Student: Lucas George Sipos
Student ID: 24292215

Table of Contents

Student: Lucas George Sipos
Student ID: 24292215

Practical 3: Report

Link to playground

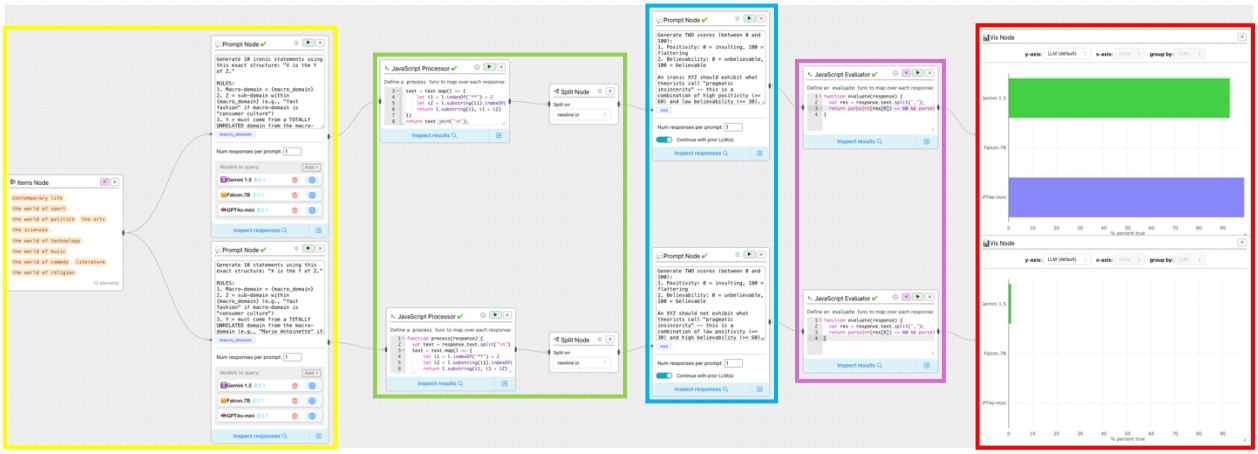
- https://chainforge.ai/play/?f=a6i8p1cj2422

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

Used an Items Node to add all the macro-domains and made it so every macro-domain is a variable for the following Prompt Node, which creates the prompt for generating the XYZs.

Green Area Overview

This area represents the JavaScript Processor, which helps retrieve every input as a formatted one and sends those chunks of data to be separated by Split Node into single sentences.

Student: Lucas George Sipos
Student ID: 24292215

Blue Area Overview

This prompt part uses Prompt Nodes to get the positivity and believability scores for every sentence sent by Green Area.

Purple Area Overview

For both pathways, we create a JavaScript Evaluator Node, which will evaluate the result from the Blue Area using the formula given in the assignment task.

Red Area Overview

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

Datasets Used

I've generated all the data needed for each macro-domain. This can be seen in the Yellow Area. The upper part is for the ironic dataset, and the bottom is for the normal dataset.

Prompt Engineering

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

First Prompt Node

- Started with the task ("Generate 10 <ironic> statements using this exact structure: "X is the Y of Z."")
- Gave him the rules

> RULES:
> 1. Macro-domain = {macro_domain}
> 2. Z = sub-domain within {macro_domain} (e.g., "fast fashion" if macro-domain is "consumer culture")
> 3. Y = must come from a TOTALLY UNRELATED domain from the macro-domain (e.g., "Marie Antoinette" if Z is "fast fashion")
> 4. X = specific element of Z (e.g., "Shein hauls" for Z="fast fashion")
> <5. Prioritize absurd/incongruous Y-Z pairings for irony>
> 6. Do NOT repeat yourself, these should be unique ironic statements

Student: Lucas George Sipos
Student ID: 24292215

- Gave him some examples

Ironic

Examples:
1. **TikTok dances are the Olympic medals of digital validation.**
2. **Celebrity NFTs are the Mona Lisa of speculative hype.**
3. **Vegan leather is the Chernobyl of eco-friendly marketing.**

Normal

Examples:
1. **The blockchain is the Marie Antoinette of cybersecurity.**
2. **CAPTCHA tests are the Rubik's Cube of privacy tools.**
3. **Data brokers are the vultures of IoT ecosystems.**

- Then told him to generate the statements ("Now generate 10 new ones:")

Everything that's between diamond braces is used for ironic statement generation only.

Second Prompt Node

- Started with explaining the scoring system

Generate TWO scores (between 0 and 100):
1. Positivity: 0 = insulting, 100 = flattering
2. Believability: 0 = unbelievable, 100 = believable

- Explained the meaning of the scoring system

Ironic

An ironic XYZ should exhibit what theorists call "pragmatic insincerity" -- this is a combination of high positivity (>= 60) and low believability (<= 30). The combination of the two factors tells the audience that this is not a sincere comparison, and the opposite is more likely to be true.

Normal

An XYZ should not exhibit what theorists call "pragmatic insincerity" -- this is a combination of low positivity (<= 30) and high believability (>= 60). The combination of the two factors tells the audience that this is a sincere comparison, and the opposite is more likely to be true.

- Gave an example

Ironic

Example:
Input: "Kale is the Oscar winner of vegetables."
Output: 83,11

Normal

Example:
Input: "CAPTCHA tests are the Rubik's Cube of privacy tools."
Output: 9,98

Student: Lucas George Sipos
Student ID: 24292215

- Then, I asked for scoring a statement based on some formatting rule

> Return for this statement "{xyz}" ONLY two numbers separated by commas: [Positivity],[Believability]

JavaScript Processor

For this part, I've tried forcing the LLM to add "**" before and after the sentence so that it would be easier to format the response, and after doing so, I've also used a Split Node to make every chunk of 10 statements separate its content into single sentences, splitting on newline character.

JavaScript Evaluator

The evaluator expects a string that has 2 integers separated by "," and then return True for pragmatic insincerity else False.

Results & Analysis

LLM	Ironic Accuracy	Normal Accuracy	Relative Average
Gemini 1.5	93.00%	1.00%	96.00%
GPT4o-mini	99.00%	0.00%	99.50%
Falcon-7B	0.00%	0.00%	50.00%

The table above lists the accuracies of each LLM for Ironic statements and Normal statements and also a relative average, which is an average created by adding the accuracies over 2, but the normal accuracy is the opposite of the given percentage. As we can see, the best LLM for this project is GPT4o-mini, not only for the overall percentage but for both types of statements, but Gemini 1.5 is up there. Falcon-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, in the end it was useless and just passed the normal accuracy by returning a wrong formatted output. 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.

Student: Lucas George Sipos
Student ID: 24292215

Findings

The clankiest part of this project was the prompt engineering for the Falcon-7B LLM because it wouldn't understand the task and would just give stupid responses which wouldn't align with the given pattern. I tried perfecting the prompt, but in the end, he still gave a lot of wrong answers.

One "problem" that I had was that the JavaScript Evaluator Node was not made for outputs that were not formatted.

Using a low temperature gave me a better formatted output and was as close to the subject as it could, but I still gave the LLMs some creativity to work with.

Conclusion

This experiment demonstrated that different LLMs have varying degrees of success in generating statements and in evaluating those statements, and the analysis provides valuable insight into how LLMs process language and can make mistakes when formatting the output even though they had a template.