ASSIGNMENT 5
EXPLORING LARGE LANGUAGE MODELS

CS 478 NATURAL LANGUAGE PROCESSING (FALL 2024)
https://nlp.cs.gmu.edu/course/cs478-fall24/

OUT: October 28, 2024
DUE: November 13, 2024

Your name: Arron Birhanu

Your GID: G01315277

Overview and Submission Guideline: This assignment has two parts. Part 1 introduces you to the Prompt Engineering of large language models (LLMs), which needs to be completed **independently**. Part 2 requires you to implement an LLM agent, which can be done either **independently** or **in a team of no more than 3 people** in this class. Both parts require the use of LLM APIs from OpenAI (Part 1) and optionally other companies (Part 2).

How to fill out this PDF? This PDF also includes required contents for Part 1. In some answer blanks, you will be asked to provide your code implementation, whereas in others, you will only need to provide the execution output of your code, or describe what you observed. Please note that to obtain the full credits, your answers to the "what did you observe" type of questions need to be specific and clear. For Part 2, you will need to submit a separate report (see Part 2 in this PDF).

What and Where to Submit? To summarize, you will submit four items from this assignment:

- A completed PDF compiled from this LaTex source file to Gradescope for Part 1, which should be completed independently.
- A completed notebook for Part 1 to Blackboard. For ease of grading, we ask you to provide an accessible link to your notebook in the answer field below as well.
- A PDF report for Part 2 to Gradescope. This can be completed either independently or in a team.
- A zipped folder of your coding materials for Part 2 to Blackboard.

Note: Please make sure to REMOVE your private API key from your notebook (Part 1) and the code folder (Part 2) before submission. This key should not be shared with the instructor or the TA.

Your Notebook solution for Part 1:
https://colab.research.google.com/drive/1CDP0DAaIrMQI64SrnJnTTWbfUflBcheX?usp=sharing

Graded Questions: 100 points
Bonus Questions: 30 points
Total Points Available: 130/100

Part 1: Introduction to Prompt Engineering [40 pts]

This section uses the paired Python notebook. Make sure to set up the Python environment and OpenAI API key (Task 0) before you run the notebook.

Task 1: Story Generation with Different Sampling Strategies [20 points]

1. Can you use the ChatCompletion function, and "Generate a short story about a student studying abroad" as the prompt, to instruct GPT-4o-mini to generate a story? Please use the default setting for other configurations. [5 points]

Your Code Solution

```
prompt = "Emi, a student studying abroad, struggled with
homesickness and the language barrier in her new city. One day,
she met an elderly local who became her mentor, teaching her not
just the language but how to find joy in the unfamiliar. By the
end of her semester, Emi had transformed her struggles into a
journey of growth, discovering a second home in a place that once
felt foreign. Guess the country she visited."
ChatCompletion(prompt)

```

2. Now, can you do it with the same prompt but try to get 2 generations (keep "top_p" and "max_completion_tokens" default)? [2.5 points]

Your Code Solution

```
prompt = "Emi, a student studying abroad, struggled with
homesickness and the language barrier in her new city. One day,
she met an elderly local who became her mentor, teaching her not
just the language but how to find joy in the unfamiliar. By the
end of her semester, Emi had transformed her struggles into a
journey of growth, discovering a second home in a place that once
felt foreign. Guess the country she visited."
ChatCompletion(prompt, 2)

```

3. How about 2 generations with "top_p" set to be 0.1 (keep "max_completion_tokens" default)? [2.5 points]

Your Code Solution

```
prompt = "Marcus is learning the country's language very quickly thanks to a student exchange program at his college. Be sure to mention the country you think he's visiting!"
ChatCompletion(prompt, 2, 0.1)

```

4. Did the different "top_p" configurations give you the same or different results? Why do you think it could happen? [5 points]

Your Answer

The lower the "top_p" value, the lower the margin of textual difference is present between the "n" different responses.

