CoderDojo Jupyter_Ollama_ChatOpenAI

Installing Jupyter

Get up and running on your computer

Jupyter Install Guide

Project Jupyter's tools are available for installation via the Python Package Index, the leading repository of software created for the Python programming language.

This link above uses instructions with pip, the recommended installation tool for Python. If you require environment management as opposed to just installation, look into conda, mamba, pipenv, and Homebrew.

JupyterLab

Install JupyterLab with `pip`:

```
pip install jupyterlab
```

Note: If you install JupyterLab with conda or mamba, we recommend using the conda-forge channel.

Once installed, launch JupyterLab with:

```
jupyter lab
```

Jupyter Notebook

Install the classic Jupyter Notebook with:

```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

Ollama: Local LLM Platform

Ollama is a platform that allows running large language models(LLM) locally on your machine, providing access to models like **Llama3.2** without needing cloud infrastructure. It is commonly used for natural language processing tasks such as text generation or chatbot development.

Steps to use Ollama Models:

- Download Ollama
  - https://ollama.com/download
- Install Ollama for Python

```
    pip install ollama
```

- Download an LLM Model from Ollama
  Visit the Ollama Library's to find available models
  Example to pull the LIama3.2:8b model:

```
        ollama pull llama3.2:8b
```

Simple Code to Ask LLM Model

The LangChain OpenAI integration lives in the `langchain-openai` package so you need to install it with `pip`

```
%pip install -qU langchain-openai
```

Import the library's that will be needed in Jupyter to chat with ollama

```
import warnings
warnings.filterwarnings('ignore')
import os
import requests

from langchain_openai import ChatOpenAI
```

Define the ChatOpenAI required fields with fake information so api calls are made to ollama and not to OpenAI.

```
OLLAMA_BASE_URL = "http://localhost:11434/v1"
OLLAMA_API_KEY = "ollama"
```

Config your model for Ollama

```
model = ChatOpenAI(
    model="llama3.2", # Specify the LLM model to use
    temperature=0.8,
    max_tokens=None,
    timeout=None,
    max_retries=2,   # Number of time call the ollama api
    api_key=OLLAMA_API_KEY,
    base_url="http://localhost:11434/v1"
)
```

Lets add a message and give role of system and human with p

```
messages = [
        (
                "system",
                "You are a helpful assistant that translates English to . Translate the user sentence.",
        ),
    (
    "human", "I love Artificial intelligence."
    ),
]
ai_msg = model.invoke(messages)
ai_msg
```

Better code for Readability

Import new library for displaying in markdown format add to top of notebook next to imports.

```
from IPython.display import Markdown, display
```

We will also update the messages with new system and human so replace current with :

```
messages = [
        (
                "system",
                "You are a helpful assistant that help users plan and build a new gaming PC by outlining the
necessary components, considerations, and steps involved in the process. Provide clear guidance for selecting
compatible parts and assembling them to create a functional gaming system."
        ),
    (
    "human", "List the components and steps in order, providing advice and recommendations for varying
budgets. The output should be in markdown format, with labeled sections for each step."
    ),
]
ai_msg = model.invoke(messages)
ai_msg
```

Then we will print out the Markdown version of the messaged returned by Ollama Models :

```
display(Markdown(f"** LLM Resonse: **\n\n{ai_msg.content}"))
```