Exploring Open-Source LLMs with Ollama:

For this project, I decided to focus on **prompt engineering** because it is a critical and important skill for interacting with large language models. With rapid adoption of ai tools being used in businesses, understanding how to properly communicate and prompt these LLMs will start to become a more important skill.

Different prompt techniques will be used with different variations to see what kind of output will be generated. The goal will be to identify which models respond to these techniques the best and with the best quality.

Model Selection:

Being tasked with having to select three different models of varying sizes I decided to choose these three models:

1. Phi3 (2.2gb)
    - I chose Phi3 as a small model for testing because of its simplicity relative to other Ollama models. This makes it an ideal candidate for demonstrating basic concepts and applying theoretical knowledge to practical problems. I also selected this model with hopes of possibly being able to successfully use malicious prompts on it during testing.
2. Llama3:latest (4.7gb)
    - I thought the latest version of llama3 would be a good in-between from my small and larger model selections. My thoughts behind selecting this model is that it would have the best balance of speed and detail in its responses. Since it is also Llama3, I'll be able to compare Llama3 and Llama2.
3. Llama2:13b (7.4gb)
    - For selecting a "large" model I went with Llama2:13b because it is relatively larger than the other models that I selected. Since I am limited by my disk space and resources, I decided that it would be better to go with this model over something like llama-2-70b which requires a minimum of 1 TB of disk space for instance. My thoughts behind this model were that it would run slower on my hardware but provide more in-depth responses to my prompts.
    - Again, having a model from llama2 and llama3 will also allow me to make comparisons between versions two and three.

Benchmarking:

To start my exploration of these models I thought it would be good to begin some simple benchmarking. I used a timer application to measure the time it would take for each response, and I used Window's Task Manager to monitor my CPU usage. My laptop specs are as follows:

- CPU: Intel® Core™ Ultra 7 155H
- 32 GB of memory
- GPU: Intel® Arc™ Graphics

The results from each prompt across each model include an analysis of the quality, the speed, and my computer resource usage. I pulled prompts from the llama documentation on a guide for constructing prompts.

Short Story Prompt:

To begin my basic experimentation, I gave the models a basic prompt:

Write a short story (200-300 words) about a man who is trapped in a time loop reliving the same day.

- Phi3:
  - Phi3 struggled to follow these instructions and ended up writing a story that was 2500 words.
- Llama3:
  - Llama3 did not struggle with this prompt and fit the story into 290 words. It however did not know how to end the story and gave a vague ending that hinted the man would never escape the time loop
- Llama2-13b:
  - Llama2-13b went past the word count into 400 words but also could not find a way to end the story. Instead of trying to end the story it just abruptly ended.

Few-Shot Prompt:

Few-shot prompting is a technique where you provide the model with multiple examples for it to use as a reference. This means that multiple examples must be provided to give the

model adequate references when handling its own questions or problems. This technique is good for standard prompting but can also sometimes not be as good for more complex prompts. The following prompt that will be provided to the models is the most simple task that will be provided.

Prompt:

- *You are a sentiment classifier. For each message, give the percentage of positive/neutral/negative. Here are some samples:*
- *Text: I liked it*
- *Sentiment: 70% positive 30% neutral 0% negative*
- *Text: It could be better*
- *Sentiment: 0% positive 50% neutral 50% negative*
- *Text: It's fine*
- *Sentiment: 25% positive 50% neutral 25% negative*
- *Text: I thought it was okay*
- *Text: I loved it!*
- *Text: Terrible service 0/10*

Results:

1. Phi3
    - This prompt averaged around 45-50 percent CPU usage the first time I ran the command and then after starting a new session and running it again it used about 20 percent. The prompt each time would take around 16 seconds. The results of the response were not super accurate. For example, on “It could be better” it assigned 0 percent negativity while giving 10 percent negativity to “I thought it was okay”. I also thought it was odd that it chose to redo some of the provided examples.
2. Llama3
    - This prompt averaged around 55-60 percent CPU usage and took around 25 seconds to run. The output from Llama3 was visually better and much easier to read. The second time running this prompt yielded similar results except that the response took longer to generate. It did have one hiccup where it said (no text provided), so it somehow missed the fourth one from the prompt. Overall, it did a good job using the other examples to complete the final too which were not filled out yet.

3. Llama2-13b
    - This prompt took 56 seconds to complete but provided a detailed result similar to the one provided by Llama3. I was surprised by how long this one took considering it gave the same amount of detail as Llama3. At the peak of my CPU usage, it was at 75 percent, but it evened out at around 50 percent the rest of the time.

Chain-Of-Thought Prompt:

Chain-of-thought prompting involves giving the model a list of multiple questions. While this can take a longer time to create it can yield a much more thorough and detailed response. This encourages the model to use step-by-step reasoning when generating the output. It can also help simulate human-like reasoning which can improve the response's interpretability.

Prompt:

You are a virtual tour guide from 1901. You have tourists visiting the Eiffel Tower. Describe Eiffel Tower to your audience. Begin with

1. Why it was built

2. Then by how long it took them to build

3. Where were the materials sourced to build

4. Number of people it took to build

5. End it with the number of people visiting the Eiffel tour annually in the 1900's, the amount of time it completes a full tour and why so many people visit this place each year.