5. Let's go back and do the same as in Question 2, but set the number of generations or samples to 10 (keep "top_p" and "max_completion_tokens" to be the default values). Read the 10 stories, check which country the character studies internationally, and report the statistics below (feel free to delete or add rows based on the responses you received). [5 points]

Your Answer

Country	Counts
Japan	8 out of 10
Germany	1 out of 10
France	1 out of 10

What did you observe? Are the generations regionally diverse? Why do you think it could happen?

Your Answer

I noticed that it favored Japan a lot during its country selection. There are plenty of countries where one can experience homesickness at, so there's likely some bias with the AI's choice which could be due to specific tokens inputted by other users. Furthermore, I ran this exact prompt several times to find that the countries would change for this prompt. However, Japan remained ahead of the other countries.

Task 2: GPT-4o-mini for Solving Mathematical Problems [20 points]

You are presented with a mathematical problem: *Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?* In this task, you will explore ways to prompt the LLM to solve the problem.

(Acknowledgment: The math problems used in this notebook come from the GSM8k dataset [1].)

6. Call the ChatCompletion function and prompt GPT-4o-mini to solve the problem. You can directly use "math_problem" as the prompt. To understand how reliably the LLM can solve this problem, let's generate 5 answers in a similar way as Q2 (keep "top_p" to be the default value but set "max_completion_tokens" to be '1000'). [10 points]

   Below, attach the first response (i.e., "Response 0") you received *(Note: in case your response is too long, truncate some middle parts to fit the height limit)*:

Your Answer

Response 0: Let's denote the number of vacuum cleaners Melanie started with as x.
1. **Sold at the green house:** She sold a third of her vacuum cleaners:

$$\text{Sold at green house} = \frac{x}{3}$$

After this sale, the number of vacuum cleaners left is:

$$x - \frac{x}{3} = \frac{2x}{3}$$

2. **Sold at the red house:** Next, she sold 2 more:

$$\text{Remaining after red house} = \frac{2x}{3} - 2$$

3. **Sold at the orange house:** Then, she sold half of what was left at the orange house:

$$\text{Sold at orange house} = \frac{1}{2}\left(\frac{2x}{3} - 2\right)$$

The number of vacuum cleaners left after this sale is:

$$\text{Remaining} = \left(\frac{2x}{3} - 2\right) - \frac{1}{2}\left(\frac{2x}{3} - 2\right)$$

To simplify this: - First express the term:

$$\frac{1}{2}\left(\frac{2x}{3} - 2\right) = \frac{2x}{6} - 1 = \frac{x}{3} - 1$$

Now, substituting this back we get:

$$\text{Remaining} = \left(\frac{2x}{3} - 2\right) - \left(\frac{x}{3} - 1\right)$$

Simplify this:

$$\text{Remaining} = \frac{2x}{3} - 2 - \frac{x}{3} + 1$$

$$\text{Remaining} = \left(\frac{2x}{3} - \frac{x}{3}\right) - 1 = \frac{x}{3} - 1$$

4. **Setting up the equation:** We know that after all these sales, she has 5 vacuum cleaners left:

$$\frac{x}{3} - 1 = 5$$

5. **Solving the equation:**

$$\frac{x}{3} = 6$$

$$x = 18$$

Thus, the number of vacuum cleaners Melanie started with is $\boxed{18}$.

Are all the solutions correct? Read them carefully and report what you observed (e.g., how did the model solve the problem? did it solve the problem in the same way in its 5 solutions? anything interesting from its solutions?).

Your Answer

All the solutions were correct. The manner in which each response approached the problem however differed. While response 0 utilized system of equations, response 3 used the substitution. This begs the question, how many different responses until we get a matching algorithm in the way the problem is solved ('top_p' becomes irrelevant)?

7. As you can see from the prior answer, LLM's response could be highly unstructured, so if you want to check its math-solving answer, you have to read the answer carefully. While the model solves the math problem step by step, it does not always indicate the step number. [10 points]

   To make the LLM response more structured, OpenAI has introduced the capability called "structured output". A brief guide of its usage can be found here.

