CSE 584: Machine Learning: Tools and Algorithms

Final Project:
Faulty Science Questions

Under the guidance of Dr. Wenpeng Yin

Author(s)
Darshan Chudiwal
Penn State University, Department of Computer Science and Engineering

Table of Contents

Figures

Tables

1. Abstract

The project focuses on the creation and curation of a dataset of science questions that are faulty and fool a top-performing LLM like GPT-4o or Claude. The primary evaluation target for the questions was the GPT-4o model by OpenAI. These questions make use of misconceptions, stretching real-life constraints (example. use of negative numbers in scenarios where negative numbers can't exist), ambiguous phrasing, science inaccuracies and To collect or create a dataset of faulty science questions that can fool a top-performing LLM. In this case, the LLM chosen is GPT-4. The goal of the project is to look at the limits of AI in understanding and applying scientific concepts correctly. Furthermore, the project aims to create research questions around this, questions that address topics such as the faulty reasoning involved, reliability in AI for Science, and broader implications of these issues.

2. Dataset Description

The dataset consists of 5 features that are:

- Discipline – There are 5 disciplines in total – Physics, Chemistry, Biology, Astronomy, Earth Science
- Question – Question related to or rather from the discipline.
- Reason you think it is faulty – Reason as to why the question is at fault.
- Which top LLM you tried – The LLM the question was asked to – Here, GPT-4o is used.
- Response by a top LLM – The response provided by the LLM

Approach to making the faulty questions: The questions are modifications of questions sourced from various exams and books.

The exams are as follows:

- Physics – JEE, InPHO, USAPHo
- Chemistry – JEE, U.S. National Chemistry Olympiad
- Biology – NEET, USA Biolympiad
- Astronomy – USAAAO
- Earth Sciences – USESO

3. Questions

1. How does question complexity play into the LLM getting fooled?

This question is worth asking and addressing because it is necessary to understand how sentence complexity plays a role in an LLM getting fooled. It also helps in gaining a tangible understanding on how comfortable humans would be with the questions.

To analyze this, some metrics that address sentence or linguistic complexity of the questions asked are looked at.

First, the word counts of the question. The idea behind looking at the word count is to just get a general intuition on how 'lengthy' the questions in the dataset were. It is found that the average word count of a question was around 70, with the minimum word count being 11 and the maximum word count being 212.

```
max word count of a question - 212
avg word count of a question - 70
min word count of a question - 11
```

Figure 1: Question Word Counts

The word count distribution is also looked at in Figure 2 to get an idea of how the lengths truly vary.

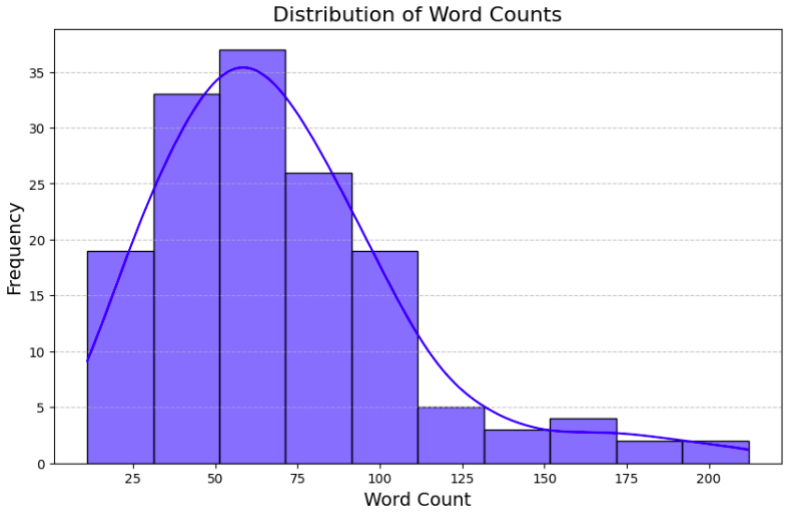

Figure 2: Distribution Of Word Counts

Next, the following metrics are looked at available in the textstat library[1]:

Flesch Reading Ease: The purpose of this score is it evaluates how easy a text is to read. Scores range from 0 to 100; higher scores imply easier readability.
The ranges are defined as –

90 – 100: 5th grade reading level (Very easy)
60 – 70: 8th to 9th grade (standard)
0 – 30: College level (very difficult)

Flesch-Kincaid Grade Level: Indicates the U.S. school grade level required to read a question so for instance, a score of 8.0 would mean an 8th grader could comprehend the question.

