COMP 10261: The GPT Chat Bot Project

Sam Scott, Mohawk College, 2023

In this project you will develop a Discord bot that uses one or more OpenAI models to produce responses to the user. You will make use of competion and/or chat models, and you will engage in creative prompt engineering, appropriate parameter use, and possibly even fine-tuning to get the behavior you want from the bot.

For this project, you can work with a partner if you like. If you choose to do that, you'll be asked to fill in a report that details what each of you did for the project. Each partner is expected to do some work on every aspect of the project (coding, prompt engineering, testing, report writing, etc.)

The Focus of the Bot

Your bot should have a focus, task, or knowledge area that it is tuned for, but the exact nature of that focus, task, or knowledge area is up to you.

Here are some ideas:

- ✓ A reboot of your FAQ Plus bot, better, stronger, and with some ability to provide answers to questions it hasn't seen before.
- An expert assistant, a bit like an FAQ bot, but tuned to a certain knowledge area and able to answer questions, give advice based on its knowledge area, etc.
- A problem-solver that is tuned to solving a particular type of math problem, logic problem, riddles, etc. (this could involve a function calling model)

If you have an idea for the bot but you're not sure if it would be appropriate, ask your instructor.

The API

You are to use at least two different OpenAI API calls in response to every utterance from the user. You should use appropriate **engine**, **temperature**, and **max_token** settings for each call to the API. You are required to use the completion API for at least one call. The use of other APIs (chat, insert, edit, etc.) is optional – use whatever is most appropriate to the task.

Using two API calls might not be 100% realistic depending on the nature of your bot, but this is a learning exercise. Using two API calls is a chance to do two different types of prompt engineering and/or use two different types of models.

Here are some ideas for how to incorporate 2 different API calls.

- ✓ Use the first call to classify the utterance (see the material on machine learning next week), and the second to answer the question. For example, the first could classify as relevant vs irrelevant, question vs statement vs greeting, rude vs polite, happy vs sad vs angry, etc. Then this could gate the response. Maybe angry utterances get a stock response, or maybe you use a different approach to prompt engineering to answer questions vs. statements?

- Use the two different calls to respond to different aspects of an utterance. Perhaps the first call responds based on tone while the second responds based on the content (perhaps prompt-engineered by the tone as well?)
- Use one call to maintain a summary of the dialog, then use that summary for dialog management in the second call.
- Use two different prompt engineering techniques to generate two different responses. Then make a third API call to decide which of the two is most appropriate based on some prompt-engineered criteria.
- Combine prompt engineering with a function call to answer some queries from the user. This requires a second call to the API after you have called the function.

Prompt Engineering

You must use careful, creative, and thoughtful prompt engineering for all API calls. You should test thoroughly and try to jailbreak your app as much as possible. See if you can get it to violate the rules you set out in the prompt, veer from the topic at hand, etc. Then try to plug the holes you find with better prompt engineering. Keep a record of your test cases. It's not expected that you will be 100% successful, but your prompts should show that you have thought carefully about how to instruct the bot.

As an alternative, you could also attempt some fine-tuning on one of the base GPT-3 models. But unless you have a large data source, prompt engineering the variously tweaked GPT-3 models will likely get you better results.

Dialog Management

The bot should have some form of dialog management. I should be able to ask it follow-up questions and it should be able to respond based on recent context.

Discord

Your bot should interact with the user through discord. Make sure that you have some kind of method of managing what the bot responds to (a command prefix, a special channel, etc.) It should not respond to every message in the server.

Testing

You should assemble a list of at least 20 test cases, including both simple and dialog-specific test cases.

A simple test case should look like this:

> **Prompt:** <the exact text of a prompt you will use>
> **Response:** <a description of what should be in the response (it might not always be exactly the same if the temperature is not 0)>

A dialog-specific test case should look like this:

> **Dialog:** <a record of past utterances and responses – doesn't have to exactly match>
> **Prompt:** <the exact text of a prompt you will use>
> **Response:** <a description of what should be in the response, highlighting the importance of responding in the context of the dialog>

Your test cases should include simple functionality tests (success path), as well as tests in which the prompt is off topic or a jailbreak attempt (failure path). You should also include test cases that show off the cool things the bot can do.

Reporting

Write a quick report with the following content:

Name of the Bot

Give it a cool name!

Description of the Bot

One short paragraph. What is the focus, task, or knowledge area of your bot? What kinds of things can it do?

Invitation Link

Link to a bot invitation (not a server invitation)

API Calls

One short paragraph for each API call. How did you use each API call? For each call: what is it for, what settings did you use, and how did you use the output?

Prompt Engineering

One short paragraph for each API call. What was the approach to prompt engineering? You don't have to list all the text, just describe what sort of prompt engineering you used (e.g., context, examples, chain of thought, constrained, etc.) and describe where you got the text for any examples, context, constraints, etc.

Test Cases

List your 20 test cases in the format shown above. Give each one a score (Pass, Partial Pass, Fail)

Performance

How well does your Bot work? If you were going to keep working on it, what changes would you make to improve it?

Handing In

Zip up all your code, supporting files, and report. Make sure that you have included your API key so that the instructor can run the bot. Don't forget the invitation to the bot!

~~If you worked with a partner, each partner should separately hand in a short Statement of Effort. This statement should include: Your name, your partner's name, the name of the bot, the percentage of the total work you performed towards the creation of the bot, and then a paragraph that summarizes your contributions to that work (coding, testing, prompt engineering, report writing, etc.)~~

Evaluation

See the rubric on Canvas for the evaluation method.