Scientific Discovery

An attempt to use LLM to simulate Scientific Fact Verification

By Visalakshi Iyer — Submitted to INFO 555: Applied NLP
visalakshiiyer@arizona.edu

Abstract

Scientific claim verification is a task in natural language processing (NLP) that involves assessing the veracity of claims based on available evidence (scientific literatures). This project investigates the use of large language models (LLMs) to generate simulation code aimed at verifying scientific claims, utilizing the SciFact dataset—a collection of 1,409 expert-written claims paired with evidence-containing abstracts(Wadden et al.2020).

The methodology involved working on the SciFact training subset to include only claims accompanied by cited evidence. After performing some annotations to segregate the claims into SIX categories, LLMs were employed to generate simulation code for these claims.(Wang et al.2024) The experiment highlights several challenges in scientific claim verification. One major hurdle is the dependency on highly specific and accurate scientific knowledge to construct meaningful simulations. Our LLM-based system successfully generated simulations for 58% of the dataset, showcasing the model's ability to identify and utilize relevant evidence for simulation creation. However, the results also underscore the limitations of LLMs when dealing with incomplete or ambiguous scientific context, which significantly impacts the accuracy of generated simulations. This project is uploaded on:
GitHub Link:
applied-nlp-project3 [1]

[1]GitHub: github.com/visalakshi2001/applied-nlp-project3

1 Introduction

Scientific claim verification is an essential task in natural language processing (NLP), aiming to assess the validity of claims based on scientific evidence. With the exponential growth of research publications, automating this verification process has become crucial to combat misinformation and provide reliable scientific insights. This project focuses on integrating simulation-based methodologies with Large Language Models (LLMs) to validate scientific claims, particularly leveraging the SciFact dataset as a robust source of annotated scientific claims and evidence.

SciFact contains a robust structure, which integrates claims, reference texts, and annotated evidence, enabling a comprehensive exploration of LLM capabilities. Additionally, we filtered and manually labeled the dataset into six distinct categories—biology, cellular, directional, population, medicine, and epidemiology—to better understand the contextual diversity of scientific claims and tailor simulations accordingly.

Following the filtering and labeling process, a novel approach was employed, wherein LLMs were prompted to generate Python-based simulations using a base template, reference texts, and gold evidence sentences. The generated simulations aimed to model the scientific phenomena described in the claims. The outputs of these simulations were further analyzed by the LLM to produce definitive verification results, categorizing the claims as "Supported" or "Refuted."

During the experiment, simulation code generation was performed on 564 scientific claims of which simulations were successfully generated for 58% of the dataset. Results demonstrated that the success of simulations largely depended on the availability of domain-specific scientific information. While the LLM effectively generated simu-

lations for a significant portion of the dataset, inaccuracies arose in cases where critical scientific details were missing or ambiguous.

2 Data

2.1 Data Labeling

The SciFact dataset, designed for scientific claim verification, serves as the foundational corpus for this project. To enhance the dataset's utility for simulation-based fact-checking, I applied a multi-step filtering and annotation process on the **train dataset** which initially consisted of 810 claims. I focused on claims that explicitly cited evidence, ensuring that each selected claim had a supporting passage or specific evidence sentences (referred to as "gold evidence"). This pre-filtering step was crucial to align the claims with the requirements for simulation generation and analysis.

To further streamline the process, I manually labeled the dataset into six distinct categories based on the contextual and topical focus of the claims:

1. **Biology**: Claims related to biological processes, organisms, or phenomena.
2. **Cellular**: Assertions focused on cellular mechanisms or components.
3. **Directional**: Claims describing trends, increases, or decreases in scientific observations.
4. **Population**: Claims addressing demographic or ecological population trends.
5. **Medicine**: Assertions related to medical research or findings.
6. **Epidemiology**: Claims centered on disease spread, risk factors, or public health interventions.

The final annotated dataset consisted of 564 claims distributed across the six categories. This distribution reflects the dominance of biological and cellular topics within the dataset, consistent with the broader scientific research landscape represented in SciFact. The focus on these categories was motivated by their relevance to real-world applications of scientific verification, particularly in domains like epidemiology and medicine, where accurate fact-checking can have significant implications.

Category	Count
Biology	267
Cellular	126
Directional	92
Population	53
Medicine	22
Epidemiology	4

Table 1: Count of Annotated Categories

3 Methodology

3.1 CoT Prompting for Code Generation

The process begins with the integration of COT prompting, which enables step-by-step reasoning for simulation generation. Each claim, along with its corresponding reference text and gold evidence, is provided to the LLM. The system utilizes a carefully crafted base template, incorporating the structure and functions defined in **simulation_utils.py** and the example simulation provided in **bacteria.py**. These files serve as foundational components: **simulation_utils.py** contains reusable utility classes and functions for constructing simulations, while **bacteria.py** demonstrates how to apply these utilities in a concrete scientific context.

