Building Text Generation Applications

Introduction

In this chapter, you will:

- Learn about the openai library and it's core concepts.
- Build a text generation app using openai.
- Understand how to use concepts like prompt, temperature, and tokens to build a text generation app.

Learning goals

At the end of this lesson, you'll be able to:

- Explain what a text generation app is.
- Build a text generation app using openai.
- Configure your app to use more or less tokens and also change the temperature, for a varied output.

What is a text generation app?

Normally when you build an app it has some kind of interface like the following:

- Command-based. Console apps are typical apps where you type a command and it carries out a task. For example, `git` is a command-based app.
- User interface (UI). Some apps have graphical user interfaces (GUIs) where you click buttons, input text, select options and more.

Console and UI apps are limited

Compare it to a command-based app where you type a command:

- **It's limited**. You can't just type any command, only the ones that the app supports.
- **Language specific**. Some apps support many languages, but by default the app is built for a specific language, even if you can add more language support.

Benefits of text generation apps

So how is a text generation app different?

In a text generation app, you have more flexibility, you're not limited to a set of commands or a specific input language. Instead, you can use natural language to interact with the app. Another benefit is that because you're already interacting with a data source that has been trained on a vast corpus of information, whereas a traditional app might be limited on what's in a database.

What can I build with a text generation app?

There are many things you can build. For example:

- **A chatbot**. A chatbot answering questions about topics, like your company and its products could be a good match.
- **Helper**. LLMs are great at things like summarizing text, getting insights from text, producing text like resumes and more.
- **Code assistant**. Depending on the language model you use, you can build a code assistant that helps you write code. For example, you can use a product like GitHub Copilot as well as ChatGPT to help you write code.

How can I get started?

Well, you need to find a way to integrate with an LLM which usually entails the following two approaches:

- Use an API. Here you're constructing web requests with your prompt and get generated text back.
- Use a library. Libraries help encapsulate the API calls and make them easier to use.

Libraries/SDKs

There are a few well known libraries for working with LLMs like:

- **openai**, this library makes it easy to connect to your model and send in prompts.

Then there are libraries that operate on a higher level like:

- **Langchain**. Langchain is well known and supports Python.
- **Semantic Kernel**. Semantic Kernel is a library by Microsoft supporting the languages C#, Python, and Java.