Student: Lucas George Sipos
Student ID: 24292215

Table of Contents

Student: Lucas George Sipos
Student ID: 24292215

Practical 4: Report

Link to playground

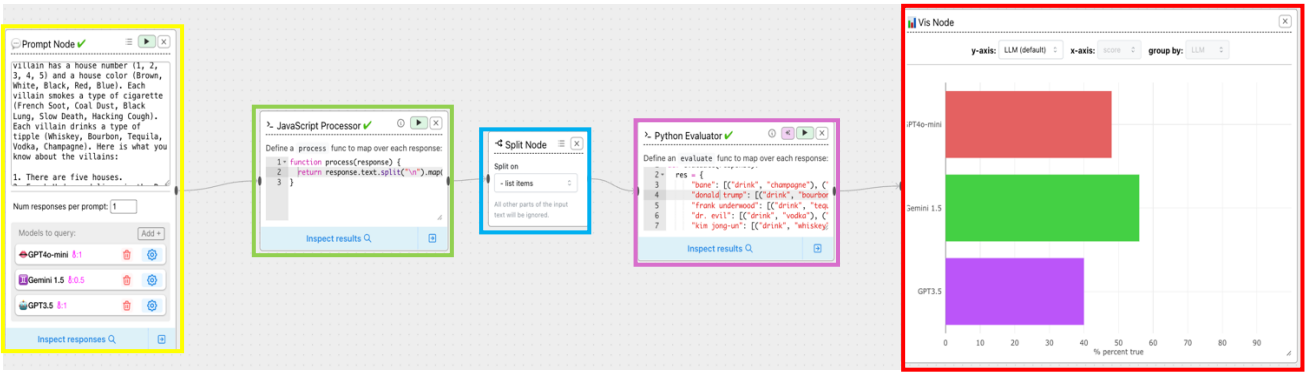
- https://chainforge.ai/play/?f=1c7j9yi7hv984

API Keys

- HuggingFace: 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
- Google AI Studio Key: AIzaSyDTYzVWWU_NB1K4WXSp4HpfhkPjFYNWEHw
- OpenAI key: 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

Working flow

Below, I will insert an image of the playground, which will have rectangular colour-coded areas for better visualisation of the workflow steps.



Yellow Area Overview

This area uses a Prompt Node to give the LLMs all the information needed about the riddle and how to format the answer.

Green Area Overview

This area represents the JavaScript Processor, which helps retrieve every input as a formatted one and sends chunks of data to Blue Area.

Student: Lucas George Sipos
Student ID: 24292215

Blue Area Overview

This Split Nodes separates into single sentences the text sent by Green Area.

Purple Area Overview

The Python Evaluator Node evaluates every sentence by checking into a dictionary of lists of tuples.

Red Area Overview

We have a Visualisation Node so that we can clearly see the difference in result findings.

Datasets Used

Everything regarding the Hawking Riddle (the explanation and the starting point).

Prompt Engineering

To ensure optimal LLM performance, I had some key factors in building the prompt:

- Gave him an overview of the problem

> Here is the information needed to write the response: you have 5 villains (Donald Trump, Dr. Evil, Frank Underwood, Bane, Kim Jong-un). Each villain has a pet (Octopus, Robot Chicken, Shark, Cyborg Tortoise, Armadillo). Each villain has a house number (1, 2, 3, 4, 5) and a house color (Brown, White, Black, Red, Blue). Each villain smokes a type of cigarette (French Soot, Coal Dust, Black Lung, Slow Death, Hacking Cough). Each villain drinks a type of tipple (Whiskey, Bourbon, Tequila, Vodka, Champagne).

- Gave him the starting point

> Here is what you know about the villains:
>
> 1. There are five houses.
> 2. Frank Underwood lives in the Red house.
> …
> 14. Kim Jong-un smokes Hacking Cough.
> 15. Bane lives next to the Blue house.

- Then told him to give me the answer to the riddle in the given format

> Now only give me a list of 25 factoids in the following format:
> 1. <villain> drinks <tipple>.
> 2. <villain> smokes <cigarette>.
> 3. <villain> lives in <house number>.
> 4. <villain> house color is <house color>.
> 5. <villain> has pet <pet>.

Everything that's between diamond braces is used for generalisation.

JavaScript Processor and Split Node

For this part, I've tried forcing the LLM to give me an ordered list, but it wasn't like that for every LLM, so I had to delete all the lines that don't start with a number or are just new lines character and then returning the entire text formatted as a ordered number list and the Split Node split the text into single sentences.

Python Evaluator

The evaluator uses a dictionary of lists of tuples having the following format:

- keys: villain names
- value: list of tuples, each tuple having 2 values representing:
  - 1st value: the action to identify the sentence meaning
  - 2nd value: the subject which verifies if the result matches the solution

And for every sentence I looped through the keys and if I find a villain, then I find an action to identify the sentence, and then I see if the subject in the response is the same as the subject of the solution. If it is, I return True, and for any other case I return False.

Results & Analysis

LLM	Accuracy
Gemini 1.5	56.00%
GPT4o-mini	48.00%
GPT3.5	40.00%

The table shows the accuracy of different LLMs in solving the classic Einstein-style logic puzzle. The results highlight that Gemini 1.5 performed the best with 56% accuracy, followed by GPT-4o mini at 48%, and GPT-3.5 at 40%. Gemini 1.5 likely excels because it demonstrates better multi-step reasoning and constraint satisfaction abilities, both of which are essential for solving

logic puzzles. It appears to be more effective at keeping track of dependencies between clues and applying them systematically. GPT-3.5 likely struggles due to weaker constraint propagation and limited depth of inference, making it harder to connect indirect clues or maintain the puzzle's overall structure.

Findings

The performance gap highlights how different LLMs handle logical deduction and memory management. Models like Gemini 1.5 might be optimized for multi-step reasoning, while earlier models like GPT-3.5 focus more on surface-level understanding. As LLMs improve, so does their capacity to solve highly structured, multi-faceted problems like this puzzle.

Conclusion

The results show that newer LLMs like Gemini 1.5 and GPT4o-mini outperform older models in solving complex logic puzzles, likely due to enhanced reasoning capabilities and better constraint management. The gap between Gemini 1.5 and GPT-3.5 highlights how advancements in architecture and training methods significantly improve performance in tasks requiring multi-step deduction. However, even the best-performing model only reached 56% accuracy, suggesting that while LLMs are improving, they still face challenges in fully mastering logical inference; a critical step toward achieving more advanced forms of artificial intelligence.