HOMEWORK 2

PRACTICES OF LARGE LANGUAGE MODELS[1]

CS 678 ADVANCED NATURAL LANGUAGE PROCESSING (SPRING 2024)
https://nlp.cs.gmu.edu/course/cs678-spring24/

OUT: February 20, 2024
DUE: March 8, 2024

Your name: _____

Your GID: _____

IMPORTANT: The homework is accompanied by a Python notebook and a GitHub code repository. **For Part 1**, after copying this notebook to your Google Drive or One Drive, please paste a link to it below. To get a publicly accessible link, hit the Share button at the top right, then click "Get shareable link" and copy over the result. **For Part 2**, fork the public code repository to your own space and share a link to your code repository in the link below. When you are done, submit your completed .pdf report to Gradescope.
If you fail to do this, you will receive no credit for this homework!

Note 1: You will only need to run lightweight code for this assignment, which can be done locally on your machine (i.e., ORC GPUs are not required).
Note 2: Both parts require using OpenAI API. Please make sure to REMOVE your private key from your notebook and code repository before submission.
Your Notebook solution for Part 1: _____
Your GitHub code repository for Part 2: _____

Graded Questions: 100 points
Bonus Questions: 20 points
Total Points Available: 120/100

Additional Notes:

- Upload the whole report PDF (including all pages, also if you're scanning it).
- All questions only require written answers in this PDF. For coding problems, you will have to fill out all code blocks that say `YOUR CODE HERE` in your notebook or per instruction.
- For text-based answers, you should replace the text that says "Write your answer here..." with your actual answer.

[1]Compiled on Tuesday 20th February, 2024 at 21:16

Part 1: Introduction to Prompts [50 pts]

This section uses the paired Python notebook. Below, provide the answers to the questions as generated by the completed code blocks in the similarly marked questions. Please also make sure to set up the Python environment (Task 0) before you run the notebook.

Task 1: Story Generation with Different Sampling Strategies

1. (5 points) Can you use the ChatCompletion function to generate a story about an Indian student studying abroad (e.g., at George Mason University)? Please use the default setting and generate only one story.

Give the output of your GPT-3.5 here.

2. (5 points) Now, can you do the same but try to get 2 generations with "top_p" set to be 1?

Give the output of your GPT-3.5 here.

3. (5 points) How about 2 generations with "top_p" set to be 0.5?

Give the output of your GPT-3.5 here.

4. (10 points) What did you observe from Q1 - Q3? Did the different "top_p" configurations give you the same or different results? Why?

Give the output of your GPT-3.5 here.

Task 2: GPT-3.5 for Solving Mathematical Problems

5. (5 points) You are presented 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?* Can you use the ChatCompletion function and prompt GPT-3.5 to work out the problem?

Give the output of your GPT-3.5 here.

Did GPT-3.5 solve the problem correctly? Where did it go wrong?

Provide your observations and findings here.

6. (10 points) Now, try to get 10 solutions from GPT-3.5 with "top_p" set to 0.7. Likely, you can now see multiple different answers produced by GPT-3.5. Summarize them in the table below and give one example output from GPT-3.5 each. Did GPT-3.5 do right in any of the solutions? What are the common errors that GPT-3.5 makes?

Answer	Count	Example Output
Ans1	counts out of 10	Give ONE output with this answer here
Ans2	counts out of 10	R2
Ans3	counts out of 10	R3

Provide your observations and findings here.

7. (10 points) Can you try other ways to prompt GPT-3.5 to give correct solutions more stably? Be creative! Include your prompt design and the answer in the report. Why do you think it works or not?

Prompt:
Provide your prompt here

Output:
Provide the GPT-3.5 output here

Findings: Provide your observations and findings here.

Part 2: Build an LLM Agent with Gentopia [50 pts + 20 bonus pts]

In this part, we will switch to an extended topic called "LLM agents". In Part 1, we have mainly used GPT-3.5 as a question-answer system, but an LLM can be formulated to be a "vivid" agent who learns to use tools and helps us in broader tasks, just like a virtual assistant!

To this end, we will be using an open-source LLM agent implementation framework, called "Gentopia" [2]. The original code repository can be found at https://github.com/Gentopia-AI, **but note that for this assignment, we will use this adapted "Gentopia-Mason" version at https://github.com/LittleYUYU/Gentopia-Mason/tree/main**.

Prerequsite: Library Installation

As the first step, please make sure to install all required Python packages following the instructions on README. More specifically, it means to run the following lines of code:

```
# Clone the repository
git clone git@github.com:LittleYUYU/Gentopia-Mason.git
cd Gentopia-Mason

# Create a conda virtual environment
conda create --name gentenv python=3.10
conda activate gentenv
pip install -r requirements.txt

# Set up global environment
export PYTHONPATH="$PWD/Gentopia:$PYTHONPATH"

# Set up OpenAI API key
cd GentPool
touch .env
echo "OPENAI_API_KEY=<your_openai_api_key>" >> .env
```

Task 3: Build a Vanilla LLM Agent

In this task, you will follow the instructions at https://github.com/LittleYUYU/Gentopia-Mason/tree/main?tab=readme-ov-file#quick-start-clone-a-vanilla-llm-agent to create a vanilla LLM agent. This agent is essentially the same question-answer model you have tried in Part 1 (though based on GPT-4 rather than GPT-3.5).

