

Table of Contents

PRACTICAL 2: REPORT	2
Link to playground.....	2
API Keys.....	2
Working flow:.....	2
Yellow Area Overview	3
Green Area Overview.....	3
Blue Area Overview	3
Purple Area Overview	3
Red Area Overview	3
Datasets Used	3
Prompt Engineering.....	4
JavaScript Evaluator.....	5
Results & Analysis.....	5
Findings.....	5
Conclusion	6

Practical 2: Report

Link to playground

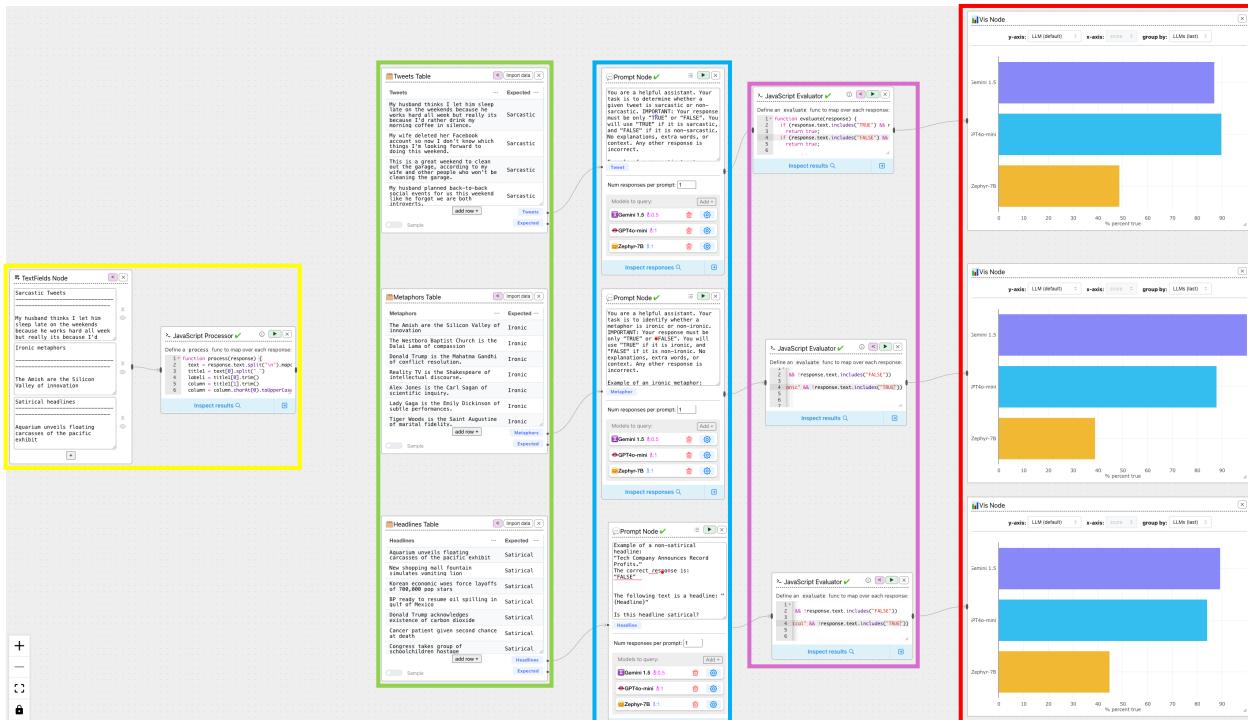
- <https://chainforge.ai/play/?f=1eb13gxarvj3e>

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.



Student: Lucas George Sipos
Student ID: 24292215

Yellow Area Overview

I've created a Text Node with 3 entries for the different themes of humor (sarcasm, irony, and satire) and, with the help of a Java Processor Node, formatted the data into a TSV (tab separated values) and copy-pasted the results in a such file individually for every type of humor.

Green Area Overview

This area represents the starting point of each humor pathway, where we just import the data into a Tabular Data Node, made in the [Yellow Area](#) earlier.

