# CSCI 4820/5820 (Barbosa)
# Project 6: Prompt Engineering for GPT-2 Medium Model (345M) on an NLP Task

Due: **See course calendar**. May not be turned in late.

Assignment ID: **proj6**
File(s) to be submitted: **proj6.ipynb, proj6.pdf, proj6doc.pdf, <x><y>_input.txt**

345M Parameters

**Note:** Your submitted code must run on the **TAMU FASTER** system – Ensure you test your code there before submission.

****** This project requires the **Fall 2024 SIF** on TAMU FASTER (see the video on D2L) ******

**Objective(s)**: As demonstrated in class, the pre-trained 345 million parameter GPT-2 Medium model is the largest that will fit on a GPU with 24 GB GPU. The 345M model was one of the intermediate sized models mentioned in the GPT-2 paper, but was not the main model for which results were reported. In this project you will explore and report the capabilities of this model on an NLP task you choose (and that follows the usage guidelines in the paper/literature).

**Project description**: The starter code and required modified files to download and use GPT-2 Medium were demonstrated in class and have been tested on the TAMU FASTER system. Carefully follow notebook instructions.

**Note**: The **T4 GPU** on TAMU FASTER has **insufficient memory** for this task. Use one of the following GPUs: A10, A30, A40, A100. Note that there are fewer of these GPUs, so plan accordingly.

Evaluate and modify aspects of this model to incorporate the following:

- Ensure you download the starter code and zip file into the same directory. On the first run, execute cells one at a time and comment out those that need only run once.
- Recall that the GPT-2 model was trained on multiple tasks, without fine-tuning. Presenting prompts in the proper format to the model can induce responses on NLP tasks for which it was not trained. In addition to prompts used for the various tasks listed in the GPT-2 paper, a lot of work has been done by researchers and users to extract valuable output from the model. To evaluate the model:
  - Do some research and experimentation on prompt engineering to get answers to a non-trivial NLP task which you will define.
  - Explore with different tasks and prompts before choosing your task. Note: the exact format of many "successful" prompts are not always clearly stated, and one's interpretation of what was fed to the model can result in poor performance at times. For example: is an input a single string with x number of delimited components? OR is it a list or tuple composed of x strings? Ask yourself questions: What's the disconnect? Is there a similar prompt that works better? Is there a different way of presenting the data that is optimal for the task? What examples are likely to result in good model output.
  - **Clearly identify the sources from which you obtained ideas and submit at least 20 examples of prompts for the task you settled on (do not submit prompts for multiple tasks)**.
- Modify the *interact_model* function in the notebook so that it is NOT keyboard-interactive.
  - It should accept model prompts generated from a loop and return each output.
  - The model prompts should come from an input file - Prompt the model with your samples in an input file, so it runs without requiring user interaction at the keyboard.
  - Name the file *<initial of your first name><initial of your last name>_input.txt* (example: ***sb_input.txt***)
  - The file's format should include the complete input prompt (including any data), and the expected output prompt. The prompt and input should be separated by a tab.

- Only input prompts should be sent to the *interact_model* function. The expected responses are used for gauging the model's performance by comparing them with the output.
    - Submit an excerpt of at least **20 samples** for your chosen task in a file.
    - **Vary arguments to the interact_model function (length, temperature, top_k) to improve performance**.
- Trim the output: GPT is a generative model and will generate the requested number of tokens. The output should be processed to remove extra components. For example:
    - If a question is being asked of the model, and question/answer pairs are provided as examples, the model may go beyond answering the question posed and provide its own question/answer pairs (in addition to those asked, up to max tokens). These should be removed from the output.
    - Anything after the model outputs ***<|endoftext|>*** is usually outside the scope of what is expected from the prompt, and unrelated to the expected output. This should be removed from the output.
- **Display the input prompt and model output (on separate lines) for the samples in the input file. Separate the input/output pairs from one another with a blank line.**

**Requirements**: Where two weights are shown for an item, the first is for CSCI 4820 and the second for CSCI 5820 credit.

1. (**85%/70%**) Your program must provide the functionality listed below, in a single file named ***proj6.ipynb***:

    - Modify the code per these requirements. Determine the format of the prompt to get the best performance from the model on your task by consulting all resources at your disposal. **Ensure you properly cite/attribute all ideas/works you use in your submission**.

    **The quality of your chosen NLP task, the prompts used, and the output obtained from the model matter. Do not simply repeat examples covered in class (this will not earn significant credit)**.

2. (10**%**) Test your program – Test your code. Ensure your submitted model fully executes and produces results on the TAMU system.

3. (**5%**) Code comments - Add the following comments to your code:

    - Place a Markdown cell at the top of the source file(s) with the following identifying information:

        Your Name
        CSCI <course number>-<section number>
        Project #X
        Due: mm/dd/yy

4. (**CSCI 5820 Students only 15%**) Analysis – Provide an analysis (in a Markdown cell) addressing the following topics

    a) Describe at least one NLP task you considered/attempted and the reason(s) why you abandoned making that your main task (what you tried – specific prompts and input format, why you assumed it would work, and what steered you away from the task).
    b) Provide details the NLP task you settled on to demonstrate GPT-2 Medium capabilities, focusing on what the task is, the prompt you decided on, and the quality of the output. Give some examples of real-world problems that might use this task, and how your solution would fit into that architecture.
    c) Describe the process by which you arrived at your final prompt, by comparing it to at least two other prompting methods attempted that were not as successful.

5. Generate a pdf file of the notebook (from the terminal):

```
$ jupyter nbconvert --to html proj6.ipynb
$ wkhtmltopdf proj6.html proj6.pdf
```

6. Document all input and output from AI and other external tools and sources in the file ***proj6doc.pdf***

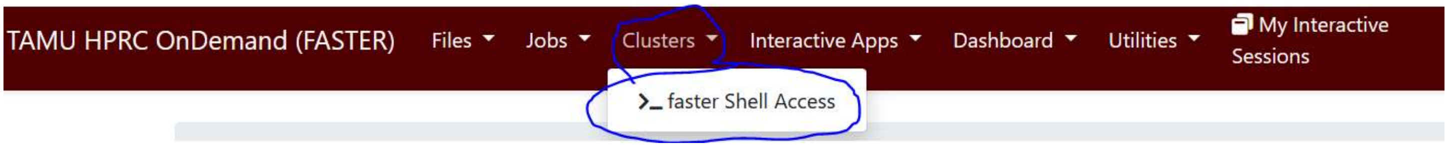
7. Submit required files via the D2L dropbox for this project. **Ensure your input file is submitted**.

# TAMU FASTER Troubleshooting

****** This project requires the **Fall 2024 SIF** on TAMU FASTER ******

If you have installed packages on TAMU FASTER prior to this assignment you may encounter some errors regarding missing or incompatible packages. The notebook has been tested and works on the default container without problems.

To **revert to default container**, log on to TAMU and before starting a notebook instance, bring up a shell as shown below:



Once in the shell change to your home directory (using the ***cd*** command) and type the following command (**note this command will remove all of the contents of the .local directory, including previously installed packages**):

```
rm –fr .local/*
```

Then instantiate a notebook following the video instructions on D2L.

After running any pip install command, an error may appear saying that one or more modules just installed were **not found**. Simply restart the notebook and re-run it.

Ensure that you comment out any cells that perform a pip install or download after the first time the notebook runs, to suppress unnecessary re-attempts and messages.