   Can you complete the following code and define a "StructuredChatCompletion" function to return step-by-step math solution? Attach your code solution for the "TODO" parts below:

Your Code Solution

```python
from pydantic import BaseModel, Field
from typing import List

# TODO: Define your structured object
class MathStep(BaseModel):
    explanation: str = Field(..., description="A string providing an explanation for the current step")
    output: str = Field(..., description="The result or output of the current step")

class MathResponse(BaseModel):
    steps: List[MathStep] = Field(..., description="List of steps in the solution process")
    final_answer: str = Field(..., description="The final answer of the math problem")

def StructuredChatCompletion(prompt, n_samples=1, top_p=1.0, max_completion_tokens=500, return_object=False):
    assert n_samples >= 1
    assert top_p <= 1 and top_p > 0

    # TODO: add a `response_format` argument and specify the structured output format
    response_format = MathResponse

    completion = client.beta.chat.completions.parse(
        model="gpt-4o-mini-2024-07-18",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        n = n_samples,
        top_p=top_p,
        max_completion_tokens=max_completion_tokens,
        response_format=response_format
    )

    # The following code has been modified to parse the structure output. No further edits are needed.
    # ... [the remaining code is omitted here]


```

Use this `StructuredChatCompletion` function and prompt GPT-4o-mini to generate 5 solutions following the same setting as in Q6.

Include the first response you received (i.e., "Response 0") below:

Your Answer

```
*Response 0*:
[MathStep(explanation='Let the total number of vacuum cleaners Melanie
     started with be x.', output='x'), MathStep(explanation='She sold
    a third of them at the green house, so she sold (1/3)x there. This
     leaves her with (2/3)x vacuum cleaners.', output='(2/3)x'),
    MathStep(explanation='Next, she sold 2 more at the red house.
    After this sale, she has (2/3)x - 2 vacuum cleaners left.', output
    ='(2/3)x - 2'), MathStep(explanation='Then she sold half of what
    was left at the orange house. The amount left before selling at
    the orange house is (2/3)x - 2. Half of that is (1/2)((2/3)x - 2)
    = (1/3)x - 1.', output='(1/3)x - 1'), MathStep(explanation='After
    selling this half at the orange house, the remaining number of
    vacuum cleaners is: ((2/3)x - 2) - ((1/3)x - 1) = (2/3)x - 2 -
    (1/3)x + 1 = (1/3)x - 1.', output='(1/3)x - 1'), MathStep(
    explanation='We know that Melanie has 5 vacuum cleaners left, so
    we set up the equation: (1/3)x - 1 = 5.', output='(1/3)x - 1 = 5')
    , MathStep(explanation='Now we solve for x. Adding 1 to both sides
     gives us (1/3)x = 6. Multiplying both sides by 3, we find x = 18.
    ', output='x = 18')]
18
----------

```

Describe what you observed (e.g., did the model follow the structured format? did forcing it to structure its output hurt the model performance? share anything you found!)

Your Answer

It appears as if the model separated the ”steps” section of the response from the answer and the ”initial response” to the question. While the mathematical approaches remained similar, from a visual sense, there was definitely change. The steps are all inserted onto one line for the sake of visibility. Unfortunately, one approach resulted in a wrong answer (30) in response 2.

Part 2: Build an LLM Agent [60 pts + 30 bonus pts]

In this part, we will switch to an extended topic called "LLM agents". In Part 1, we have mainly used GPT-4o-mini as a question-answering system, but as we introduced in class, an LLM can be formulated to be a "vivid" agent that learns to use tools and helps us in more complicated tasks, just like a virtual assistant!

For Part 2, you are allowed to team up with your classmates (no more than 3 people in a team) and build your LLM agent together! There are two routes for you to consider, and you only need to pick one of them:

- **Route 1:** Building your own LLM agent using an open-source Python library called "Gentopia" [2]. The original library can be found here (check out its demo videos and documentation!), and we have made a customized version ("Gentopia-Mason") for this NLP class, which can be accessed here. Your task in this route is to implement your own *tool-augmented* LLM agent for anything you find helpful.
- **Route 2:** Evaluating an LLM on a public agent benchmark datasets. Researchers have introduced agent benchmarks such as OSWorld and WebArena to assess the progress of LLM agent advancement. Your task in this route is to use one of these benchmark datasets, configure the same sandbox environment, and evaluate one LLM on the dataset. You are not limited to using either OSWorld or WebArena; if you identify other agent benchmark datasets, you are welcome to try them!