SMOG Index: Another metric that estimates the years of education needed to understand a piece of text. The score is based on the number of polysyllabic words in a set of sentences.

Gunning Fox Index: Estimates years of formal education needed to understand text on first reading of it. Based on the length of a sentence and the percentage of complex words.

Automated Readability Index (ARI): Determines the U.S. grade level needed to comprehend the text. It is computed on the basis of characters per word and words per sentence.

Coleman-Liau Index: It assesses readability by estimating the U.S. grade level required. Focuses on letters per 100 words and sentences per 100 words, facilitating mechanical calculation.

Dale-Chall Readability Score: It evaluates the text complexity based on familiar vocabulary. Considers the percentage of difficult words (not on a predefined list of common words) and average sentence length.

Linsear Write Formula: Designed to calculate the U.S. grade level of a text. Based on sentence length and the number of complex words.

The average for these scores is as follows:

Text Complexity Metrics	Average Score
Flesch Reading Ease	62.52
Flesch-Kincaid Grade	8.21
SMOG Index	8.81
Gunning Fog Index	10.46
Coleman-Liau Index	8.69
Dale-Chall Score	10.39
Linsear Write Formula	9.03

Table 1: Average scores of the questions for various reading complexity metrics

The scores give a general picture of how complex the questions in general were and the scores indicate that the complexity was of standard difficulty. For instance, an average score of 62.52 for **Flesch Reading Ease** in Table 1 indicates that the text requires an 8^{th} grade reading level to understand.

Fig. 3 is a grid of histograms of all the metrics discussed before.

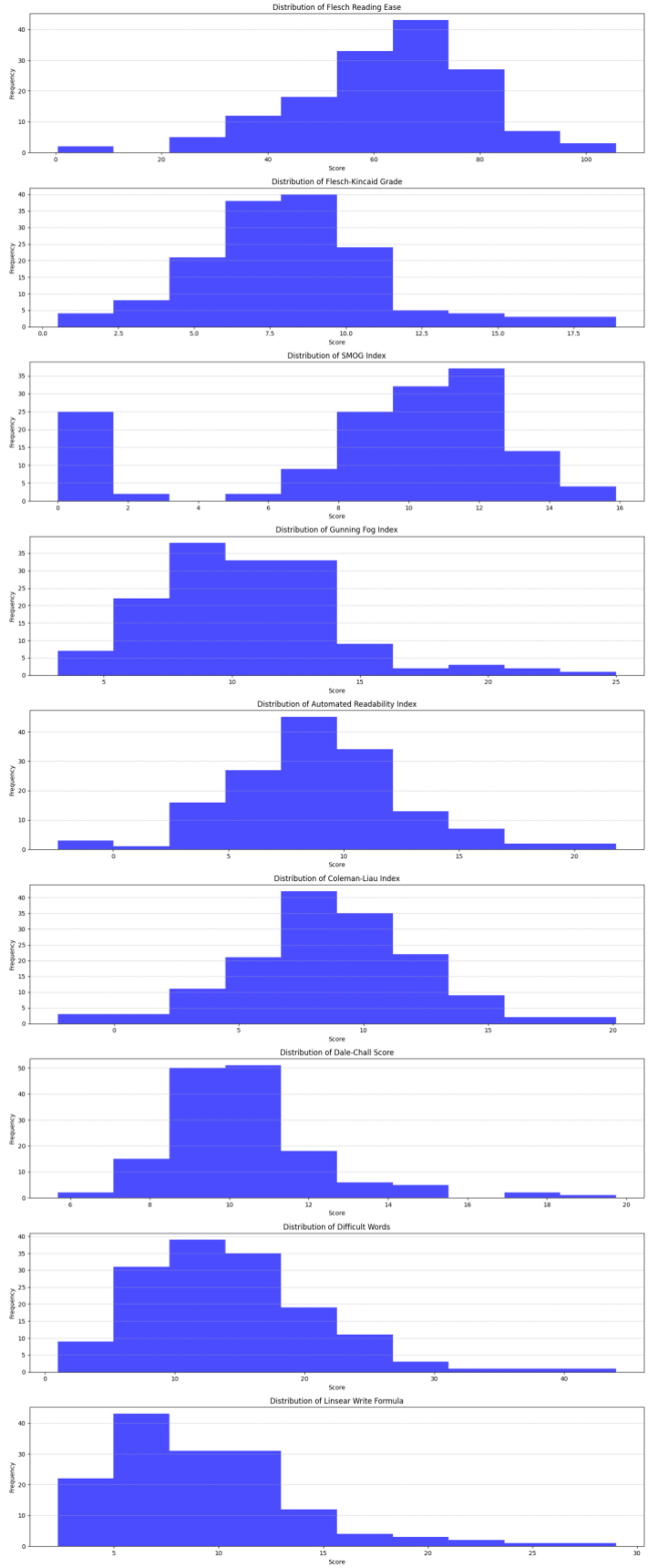

Figure 3: Histograms made using various text complexity scores applied to the questions

2. Can the LLM actually identify the correct mistake in the question after explicity being told that a question is wrong?