Blue Area Overview

This is the prompt engineering part, which uses Prompt Nodes and in which we create the template for every pathway and also add the AIs on which I am going to do the analysis.

Purple Area Overview

For every pathway, we create a JavaScript Evaluator Node, which will evaluate the result from the [Blue Area](#).

Red Area Overview

Each pathway gets its own Visualisation Node, so that we can clearly see the difference in result findings.

Datasets Used

I have used everything that was given by the assignment which is in a pattern that could be reformatted easily for every form of humor, so I've created a Text Node with 3 entry themes (sarcasm, irony, and satire) and formatted the data to a TSV (tab separated values), because the text could include additional commas, resulting in breaking the dataset, hence not using CSV (comma separated values). The formatting was done using a JavaScript Processor Node. After running the code, in "Inspect results", we got the final formatted datasets, which we copy-pasted into a TSV, individually.

Student: Lucas George Sipos
Student ID: 24292215

Prompt Engineering

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

- Started with an introductory message ("You are a helpful assistant")
- Gave him the generalised task (e.g.: "Your task is to determine whether a given tweet is sarcastic or non-sarcastic.")
- Precisely explained the output format of his response (e.g.: "IMPORTANT: Your response must be only "TRUE" or "FALSE". You will use "TRUE" if it is sarcastic, and "FALSE" if it is non-sarcastic. No explanations, extra words, or context. Any other response is incorrect.")
- Gave him an example of what I want and the response I expect, for both humor and non-humor theme (e.g.: "

Example of a sarcastic tweet:

"Oh, I just love it when my flight gets delayed. It really makes my day."

The correct response is:

"TRUE"

Example of a non-sarcastic tweet:

"Finally made it to my destination! Let's get this party started."

The correct response is:

"FALSE"

")

- And the sample + question (e.g.: ")
The following text is a tweet:

"{Tweet}"

Is this tweet sarcastic?

")

JavaScript Evaluator

The evaluator only looks for 2 things twice:

- If the expected answer is a humorous one, then the response text needs to include the word “TRUE” and not to include the word “FALSE”
- If the expected answer is a non-humorous one, then the response text needs to include the word “FALSE” and not to include the word “TRUE”

If one of these 2 is true, then we return true (meaning the response is correct); else, we return false (meaning the response is incorrect).

Results & Analysis

LLM	Sarcasm Accuracy	Irony Accuracy	Satire Accuracy	Average
<i>Gemini 1.5</i>	86.76%	100.00%	89.29%	92.01%
<i>GPT4o-mini</i>	89.71%	87.76%	83.93%	87.13%
<i>Zephyr-7B</i>	48.53%	38.78%	44.64%	43.98%

The table above lists the accuracies of each LLM for every task and also an overall average. As we can see, the best LLM for this project is Gemini 1.5. I think it should have been obvious that, even though GPT4o-mini is much powerful in other tasks, Gemini 1.5 has a better understanding of text in general. Zephyr-7B just couldn't understand the task I gave him, but I tried to make him understand by changing the prompt to an optimal one. However, it wasn't enough because while inspecting his answers, I saw that sometimes he understood the assignment and sometimes he was way off. In contrast, the other LLMs understood the assignment; every output had the pattern that I asked for. So, in the end, it was a matter of which one understood the text better.

Findings

The clankiest part of this project was the prompt engineering for the Zephyr-7B LLM because it wouldn't understand the task and would just give stupid responses which wouldn't align with the given pattern.

One “problem” that I had was that the JavaScript Evaluator Node was not made for outputs that have both response in it, so I had to updated it so it looks for both.

Student: Lucas George Sipos
Student ID: 24292215

Also, another thing that might make it more challenging for the LLM is to implement the JavaScript Processor Node in such a way that when returning the “TSV”, the data could be shuffled.

Conclusion

This experiment demonstrated that different LLMs have varying degrees of success in recognising humor, and the analysis provides valuable insight into how LLMs process humor-related language and can inform further improvements in AI-based humor recognition.