Generative AI Assignment 1

Due date: 10/20/2025 at 5:00 PM

Please only use a low cost LLM in your API calls since this assignment requires a lot of API calls. If you use a model on a high end you will quickly run out of credits since these API calls are much more expensive. It is also best to prototype/experiment your code on a small set of your data and keep the full dataset for final runs. This keeps the number of API calls to a reasonable level.

Prompt Engineering: Manual and Automated

1. Find task/data and create scoring method

1. You have a choice whether you want to use your own task/data or one that is provided below. We suggest you read through the full assignment before you make that decision. Some examples of prompt engineering datasets for question-answering are given below. You should select a subsample of around 250-500 questions from these datasets.
    a. BIG-Bench-Hard tasks https://github.com/suzgunmirac/BIG-Bench-Hard/tree/main/bbh
    b. GSM8K (Grade School Math 8K) https://huggingface.co/datasets/openai/gsm8k
    c. TruthfulQA https://github.com/sylinrl/TruthfulQA/tree/main/data/v1
        i. The answers to these questions are free-formatted text, so evaluating if the LLM answered it correctly is more challenging. See Step 1.3b for more information.
    d. Ensure that the LLM answers incorrectly for some of the base questions on these datasets without prompt engineering
2. If you are not using a prompt engineering dataset we provide above, come up with your own task/questions and create at least 100 questions that you want an LLM to answer. These should just be the basic questions without any prompt engineering.
    a. You should know the answer to these questions (labeled with correct answer)
    b. Some of the questions should not be answered correctly by your LLM in the base question form (without prompt engineering). This is why it may be easier to use one of the question-answer datasets we share above.
3. Create a scoring function that takes in a prompt, runs the prompt through your LLM, parses the output for the answer, and checks if the parsed answer is correct or not

    a. At the end of the prompt you will need to have a line that specifies how to format the final answer so that you can properly parse the LLM output for the answer.
        i. Example: “Output the final answer at the end with the prefix ####. For example: #### ANSWER”.
    b. If the answer is something like a number, then this can be checked for accuracy easily. However, for free-form text-based answers this is more difficult to handle. One approach is to use a scoring LLM that you ask if the provided answer is semantically similar to the ground truth correct answer.
4. Iterate over each base prompt with the scoring function you created and find the percentage of correct answers using the base prompt without prompt engineering.

2. Create manual prompt engineering improvements and examine change in score

1. Improve your base prompt with best practice prompt engineering techniques including specifying role, giving detailed instructions, asking the model to reason through the question, etc. Iterate over each improved prompt (you can likely use the same prompt improvements to all of your questions) with the scoring function you created and find the percentage of correct answers using the improved prompt.
2. Improve your prompt with **one** of the following manual prompt engineering techniques covered in the lecture. Then repeat the scoring process with this new prompting technique and find the new accuracy.
    a. Few-shot prompting
    b. Chain of thought prompting
    c. Tree of thoughts prompting
    d. Meta prompting
    e. Self-consistency

3. Implement an automated prompt engineering method and examine change in score

1. Improve your base prompt with **one** automated prompt engineering technique. Then repeat the scoring process with this new prompting technique and find the new accuracy. If your method requires some questions/prompts to use for “training” then you will either need to create new questions for this training or take more questions from the dataset you sampled from and only use the original questions for evaluation.
    a. Google OPRO https://arxiv.org/pdf/2309.03409

b. A creative automated algorithm that you come up with. You will need to implement the algorithm yourself.

4. Submit assignment

1. This assignment (and all future) requires you to use Docker. You will need to submit a Dockerfile with your submission.
2. Submit the final project on Github. Include all the code files (this includes any data processing files – if you used that code for your project, we need to see it), a README on how to run the code (including where to download data), the Dockerfile, and a document showing sample outputs from your code.

Grading

1. Scoring
    a. Scores properly designed (e.g. using F1 on unbalanced data, using BLUE for text outputs) [5 pts]
    b. Scoring properly implemented [10 pts]
2. Enhanced manual prompt engineering
    a. Baseline reasonably set up (starting deliberately with a questionable prompt is not a reasonable setup) [5 pts]
    b. At least one prompt improvement technique that outperforms the baseline identified [20 pts]
3. Fully automated prompt engineering
    a. Correctness of the implementation [20 pts]
    b. Algorithmic prompt engineering that beats the best prompt from the manual task [30 pts]
4. Reproducibility of the code
    a. Successfully running Dockerfile on Mac with Apple silicon (platform is arm64) [5 pts]
    b. Readability of README [5 pts]