70B-Instruct for further experiments. All models were sourced from Hugging Face.

To ensure experimental reproducibility, we standardized inference parameters as follows: maximum output tokens set to 1024, temperature set to 0.6, and top-p sampling set to 0.9. All experiments were conducted using two NVIDIA H100 GPUs.

C Error Analysis

The experiment results show that performance does not consistently improve when the advanced prompts are employed (e.g. CoT prompt). Therefore, we conducted an error analysis to better understand the results.

C.1 Error Analysis on subtask 1 results

We first use the Subtask 1 & Batch 3 Data with Domain-Aware Few-Shot results to do the error analysis.

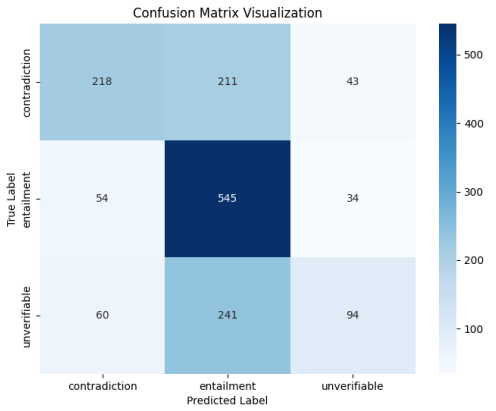

Figure 3: Confusion Matrix on subtask 1 results.

From the Figure 3, it reveals that the class *entailment* is the easiest for the model to correctly predict, exhibiting the highest accuracy. Conversely, the *unverifiable* class poses the greatest challenge, frequently misclassified as either *entailment* or *contradiction*.

C.2 Error Analysis on subtask 2 results

We then use the Subtask 2 & Batch 3 Data with Domain-Aware Few-Shot results to do the error analysis.

Class	Precision	Recall	F1-score
Entailment	0.550	0.814	0.656
Entity error	0.730	0.383	0.503
Misinterpretation	0.188	0.307	0.232
Missing information	0.333	0.041	0.073
Negation	0.447	0.328	0.378
Numeric error	0.550	0.440	0.489
Related but unverifiable	0.546	0.035	0.066
Unrelated and unverifiable	0.230	0.473	0.310

Table 9: Classification performance metrics for each class in Subtask 2.

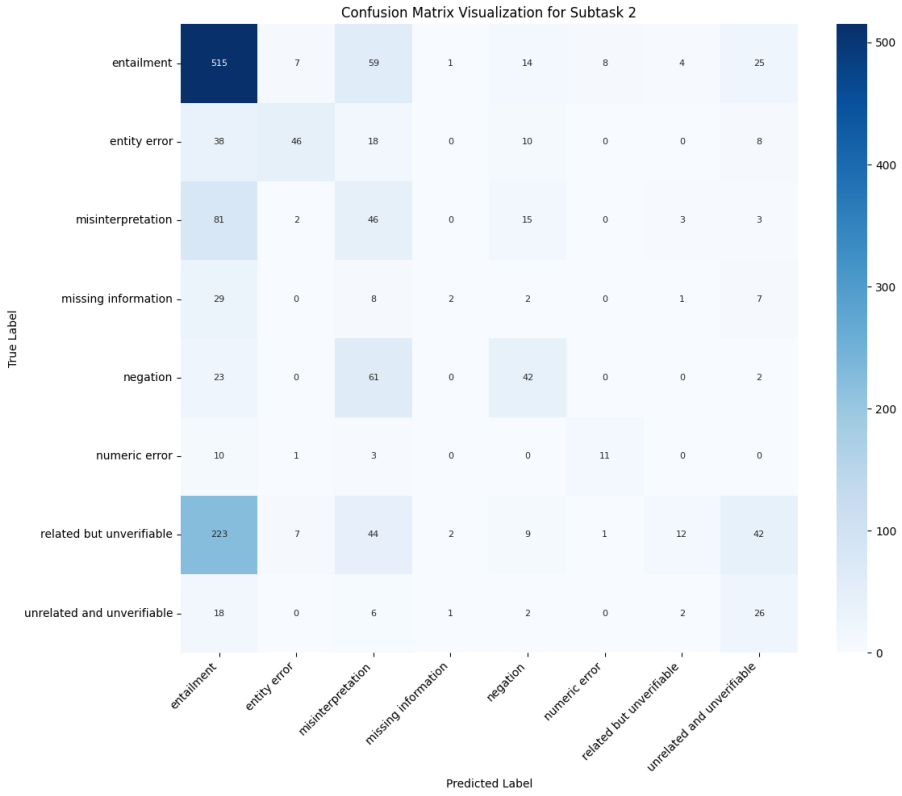

Figure 4: Confusion Matrix on subtask 2 results.

