Project 1D1: Reflection

What are the pain points in using LLMs?

- LLM can get stuck in a cycle of outputting the same problem even when you are prompting to fix an issue it produced
- LLM can be very confident in incorrect answers .Sometimes it is better to start over than it is to try and convince the LLM that it is wrong. This can be seen particularly if you ask it to make *n* things and it accidentally makes *n+1* things. It is very difficult to convince the LLM that it cannot count.
- LLMs don't always have updated information. When the most up-to-date information is necessary, you always have to provide your own context for that data.
- LLMs are only so good at improving things, some details you need to do manually. For example, they struggle with tone and word choice.
- LLMs do not always do the most thorough analysis of context that you give it and may end up regurgitating certain information that is already present.
- LLMs tend to be repetitive when asked for lengthy text based answers.
- LLMs can be too agreeable/polite, even at the expense of correctness.
- It takes a lot of time to edit and filter through LLM output to get a quality final result.

Any surprises? Eg different conclusions from LLMs?

- Different LLM can have combating outputs to the same prompt
- Different models are geared towards different tasks. Claude seemed to provide more concise and structured responses to the prompts while GPT models tend to be more 'creative' and add-on or deviate from explicitly stated instructions.
- The LLMs would sometimes anticipate the next step and ask if I would like to perform the task it thought was coming next.

What worked best?

- Asking the LLM how it defines a certain idea or problem, then give the LLM your definition and ask it to change its definition
- Telling LLM it is an "Expert" in a certain area
- Giving sequential instructions but telling the LLM to wait for your input before continuing
- Giving LLM instructions/specifications on how to perform a certain task
- Asking LLMs to expand on, refine, or summarize a previous LLM output
- Having the LLM create multiple agents to do a task, say write a paragraph about food delivery apps, then have each agent critique each other's paragraph and come up with a consensus at the end.
- Telling the LLM to delete agents after they perform their task is important here because otherwise they create these agents, and they just take up space until eventually it runs out of attention.

- Giving specific critiques for improving responses
- Giving the LLM examples of what you want output to look and sound like

What worked worst?
- Trying to get LLM out of an incorrect output cycle
- Not giving enough background information before prompting LLM
- Rapid firing of prompts without context can lead to confusing LLM
- Not breaking large chunks of text up. The LLM would skip over large parts of the text because the context window isn't big enough.
- Zero shot prompting for important or information-rich tasks
- Trying to do multi-step processes with a single prompt

What pre-post processing was useful for structuring the import prompts, then summarizing the output?
- Sequentially forming input prompt then breaking it into chunks to feed to LLM
- Asking LLM to summarize output based on some specified constraints
- Providing an example output so we don't need to spend time wrangling the LLM to output something we want.
- Telling the LLM to not respond until I give it a keyword so you can input multiple chunks of information without having to wait for it to output text

Did you find any best/worst prompting strategies?
- Best prompting strategy
  - Figure out what LLM initially thought of a term or idea, then refine the definition of what a specific term or idea actually is in the context you are looking at it
  - Give example inputs and outputs (if available)
  - Ask for a specific step, then refine prompt until that step is correct before moving on to the next
  - Asking LLM to keep track of inputs before forming an answer
- Worst Prompting Strategy
  - Just copy and pasting instructions for an assignment
  - Trying to fit all the information into one prompt
  - Telling it to "do/make a thing" without more specification on how it should be done and what it should look like in the end.
  - Doing too much with one prompt, like trying to do all of the use case expansion and narrowing in one step would yield very bad results
  - Assuming the LLM has an unlimited context window
  - Not giving a specific order for things that would need to be done in order
  - Not taking advantage of the "memory" LLMs have