Fine-tuning and Assistants

can code
communities

Review Questions

- Why are API keys used?
- What are the necessary parameters required to make a chat completions API call?
- What is function calling?

Take Home Exercise Review

Create a function calling chatbot script that supports parallel function calling. Have at least 3 functions listed in your tool calls and try to call them all in one prompt.

Fine-Tuning

Fine-tuning is a model customization technique that adds more training data to a model.

- Basically a step further than few shot learning.
- Used to get higher quality responses for task specific prompts.

Why use Fine-Tuning?

- Set the style, format, and tone.
- Improve reliability and handling edge cases
- Further its ability to answer new or complex tasks
- Reduce token costs and latency

Fine-tuning is helpful, but isn’t always necessary.

- Prompt engineering, prompt chaining, and function calling are tools that should be tried before spending the time and effort to fine-tune a model.

How to Fine-Tune

- The training data needs to be prepared and uploaded.
  - The quality of the data is an important factor to the quality of the fine-tuned model responses.
- A fine-tuning session needs to be run and then tested.
  - Evaluating the fine-tuned model will represent the impact of the training data to the model.
- Depending on the model evaluations, another round of fine-tuning might be needed.
- Once the final fine-tuned model is trained and evaluated, it can be used the same way the base GPT models are within the chat completions API.

Preparing Datasets

- Fine tuning can be done with all different types of models (natural language, computer vision, etc.)
  - Preparing a dataset for different models will have different formats.
- Pair the prompt and the response in a way that the specific model requires.

OpenAI Fine-Tuning

OpenAI’s fine-tuning format follows the format of the message list that is created for the chat completions API.

- This message list gets put into a dictionary where the key is “messages” and the value is that message list.
- It is common to only have one prompt and response pair, but it is also acceptable to build out more pairs in the same data point to train the model for a longer conversation.
  - There is an extra key in the dictionary of each assistant message in the message list called “weight”.
  - This holds a binary value that signifies whether the model should train on that assistant message (response).
  - If the “weight” key isn’t defined, it will default to 1, which means it will train on that prompt and response pair.

Dataset Size

The amount of data points that are needed to fine-tune a model effectively is dependent on the task and use case.

- OpenAI has a minimum limit of 10 examples
- Recommends 50-100 examples.
- Building the size of the dataset should help with the quality of the responses, but if there isn't any improvement shown during evaluations, that indicates that approach or the data should be reconsidered.

Train/Test Split

Once the dataset is created, splitting the full dataset into train and test datasets helps the user get better insights within training and better evaluations after training.

A normal split would be somewhere between 80/20 and 70/30.

Assistants

Assistants is OpenAI’s most recent feature in their API. The main advantage that assistants have over chat completion is knowledge retrieval and code interpreter.

- Knowledge retrieval allows files with supported file types to be uploaded and referenced outside of the context window of the model.
- Code interpreter is OpenAI’s way of writing and running code in a sandbox environment, acting somewhat like an auto coder.

Threads and Runs

Assistants operate within a thread

- Stores the chat history as well as manages the size of the history in relation to the context window limits.

Since the thread is stored in OpenAI’s storage, they are executed using runs

- Allow the assistant to generate a response with the tools available to it and the context within the thread.

When a run is done, the assistant response gets written to the thread.

- This adds some complexity when retrieving the response in a script. Every run has a run lifecycle that we can check the status of through the API.

Assistants (Beta)

- The Assistants API has been in beta for over 6 months
- The main issues
    - The retrieval isn’t consistent
    - The usage price is significantly more than other APIs due to the storage aspect (once the thread gets to a certain size, the maximum amount of tokens will be used on every run).
- Both of these “issues” can also be flipped as complements
    - OpenAI’s knowledge retrieval tool is one of the first accessible and user friendly retrieval tools available
    - The usage cost isn’t absurd for the complexity in the features it supports (retrieval, code interpreter, thread storage/management).

Overall, assistants is a helpful tool, but not ready to be implemented into any commercial tools, and should be used only when it’s needed.

Take Home Exercise

Use the Assistants and Playground tabs on OpenAI to create a coding assistant that helps build a simple pong game.