Table 9 and Figure 4 summarize the performance for Subtask 2, highlighting the strengths and challenges across different categories. Same with subtask 1, the *Entailment* achieves notably high recall (0.814) and the best F1-score (0.656), suggesting that the model effectively identifies instances belonging to this class. In contrast, the classes *Missing information* and *Related but unverifiable* exhibit extremely low recall (0.041 and 0.035, respectively), reflecting significant difficulty for accurate detection.

Additionally, Figure 4 reveals that many cases labeled as *Related but unverifiable* are misclassified as *Entailment*, likely due to subtle semantic overlaps between these categories. Similarly, the model frequently confuses *Misinterpretation* and *Negation* with *Entailment*, suggesting that nuanced distinctions among these classes pose considerable challenges. These findings underline the need for clearer category definitions and suggest that future model improvements may benefit from targeted fine-tuning or additional domain-specific examples for the most challenging classes.

C.3 Data Sample Analysis

Inspired by the confusion matrix results, we further checked the data provided by the challenge and identified instances of conflicting labels. For example, as illustrated in the figure below, both the claim and the reference discuss medical image processing; however, the content is unrelated. The claim focuses explicitly on conclusions related to ResUNet, whereas the reference addresses automatic segmentation of ultrasound breast lesions. Although they share the general domain of medical imaging, their specific topics differ significantly, rendering the reference insufficient to verify the claim. Consequently, the correct classification should be “unrelated and unverifiable.” Our pipeline made the correct prediction, and subsequent validation by three human experts unanimously supported this classification. Nonetheless, the original dataset label was “related but unverifiable."

This case demonstrates that subjective understanding of the term "related" can impact classification results. Such instances underscore the inherent complexity of accurately labeling data in the task.

A data example from Subtask 2

Claim: - ResUNet, on the other hand, does not rely on such initial conditions and is more robust to variations in image quality. Level-Set Techniques: While level-set methods can capture complex boundaries, they often struggle with initialization sensitivity and computational efficiency [5, 6].

---

reference: - "[5]: Automatic segmentation of ultrasonographic breast lesions is very challenging, due to the lesions' spiculated nature and the variance in shape and texture of the B-mode ultrasound images. Many studies have tried to answer this challenge by applying a variety of computational methods including: Markov random field, artificial neural networks, and active contours and level-set techniques. These studies focused on creating an automatic contour, with maximal resemblance to a manual contour, delineated by a trained radiologist. In this study, we have developed an algorithm, designed to capture the spiculated boundary of the lesion by using the properties from the corresponding ultrasonic image. This is primarily achieved through a unique multi-scale texture identifier (inspired by visual system models) integrated in a level-set framework. The algorithm's performance has been evaluated quantitatively via contour-based and region-based error metrics. We compared the algorithm-generated contour to a manual contour delineated by an expert radiologist. In addition, we suggest here a new method for performance evaluation where corrections made by the radiologist replace the algorithm-generated (original) result in the correction zones. The resulting corrected contour is then compared to the original version. The evaluation showed: (1) Mean absolute error of 0.5 pixels between the original and the corrected contour; (2) Overlapping area of 99.2% between the lesion regions, obtained by the algorithm and the corrected contour. These results are significantly better than those previously reported. In addition, we have examined the potential of our segmentation results to contribute to the discrimination between malignant and benign lesions.[6]: In order to improve the accuracy of breast ultrasound image segmentation, an ultrasound image segmentation method using the C-V (Chan-Vese) model based on phase is proposed. First, the ultrasound image is filtered by LOG-Gabor filters in six different orientations, and the phase feature of the image is obtained by extracting the phase information in the orientation with the maximum energy. Then, the SRAD(speckle reducing anisotropic diffusion) method is used to reduce the noise of the ultrasound image, and the processed image is multiplied by the phase features to enhance the contrast of the target and background. Finally, the target of the ultrasound image is identified by the segmentation algorithm using the C-V model, and corrosion is applied to make the edge smooth and complete. The experimental results show that compared with the C-V model and GAC (geodesic active contour) model based on image gray and the ANN (artificial neural networks) method based on phase feature, the proposed method can obviously improve the accuracy of breast ultrasound image segmentation, which is 92.40%."

