LangChain

can code communities

LangChain

LangChain is a framework that is focused on building applications that implement large language models. We can use OpenAI models within LangChain and utilize some helpful tools that are a bit more tailored for the user experience.

Getting Started

- First we need to install the langchain library using ‘pip install langchain’
- We are going to store the OpenAI API key in a .env file, so that it can be used in LangChain.
    - LangChain supports many different models, so we can store multiple API keys within the .env file and use them throughout our script.
- Once we have the OpenAI model set up with its necessary API key, we can use the ‘invoke’ function to simply prompt the model.

Prompt Template

- We can build out more detailed prompts by using prompt templates.
- ChatPromptTemplate mimics the message list we've seen with OpenAI
  - The system message is set and the prompt can be determined by the user later on.
- We can also make the prompt template even more task specific and user friendly.
  - Partial prompt templates allow for variables to be set in a prompt that aren't controlled by the user.
  - Few-shot prompt templates use the few shot prompting technique to help format the output with a couple of examples.

Output Parser

- Whenever we make an API call using the ‘invoke’ function, we get an AIMessage response object returned back to us.
  - This holds a lot of helpful extra information about usage and an ID to track specific runs
  - When building applications, all we need to display is the generated content of the response object.
- We can use a string output parser in this case that returns the content without having to find the correct path of parameter names.

Exercise

Chain together a prompt template, an OpenAI model, and a string output parser together to make a recipe generator for any meal the user inputs.

Prompt Chaining

- A technique that breaks down a prompt into multiple subtasks.
  - Similar to Chain of Thought prompt engineering, but different because these subtasks are separate prompts.
- Prompt chaining is typically used for longer generations or complex tasks that need more direction in its execution.

Tool Calling

- Tool calling is synonymous with the term function calling.
- Allows the model being used in the application to choose a function from the list provided instead of generating a natural language response.
- We can define our own tools using the ‘@tool’ decorator
    - There has to be a docstring that describes the function in natural language for the model to understand the proper use of the function.

Tool Calling

We just went over how to create our own functions and pass them as available tools for the model to choose from, but LangChain also provides a lot of tools and toolkits that save developers a ton of time when connecting other programs or services to the model.

Agents

- Looking through LangChain's tools and toolkits, we can see that some tool instructions incorporate agents to perform the desired task.
    - We typically see the use of agents with toolkits because there are multiple tools to choose from.
- An agent chooses the sequence of actions to take using a reasoning engine.
- This allows the model to take control and complete a more versatile range of tasks.
    - Instead of being hard coded down a reasoning path in a chain.

Agent Types

There are a few different types of agents that we can use that LangChain provides.

- https://python.langchain.com/docs/modules/agents/agent_types/

Take Home Exercise

Use one (or more) of the simple to use tools that LangChain provides to create an agent.

- AlphaVantage
- Brave Search
- Dall-E Image Generator
- OpenWeatherMap
- SerpAPI
- WolframAlpha