The simulator generation is facilitated by prompting the LLM with a detailed system and user prompts. The system prompt provides the LLM with its role and task, ensuring it understands the need to create simulations capable of testing the veracity of scientific claims. The user prompt includes the claim, the reference text, and the gold evidence, guiding the LLM to tailor the simulation to the context of the claim. This ensures that the generated simulation is both relevant and precise.

By the end of the code-generation, about 19.5% of the simulation code was generated erroneously, which let to RuntimeErrors during output analysis. We discuss the Error analysis in detail in the Error Analysis Section (4.1).

3.2 Simulator Output Analysis

Once the LLM generates the simulation code, it is extracted, saved, and executed. The execution process is designed to capture both the output of the simulation and any potential errors that occur during runtime. This ensures that generated simulations are thoroughly tested for validity and functionality. If the simulation generates errors, the

````
You are a scientific claim inspector, who can catch fake claims accurately and verify correct claims efficiently.
  Your main process is building simulation for building environment for testing out claims. You compare the results
  of a simulation with a given claim and provide verification results as Supported (for True claims) and Refuted (for False claims).

  You will be given the following information below:
  1. A Claim
  2. A passage of supporting text from a scientific article that can be used to help verify the claim.
    Sometimes you will be provided with specific gold_sentences where the evidence of claim's veracity can be found.
  3. An example simulation that can be used as a template to help guide your construction of a simulation

  Your output will be a Python simulation whose output will clearly determine whether the claim is likely supported or refuted.

  Also, You must output the simulation between clodeblocks (```), or the simulation will not be correctly parsed.

  For example:
  ```

  # python comment stating the input claim
  # code fo the simulation
  # print the output of the simulation
  print(result)
  ```
````

Figure 1: System Prompt for Simulation Generation

````
Claim: {claim}
Reference Text: {reference_text}

Gold Sentence that can provide the evidence of claim verification:
{''.join(gold_evidence)}

  ```
  # simulation_utils.py
  {simulation_template}

  # bacteria simulation example
  {example_bacteria}
  ```

Rules for the output:
  1. Generate simulation according to the given Claim, take Reference Text to figure out the objects of the simulation and their properties.
  2. The generated code should not generate any errors
  3. If you are extending upon the given base template, or using code from the base template, include it in your code by using either of the two ways mentioned below:
    a. import statements (importing the classes from base template simulation_utils.py)
    OR
    b. writing the whole base template code before your simulation code

  4. Make sure the output of the simulation is appropriate to the given instructions.
  5. It clearly determines/shows if the claim is supported or refuted.

  Please generate your simulation now
````

Figure 2: User Prompt for Simulation Generation

output is analyzed to identify issues, and adjustments are made to refine the process iteratively.

The analysis of the simulation output is performed in a second stage of prompting, where the LLM is tasked with verifying the claim based on the simulation results. The outputs are classified into three categories: ”Supported,” ”Refuted,” or ”None,” with the last category reserved for cases where the simulation output is ambiguous or execution errors prevent conclusive verification. This two-step prompting process ensures a comprehensive evaluation of each claim, leveraging both the LLM’s reasoning capabilities and its ability to interpret the results of computational simulations.

The generated simulations are designed to align closely with the context of the claims, thanks to the inclusion of detailed evidence and reference texts in the input prompts. The outputs of the simulations typically include clear indicators of whether the claim is valid, derived from the simulation’s results. This ensures that the methodology not only generates accurate simulations but also produces outputs that can be directly used to verify claims systematically.

4 Evaluation

The evaluation of the generated simulations and their alignment with scientific claims was carried out through a multi-step process. The primary metrics for evaluation were the proportion of simulations that successfully ran, the number of correct predictions where the simulation’s outcome matched the LLM’s verdict, and the proportion of errors or inconclusive outcomes. The methodology focused on assessing the model’s capability to provide runnable code and accurate predictions across six distinct scientific categories.

For each claim, the simulation was executed to determine whether it was free of errors and produced a meaningful output. A simulation was classified as ”runnable” if it executed without errors and produced an output that could be analyzed by the LLM. Claims were then classified as ”Supported,” ”Refuted,” or ”Inconclusive,” based on whether the simulation’s output aligned with the LLM’s verdict. If the simulation output matched the expected label (either ”Supported” or ”Refuted”), it was considered a correct match. However, if the generated code failed to execute or produced ambiguous results, the outcome was deemed ”Inconclusive.” Here are the overall results highlight varying performance levels across the six categories:

For the claims that were incorrectly classified, a detailed analysis was conducted to understand the distribution of actual labels that were misjudged. Across all categories, the errors revealed a notable trend: many claims labeled as ”Refuted” or ”Supported” were misclassified.

This analysis highlights that errors were more frequent in distinguishing between ”Supported” and ”Refuted” claims, particularly when the simulations lacked sufficient specificity or when the context of the claim was complex. In cases where the simulation output was inconclusive due to run-

Category	Correct Match	Incorrect Match	Errors	Total
biology	161	56	50	267
cellular	68	27	31	126
directional	51	19	22	92
epidemiology	3	1	0	4
medicine	13	6	3	22
population	34	15	4	53

Table 2: Results by Category

Actual labels in the dataset which were misclassified		
Category	**Refuted**	**Supported**
biology	19	37
cellular	5	22
directional	6	13
epidemiology	1	0
medicine	1	5
population	5	10

Table 3: # of Misclassified Labels

time errors or ambiguous results, the LLM was unable to provide a definitive verdict, which accounted for the "Inconclusive" outcomes.