---

label: - "related but unverifiable"

---

prediction: - "unrelated and unverifiable"

D Prompt Set

D.1 Zero-Shot prompt

We first employed Zero-Shot prompting to evaluate multiple LLMs and establish their baseline performance on this challenge. The detailed prompts used for Zero-Shot evaluation are detailed below:

Zero-Shot prompt for subtask 1

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from an academic paper, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here is the definition of each label:
entailment: The claim is supported by the reference.
contradiction: The claim is contradicted by the reference.
unverifiable: The claim cannot be verified by the reference
- You MUST strictly output your result in the following JSON format (and nothing else).
Now it's your turn.

D.2 Baseline Few-Shot Prompts

We first designed two baseline few-shot prompts as follows. We illustrate the prompting using Subtask 1 as the example; the prompt structure for Subtask 2 is the same, selecting two examples for each corresponding label.

Baseline Few-Shot prompt 1 (ref + label)

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from an academic paper, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here are some examples:
Example 1: #Claim: {...}; #Reference: {...}; #Label: {contradiction}
Example 2: #Claim: {...}; #Reference: {...}; #Label: {contradiction}
Example 3: #Claim: {...}; #Reference: {...}; #Label: {entailment}
Example 4: #Claim: {...}; #Reference: {...}; #Label: {entailment}
Example 5: #Claim: {...}; #Reference: {...}; #Label: {unverifiable}
Example 6: #Claim: {...}; #Reference: {...}; #Label: {unverifiable}
- Now, apply the same pattern:
Input: #Claim: {...}; #Reference: {...};
Output:

Baseline Few-Shot prompt 2 (ref + justification + label)

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from an academic paper, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here are some examples:
Example 1: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 2: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 3:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 4:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 5: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
Example 6: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
- Now, apply the same pattern:
- Please output the justification and then make a prediction.
Input: #Claim: {...}; #Reference: {...};
Output:

D.3 Domain-Aware Few-Shot Prompt

We first classified each data point into its respective domain. Within each domain, we selected two examples per label to serve as domain-specific few-shot prompts. Given a claim requiring verification, we identify its domain and provide corresponding examples from that domain. Below, we illustrate this process using the domain of computer science as an example.

Domain-Aware Few-Shot Prompt (ref + justification + subj + label)

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from **{Computer Science} domain**, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here are some examples about {Computer Science} domain:
Computer Science Example 1: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Computer Science Example 2: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Computer Science Example 3:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Computer Science Example 4:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Computer Science Example 5: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
Computer Science Example 6: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
- Now, apply the same pattern:
- Please output the justification and then make a prediction.
Input: #Claim: {...}; #Reference: {...};
Output:

D.4 Details on Few-Shot learning with Chain-of-Thought prompts

Few-Shot learning with Chain-of-Thought prompt for subtask 1

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from an academic paper, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here are some examples:
Example 1: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 2: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 3:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 4:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 5: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
Example 6: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
- **When you classify a claim, please follow these steps:**
1. Read the reference abstract(s) carefully.
2. Read the scientific claim carefully.
3. Analyze the relationship between the claim and reference abstract(s).
4. Determine which single category best describes the relationship.
- Now, apply the same pattern:
- Please output the justification and then make a prediction.
Input: #Claim: {...}; #Reference: {...};
Output:

Few-Shot learning with Chain-of-Thought prompt for subtask 2 (ref + just + checklist + label)

System Prompt: - You are an assistant for claim verification. Given a claim and some reference from an academic paper, please classify the claim into three labels: contradiction, entailment, or unverifiable.
- Here are some examples:
Example 1: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 2: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {contradiction}
Example 3:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 4:#Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {entailment}
Example 5: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
Example 6: #Claim: {...}; #Reference: {...}; #Justification: {...}; #Label: {unverifiable}
- **When you classify a claim, please follow the checking list:**
1. Is the claim related to the references?
2. Does the claim contain a contradiction to the references?
3. Does the claim negate parts of the references or replaces terms with their antonyms?
4. Does the claim present logical fallacies, flawed reasoning (over-claiming, under-claiming, ambiguity, or inconsistency), or illogical conclusions?
5. Does the claim contain an erroneous numeric value?
6. Does the claim contain an erroneous entity?
7. Does the claim omit critical parts from the references, changing the meaning/intent?
8. Can the claim be supported by the references?
- Now, apply the same pattern:
- Please output the justification and then make a prediction.
Input: #Claim: {...}; #Reference: {...};
Output:

Figure 5: Checking list in Chain-of-Thought prompting.