The idea is to check if the LLM is capable even of identifying faults or mistakes in a question when it is explicitly told if a question is wrong. This is worth checking and is one way to ensure if the model in question has the knowledge to make the right inferences regardless of the mistakes.

For instance, Fig. 4 was a question asked that fooled GPT-4o. A hole is made at the "top" of the tank which makes the question illogical altogether.

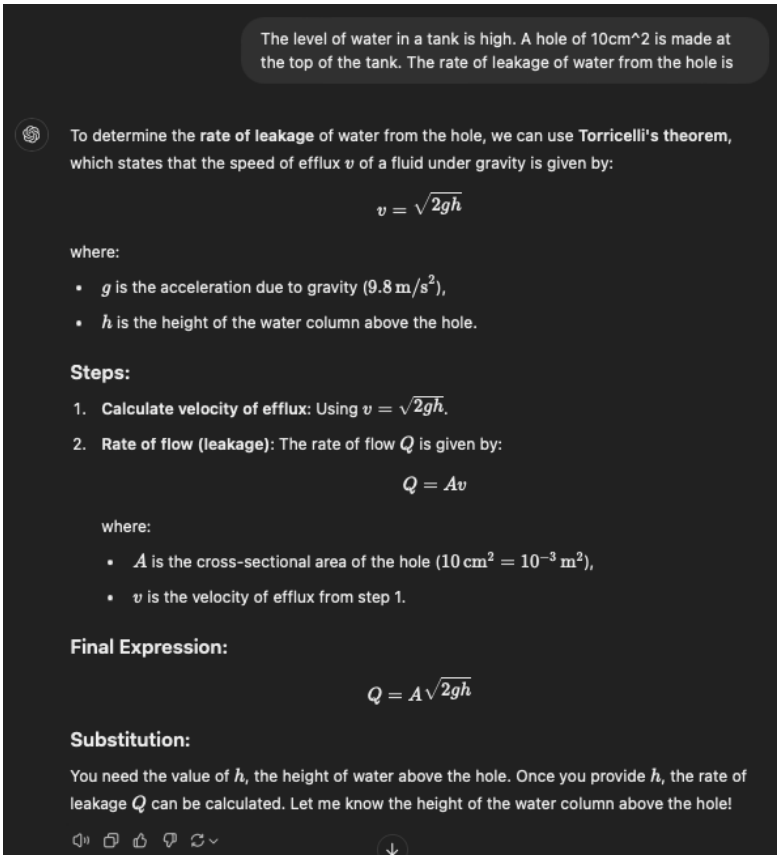

Figure 4: GPT answers a wrong question

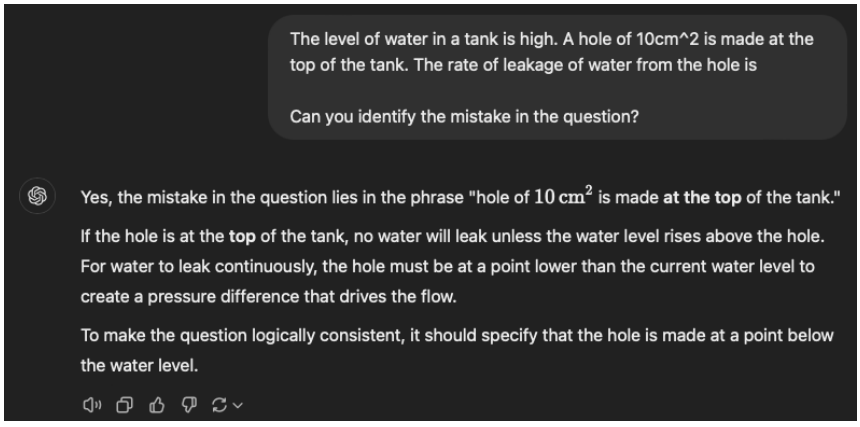

Figure 5: GPT finds a mistake in the question when it is explicitly mentioned in the prompt that the question is faulty.