The agent, under the framework of Gentopia, is configured as follows (with annotations):

```
# ./GentPool/gentpool/pool/<your_agent_name>/agent.yaml
# Vanilla agent template

name: <your_agent_name>
version: 0.0.1
type: vanilla
description: A plain gpt-4 LLM. # this is a brief description of the agent
target_tasks:
  - anything to do with an LLM
llm:
  model_name: gpt-4 # indicating the API version
  params:
    temperature: 0.0 # both temperature and top_p are used to control the
    sampling diversity
```

```
    top_p: 0.9
    repetition_penalty: 1.0 # this is for preventing repetitive tokens
    max_tokens: 1024 # maximum generation length
prompt_template: !prompt VanillaPrompt # this is prompt we used for this agent,
    see ./Gentopia/gentopia/prompt/vanilla.py
```

8. (10 points) Successfully run this agent, and copy its responses for the following conversations below:

User: Hi! Tell me something about George Mason NLP group!
your_agent_name: *Copy the agent response here.*
User: Who is/are leading the group?
your_agent_name: *Copy the agent response here.*

What problems did you see from the second response?

Provide a short answer here

9. (10 points) Now, let's do a trick here. Instead of asking two questions, let's ask a combined one. And similarly let's also try the second question asking about facts:

User: Who is/are leading George Mason NLP group?
your_agent_name: *Copy the agent response here.*
User: Find papers written by Ziyu Yao at George Mason University
your_agent_name: *Copy the agent response here.*

How do the responses look now? (Hint: you may want to verify the agent output.) Why would this situation happen? Describe your observations and findings below:

Provide a short answer here

Till now, you probably have realized that an LLM alone will be facing various problems, including not being able to access the latest knowledge. Therefore, we'd like to try something called "tool-augmented LLM agent", where an LLM agent can be equipped with external tools for tasks. This is what we will do in Task 4!

Task 4: Build a *Tool-Augmented* LLM Agent

Now, we will follow the instructions at https://github.com/LittleYUYU/Gentopia-Mason/tree/main?tab=readme-ov-file#implement-a-scholar-llm-agent-with-tool-augmentati and create a tool-augmented LLM agent. In particular, this agent, called a "scholar agent", has been equipped with tools to access Google Scholar.[2]

This tool use does not come for free; it relies on Python implementation using the "scholarly" library,[3] which can be found within the gentopia source code at https://github.com/LittleYUYU/Gentopia-Mason/blob/main/Gentopia/gentopia/tools/google_scholar.py.

10. (10 points) Now, let's the same query for Prof. Ziyu Yao's papers again:

User: Find papers written by Ziyu Yao at George Mason University
your_agent_name: *Copy the agent response here.*

Does the response look reasonable now? Based on the meta information that popped out through the agent's responding process, can you describe how the agent made it right this time?

Provide your answer here.

11. (5 points) Look at the configuration file of your scholar agent at `./GentPool/gentpool/pool/<your_agent_name>/agent.yaml` and see what other functions are supported. Try a few other things you found interesting! Some examples to consider: *Can you summarize the paper titled "Gentopia: A collaborative platform for tool-augmented llms"?*, *What papers have cited "Gentopia: A collaborative platform for tool-augmented llms"*, etc.

User: *Your query*
your_agent_name: *Copy the agent response here.*

[2] https://scholar.google.com/
[3] https://pypi.org/project/scholarly/

12. (15 points) You may have realized that the current scholar agent still fails to answer general questions such as "Tell me something about George Mason NLP group", though they are also relevant to "scholar". In addition, it cannot read PDFs for summarizing papers either, so it cannot provide answers to specific questions about the paper details. Can you figure out a way for the agent to do Google Search (5 pts) and PDF reading (10 pts), and present a few examples below? (Hint: Google Search has been provided in Gentopia's source code, whereas you could opt for any Python tools you like, e.g., PyPDF2,[4] for PDF reading.)

User: *Your query*
your_agent_name: *Copy the agent response here.*

Bonus: Build Your Own Agent! (20 pts)

We will provide up to 20 points to students who implement a different tool-augmented LLM agent *for positive use (i.e., the agent cannot be designed for unethical purposes)*. To claim bonus points, the agent needs to be substantially different from the scholar agent and others included in Gentopia's agent pool. Be creative! Partial credits may be provided based on student effort.

Acknowledgment

The mathematical problems used in this assignment come from the GSM8k dataset [1], and the source code of Gentopia comes from the referred EMNLP'23 paper [2].

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.

[4] https://www.geeksforgeeks.org/working-with-pdf-files-in-python/