Requirement: Each student or each team should perform "valid" explorations of LLM agents. For **Route 1**, at the minimum, you are expected to follow the README instructions of Gentopia-Mason and build the Scholar Agent *with one new function*. If you want to claim bonus credits (see **Bonus Points**), you are expected to build a new tool-augmented agent that is not provided by the Gentopia-Mason library. For **Route 2**, setting up the sandbox environments is not trivial, so it is suggested that you form a team (solo is not completely infeasible though). Concerning the timeline of this assignment and the cost of APIs, this route does NOT require you to run an LLM on the entire test set. Instead, your task is mainly to launch the sandbox environment and get an LLM to run in it for a few examples (say, even just 5 test examples). If you can do more than that (e.g., having a careful discussion of how the LLM works in such a complicated environment, analyzing its performance, discussing potential improvement, etc.), you deserve bonus points (see **Bonus Points**)!

What to Submit? For Part 2, you will submit (1) a PDF report shared among the team to Gradescope. See **Grading** for further requirements on writing. You/Your team will also need to submit (2) a zipped folder of relevant coding materials to Blackboard. If you Route 1, the materials include the code implementation of your LLM agent. Please make sure to include a README document describing how the grader can run your LLM agent for grading purposes. If you take Route 2 (which is mainly about evaluating an LLM), the materials include any log files or outputs generated during the evaluation process. For example, running an LLM on OSWorld will produce both log files and GUI screenshots of the agent's action trajectories in the operating system. If you have further code implementation reported in the PDF, the implementation should be submitted to Blackboard as well.

Grading: You can work independently or in a team with no more than 3 people (including yourself). Each team member will receive the same score. The following grading rubric will be "proportionally" adjusted when a team has more people. This part has a total of 60 points before bonus.

- **Writing [50 points]**: There is no a particular requirement on the report length or format, but the following content should be included and the writing should show sufficient clarity:
  - **What did you build? [10 points]** Start the report with a brief description of what you have built. For example, if you take Route 1, briefly what does your agent do? If you take Route 2, briefly what is the benchmark doing?

  - **Why is it important/useful/interesting? [10 points]** How do you think that the agent you built (Route 1) or evaluated (Route 2) can be helpful in real life?
  - **Describe your work. [20 points]** Now give more details about what you have built or evaluated– how the agent functions? what did you implement to let it work? etc. *Please include at least 1 figure showing an example of your agent's functions.*
  - **Discussion. [10 points]** Share anything you find interesting (or frustrating?) in the process! For example, did the agent powered by say GPT-4o-mini work as well as what you expected? did you see any surprisingly good or bad behaviors from the agent? If you have more time, how do you plan to improve the agent?

- **Coding Materials [10 points]**: We mainly check if you have actually implemented what you described in the report. If you take Route 1, again, make sure to include a README instruction for running your agent.

Bonus Points [up to 30 points]: We offer bonus points to students or teams who do excellent work! Note that "30 points" could immediately upgrade your letter grade–As Assignment 5 contributes to 15% of your final grade, 30 points at the scale of 100 full points translates to $30\% \times 15 = 4.5$ absolute points to your final grade!

We will provide bonus points mainly based on your workload (e.g., when the completion far exceeds the requirement), your passion (e.g., when you show to be very careful in implementing and discussing your agent), and your creativity (e.g., when the agent you implemented is particularly creative, or the evaluation you performed is particularly thought-provoking). The number of points offered will be decided by the grader and the instructor depending on the actual completion of each student or team. Students or teams delivering outstanding LLM agents or agent evaluations will be invited to present their work in the final lecture.

References

[1] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[2] Binfeng Xu, Xukun Liu, Hua Shen, Zeyu Han, Yuhan Li, Murong Yue, Zhiyuan Peng, Yuchen Liu, Ziyu Yao, and Dongkuan Xu. Gentopia.AI: A collaborative platform for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 237–245, 2023.