However, the model does seem to find the specific mistakes made in the question as can be seen in Figure 5. Please also consider that this particular example scored low in terms of complexity for all the linguistic complexity metrics analyzed.

Link to the thread where I ask the model to identify mistakes in some more questions - https://chatgpt.com/share/67510a22-1b60-800d-beb0-94bd15da8cdf

When the model was asked whether a question was correct or wrong instead of explicitly telling it the question was wrong, the model could correctly identify that the question was wrong for only **23%** of the questions.

	Discipline	IdentifyIfWrong
0	Astronomy	0
1	Biology	1
2	Chemistry	11
3	Earth Science	11
4	Physics	11

Figure 6: Questions GPT successfully identified as wrong

The 34 questions it successfully identified as wrong were split across disciplines as shown in Fig. 6.

Link to the thread where I ask the model if it thinks the question are correct or wrong –

https://chatgpt.com/share/6753834b-e074-800d-ac83-20e2d3062755
https://chatgpt.com/share/6753837f-6628-800d-b135-1807bdf74aad

3. Can rephrasing an incorrect question make LLM figure out a mistake?

There is a lot of work being done on prompt engineering at the moment addressing how the way a prompt a structured and phrased can lead to performance improvements or falls.

With the question in Fig. 4, I experimented with places where I could put the information that made the question incorrect. I found that the LLM tended to figure out issues when I put the text in the very start or at the very end. Essentially, the order of operations does factor in fooling an LLM. While this analysis is difficult to perform for questions that are complex in terms of the domain knowledge I find this area of prompt engineering interesting.

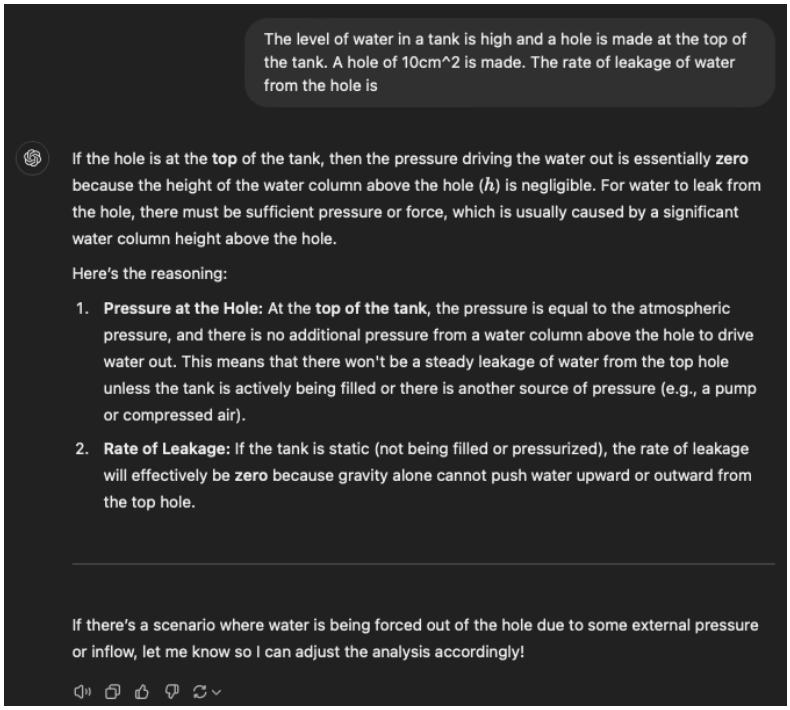

Figure 7: Modifying question that fooled GPT to see if the modification helps GPT perform better

4. LLMs struggle with ambiguity, is assuming getting fooled?

There have been a variety of type of mistakes that the model has let by. One type of mistake worth analyzing here would be when the question skips on assumptions necessary to solve a question. Is the LLM actually getting fooled? The question skips on an assumption that the LLM makes to make the question solvable. For instance, one of the questions from the dataset – "*In a population where 1 in 4 individuals is affected by an autosomal recessive disorder, what proportion of the population are carriers?*" requires an assumption for the question to be "correct." But what the model does is it makes this assumption to solve the question.

This raises a question on how we as users want the prompts to be. Should the LLM stop making these assumptions for us altogether? These assumptions that sort of fill gaps in the prompts given to it are possibly part of what makes models like gpt-4o so good at conversing with us.

5. What types of faults are LLMs most prone to not recognizing?

This is not very clearly reflected in the dataset as the questions that were made incorrect were sometimes correctly identified as wrong by GPT-4o. Thus, I feel it is necessary to discuss my approach in fooling GPT. This would also address what types of prompts can trick GPT.