4.1 Error Analysis

This error analysis evaluates the discrepancies between the simulated verification results and the actual labels of scientific claims in the preprocessed SciFact dataset. By examining the underlying causes of these mismatches, the analysis highlights potential issues in the pipeline, including errors in simulation generation, misinterpretation of evidence, and limitations in model reasoning.

In most cases, the errors involve claims that are actually Supported but are misclassified, highlighting a possible bias in the system towards refuting claims or a failure to extract sufficient support from the evidence. Here are some samples of misclassified claims referenced at Table 4.

Many incorrect classifications arise due to weak connections between claims and evidence. For example:

- In directional, the claim "Bariatric surgery increases rates of colorectal cancer" was supported by evidence stating "No association was detected between bariatric surgery and cancer." This highlights a failure to process negations and logical inferences.
- In medicine, evidence often included detailed experimental outcomes, such as reductions in headache indices. The verification system likely failed to distill such quantitative results into qualitative conclusions.

Also, ambiguities in simulation results, such as unclear numeric thresholds or causal relationships, often lead to incorrect verification during the analysis stage. For example, claims supported by evidence indicating subtle statistical outcomes were sometimes misclassified due to the lack of structured output from the simulation. Errors in the generated code, such as unhandled exceptions or flawed logic, also propagate into the verification process, leading to unverifiable or incorrect results.

Another notable challenge lies in the model's inherent limitations. The GPT model, while proficient at general reasoning, struggles with domain-specific knowledge. In the cellular category, for instance, claims about molecular pathways or genetic factors, such as "Egr2 and Egr3 regulate the homeostasis of B and T cells," require a level of logical understanding that may be absent in the current the model's capabilities. The model also exhibits a bias toward misclassifying ambiguous claims as Supported, likely due to its overreliance on surface-level text matching rather than robust logical reasoning.

5 Conclusion

The methodology demonstrated the potential of Chain-of-Thought (CoT) prompting and simulation-based analysis to address the challenging task of assessing claim veracity in scientific contexts. Using the SciFact dataset, 564 claims were filtered and categorized into six distinct scientific domains: biology, cellular, directional, medicine, epidemiology, and population. The project achieved a notable milestone, successfully generating runnable simulations for 58% of the claims, with outputs analyzed to classify claims as "Supported," "Refuted," or "Inconclusive."

The evaluation results underscore key findings. Across all categories, the accuracy of claim ver-

Status: Refuted
Evidence: In vitro, increasing microtubule acetylation using deacetylase inhibitors or the tubulin acetylase α TAT1 prevents association of mutant LRRK2 with microtubules, and the deacetylase inhibitor trichostatin A (TSA) restores axonal transport. In vivo knockdown of the deacetylases HDAC6 and Sirt2, or administration of TSA rescues both axonal transport and locomotor behavior.
Claim: Increased microtubule acetylation worsens interference of axonal transport caused by LRRK2 Roc-COR domain mutations.
Status: Supported
Evidence: Here, integrating human tumour genomics and syngeneic mammary tumour models, we demonstrate that mTOR signalling in cancer cells dictates a mammary tumour's ability to stimulate MDSC accumulation through regulating G-CSF.
Claim: G-CSF increases the expansion and infiltration of MDSCs into tumors.

Table 4: Claims that were misclassified

ification varied significantly. Categories such as biology and cellular, with 267 and 126 claims respectively, demonstrated relatively high success rates in generating correct predictions. However, challenges emerged in domains like medicine and directional, where intricate scientific contexts and evidence posed significant hurdles for accurate simulation generation and reasoning. The error analysis revealed that 19.5% of generated simulations encountered runtime errors, while 27% of the claims were misclassified due to issues such as evidence misinterpretation or oversimplification of simulations.

Despite these limitations, the project underscores several strengths. The integration of LLMs for scientific reasoning demonstrated significant potential, with correct matches in claims in the areas of medicine and cellular processes, where evidence-driven simulation provided a clear and accurate result.

However, the project also reveals critical areas for improvement. The reliance on generic LLMs without domain-specific fine-tuning limits their ability to handle complex scientific claims. Future work could involve augmenting the pipeline with domain-adapted models, refined simulation templates, and enhanced evidence preprocessing techniques. By addressing these limitations, the approach can be further optimized to achieve greater accuracy and reliability in scientific claim verification. This study highlights the feasibility of integrating LLMs into simulation-based analysis, paving the way for advancements in automated scientific reasoning and fact-checking.

References

2020“protectWadden et al.2020 David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online, November. Association for Computational Linguistics.

2024“protectWang et al.2024 Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. 2024. Can language models serve as text-based world simulators? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–17, Bangkok, Thailand, August. Association for Computational Linguistics.