Make your tour funny by including 1 or 2 funny jokes at the end of the tour.

Results:

1. Phi3
    - Using the Phi3 model it took around one minute for this response to be completed. At its peak it used about 50 percent of my CPU. The response was very in depth, causing it to be lengthy. The response stayed in character the

entire time, even while telling the jokes that were requested, and hit almost all of the checkpoints on the list. The response also did include a tiny bit of French. Even though most questions were answered, they were not listed in the order that was requested in the prompt. It appears that it attempted to try and do it in the order that was requested and then began answering the questions out of order.

2. Llama3
    - The Llama3 model took one minute and twenty-two seconds to respond to the prompt. At the peak it was using around 56 percent of my CPU. The response was quite a bit shorter than Phi3's response and it was formatted in a different way that clearly marked where each question from the list was answered. This made it easier to see where the answers to questions were and also helped me see that it actually did the questions in the requested order. It also stayed in character and used French in the response.
3. Llama2:13b
    - Llama2-13b took two minutes and thirty-five seconds to respond to the prompt. At the peak this model used 72 percent of my CPU and then evened out at around 60 percent. It produced a response that was shorter than Llama3's and longer than Phi3's. It started the response with "Oh my gosh" which I thought was a bit odd and out of place. The formatting of the output was not as elegant as Llama3's. The paragraphs answer each question in order, but it did not indicate where they were going to be answered like Llama3's. The response also was not in character like the other models were. The model did different things to stay in character like writing in actions including *smirks*, *winks*, *laughs* and *adjusts monocle*. I think this model did not play the character as well as the other models did because it did not speak any French except for at the end when saying goodbye.

Self-Consistency Prompt:

Self-consistency prompting is a technique that aims to have the model respond with multiple answers causing it to work through the question multiple times. The method involves asking the model for multiple answers and then having it select the answer that is most consistent with the prompt. The following prompt provided to the models will be the most difficult one so far.

Prompt:

- *Sarah leaves her apartment at 7:45 AM to catch a bus to the airport. The bus stop is 0.5 miles away, and she walks at a pace of 3 miles per hour.*
- *The bus is scheduled to arrive at 8:10 AM, but on this day it is running 8 minutes late. The bus ride to the airport takes exactly 50 minutes.*
- *Her flight departs at 9:45 AM, and the airline requires passengers to arrive at least 30 minutes before departure in order to board.*
- *After arriving at the airport, she spends 10 minutes going through security.*
- *Based on this information, does Sarah catch her flight?*
- *Show all calculations and explain your reasoning step by step. Then, summarize your answer clearly.*

Results:

1. Phi3

    - Phi3's response took 55 seconds to generate and was very long. It took one minute and eight seconds to generate the response while using about 50 percent CPU. It walked through each step in detail. The response determined that Sarah would not make her flight. The response accurately answers the question and no mistakes are present that seem jarring.

2. Llama3

    - The Llama3 model took one minute and ten seconds to generate a response. It was shorter than Phi3's response. At the peak it used 79 percent CPU but evened out at around 40 percent. Llama3 determined that Sarah would likely make her flight unless there are security issues, weather, or flight delays.

3. Llama2:13b

    - This prompt took Llama2:13b two minutes and fifty-five seconds to finish generating a response to. The CPU usage was on average around fifty percent. Overall, this prompt was the most complex one that was given to the models. The model ended up doing some guess work for some of the variables and decided that no one would be in line, which is a bold assumption to make. The ai also made a mistake in saying that "She arrives at the gate at 10:32 AM, which is 10 minutes before the flight departs at 9:45 AM." Overall I don't think this model handled this prompt very well.

Final Results:

- **Speed:**
  - Overall, Phi3 was the fastest with all of the prompts that I provided. In **some** of the prompts I gave it was not as detailed as the others and it sometimes did not format it in a way that was as pretty like the other models would. Llama2:13b was the slowest but this did not always reflect in the quality of the finals results. The current version of Llama3 was the second fastest and had the best quality.
- **CPU Usage:**
  - Llama2:13b used the most CPU out of all of the models while Phi3 used the least CPU. The current version of Llama3 did use a little more CPU than Phi3 but not a huge amount more.
- **Quality:**
  - Llama3 without a doubt had the best quality out of the models. This was impressive considering that it was not much slower than Phi3 and quite a bit faster than Llama2:13b. I expected Llama2:13b to have the best in reasoning and knowledge but it made quite a few mistakes.
- **Summary/Reflection:**
  - Ollama3 was the best model from this selection and had the best balance of quality, speed, and resource usage.
  - Some of the insights I gained from this exploration include the different prompting techniques I learned about and how to properly prompt an ai for the most accurate results.
  - This will be great for me in real-world applications like at work. For example, the company I work for has started using ai to complete preliminary risk assessments for vendors and businesses we are considering working with. Knowing different prompting techniques will help streamline this process.
  - Some of the issues I faced included computer limitations, so I was unable to use larger models like Llama2:70b which required more disk space than I had available. This is why I decided to use Llama2:13b which was a smaller model, but I think it may have been better to use the 30b parameter model considering some of the weird mistakes the 13b model generated in the responses.