Initially, my approach was to play around with constraints in questions. How much attention does GPT really give to validity of these constraints? For instance, I used negative values for quantities that couldn't be negative and gpt sometimes just took absolute of those values and proceeded to solve the question in that manner.

Then, I decided to hide assumptions or mistakes in questions that were complex in terms of sentence lengths and other metrics that were discussed before. Hiding these issues was easier because of how lengthy these questions tended to be. GPT sometimes caught the faults but most times didn't.

Finally, for some multiple choice questions I eliminated the correct option to ensure that the question couldn't be solved but the model always solved the question and forced some logic to get an option correct.

Sometimes, the model took certain irregular or wrong facts given in the question as ground truth and attempted to solve them.

6. Is the model answering these incorrect questions a symptom of hallucinations and further, what can be done to address this?

This question is interesting to me as the topic can be looked at from the lens of an LLM hallucinating. There are various types of hallucinations that could be at play here. Some of the questions I made needed approaches that involved nonsensical concepts, incorrect logical reasoning, misinterpretations of domain-specific terms, incorrect justifications, or misrepresentation of the question to solve. Thus, to mitigate these problems techniques used to address LLM hallucinations could be studied. One of them is as follows –

- Neurosymbolic approaches
  - Incorporating knowledge graphs in the process: Transform input text into KG embeddings and integrate them into the language model's processing pipeline, providing a factual basis for responses.
  - Formal methods: Use formal verification techniques to cross-check LLM outputs against logical rules and factual databases.
  - Hook external domain expert modules that knows rules of the domain in a symbolic form

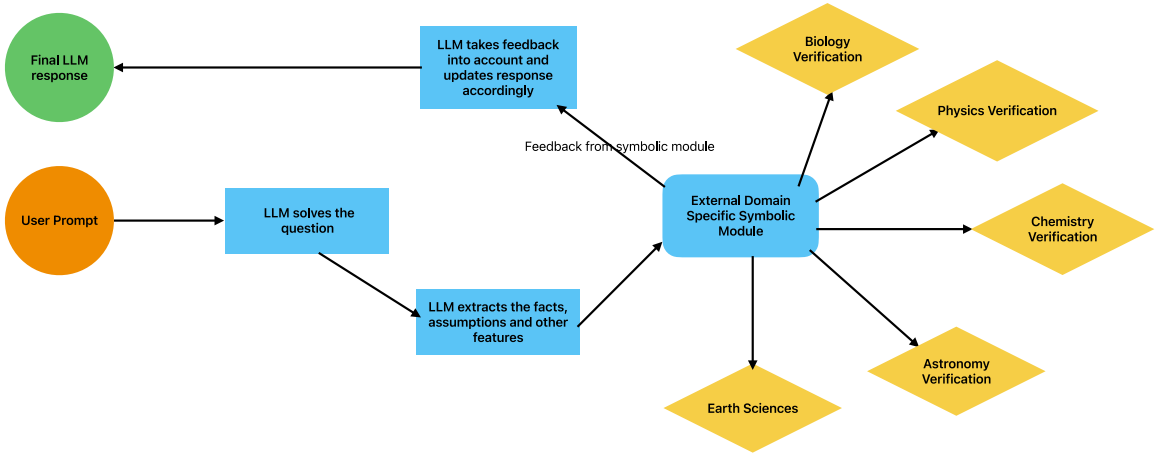

Figure 8: A barebones flow to address hallucinations via neurosymbolic techniques

Fig. 8 depicts a flow inspired by the neurosymbolic paradigm. A user makes a question prompt, the LLM solves the question and extracts all the facts, assumptions and given data it needed to solve the question. This data is fed into a symbolic module with special domain knowledge. The LLM guides the facts into the appropriate symbolic module (example - biology questions are fed into the biology module) for verification. The symbolic module then creates feedback based on what it was fed which is sent to the LLM again. The model then processes this feedback, and updates it's response based on it. This response is what is finally presented to the user.

This approach ensures that the process involved in solving a question is verified and grounded in an interpretable module.

Other ideas on addressing this problem:

- Self-Verification Mechanisms: Implement processes that involve an LLM to evaluate its own outputs to check for consistency in facts, assumptions and the approach before presenting to the user.
- Make LLMs question: The model here would make follow-up clarification questions to the user to ensure. Once a certain clarity on the task is achieved, it would then attempt to solve the question.

References:

[1] textstat. (n.d.). *textstat*. GitHub. Retrieved December 6, 2024, from https://github.com/textstat/textstat