COPILOT USING LANGCHAIN AND HUGGING FACE

-By Amisha Dhiman

Objectives

- Integrate LangChain with Hugging Face:
- Goal: Utilize the LangChain library to interact with Hugging Face models, leveraging their capabilities for tasks such as text generation and question answering.
- Demonstrate Usage:
- Show the setup and usage of different components, including pipelines for text generation and querying models for specific tasks.

Methodology

Setup Environment:

- **Install Libraries:**

langchain-huggingface: Connects LangChain with Hugging Face.

huggingface_hub, transformers, accelerate, and bitsandbytes: Access and use Hugging Face models, and manage performance.

Authenticate and Configure:

- **Get API Token:**

Retrieve the Hugging Face API token from Colab's user data storage and set it as an environment variable for model access.

- **Mount Google Drive:**

Used for accessing and saving files in Google Drive, though not directly related to Hugging Face.

Using LangChain with Hugging Face:

- **Create Model Instances:**

HuggingFaceEndpoint: Interact with Hugging Face models via their API by specifying model repository and setting parameters like max_length and temperature.

HuggingFacePipeline: Wraps Hugging Face pipelines for easier interaction with models, particularly for text generation.

- **Invoke Models:**

Text Generation: Generate responses based on provided prompts (e.g., "What is machine learning?").

Question Answering: Use prompt templates to ask specific questions and retrieve answers (e.g., "Who won the Cricket World Cup in the year 2011?").

Optimize Performance:

- **Using GPU:**

Configure the model to run on a GPU if available for faster performance by specifying the device argument when creating the HuggingFacePipeline instance.

Process

- **Setup Libraries:**

Install the necessary Python libraries for using Hugging Face models and LangChain.

- **Authenticate:**

Retrieve and configure your API token to gain access to Hugging Face models.

Create Model Instances:

Use HuggingFaceEndpoint and HuggingFacePipeline to set up models for text generation and other tasks.

- **Run Queries:**

Use the created model instances to generate text or answer questions.

- **Optimize (Optional):**

Configure the model to use GPU for enhanced performance.

Conclusion

You have developed a system that integrates Hugging Face models with LangChain, enabling:

Text Generation: Create text based on prompts.

Question Answering: Retrieve answers to specific questions using predefined templates.

Performance Optimization: Leverage GPU if available for faster processing.

This setup is valuable for applications requiring advanced natural language understanding and generation, facilitating the incorporation of sophisticated AI capabilities into your projects.

Model Summary and Parameters

1. Mistral-7B-Instruct-v0.2

- **Model Repository**: mistralai/Mistral-7B-Instruct-v0.2
- **Purpose**: Designed for instruction-following tasks and text generation.

Parameters:

- **max_length**: 128
  Description: Limits the maximum number of tokens to generate in the output. (Note: This parameter is passed through model_kwargs.)
- **temperature**: 0.7
  Description: Controls the randomness of the generated text. Lower values yield more focused and deterministic outputs, while higher values introduce more variability.

2. GPT-2

- **Model Identifier**: gpt2
- **Purpose**: A widely used model for text generation tasks.

Parameters:

- **pipeline**: text-generation
  Description: Specifies that the pipeline is used for generating text.
- **max_new_tokens**: 100
  Description: Limits the number of new tokens generated in the response.

GPU Configuration (Optional)

- **device**: 0
  Description: Specifies that the model should use the first GPU. If omitted, the model defaults to CPU.

Authentication

- **Environment Variable**: HUGGINGFACEHUB_API_TOKEN
  Description: Used for authenticating access to Hugging Face models.

Notes

- Warnings related to max_length and token suggest these parameters are handled via model_kwargs rather than directly in model initialization.

Huggingface With Langchain

Announcement Link: https://huggingface.co/blog/langchain (https://huggingface.co/blog/langchain)

In [*]:
```
## Libraries Required
!pip install langchain-huggingface
## For API Calls
!pip install huggingface_hub
!pip install transformers
!pip install accelerate
!pip install  bitsandbytes
!pip install langchain
```

```
Collecting langchain-huggingface
  Downloading langchain_huggingface-0.0.3-py3-none-any.whl (17 kB)
Requirement already satisfied: huggingface-hub>=0.23.0 in c:\users\dhima\anaconda3\lib\site-packages (fro
m langchain-huggingface) (0.24.5)
Collecting langchain-core<0.3,>=0.1.52 (from langchain-huggingface)
  Downloading langchain_core-0.2.31-py3-none-any.whl (389 kB)
                                              0.0/389.7 kB ? eta -:--:--
     -----------------                      184.3/389.7 kB 3.7 MB/s eta 0:00:01
     -------------------------------------- 389.7/389.7 kB 6.0 MB/s eta 0:00:00
Collecting sentence-transformers>=2.6.0 (from langchain-huggingface)
  Downloading sentence_transformers-3.0.1-py3-none-any.whl (227 kB)
                                              0.0/227.1 kB ? eta -:--:--
     ------------------------------------- 227.1/227.1 kB 13.6 MB/s eta 0:00:00
Collecting tokenizers>=0.19.1 (from langchain-huggingface)
  Downloading tokenizers-0.20.0-cp311-none-win_amd64.whl (2.3 MB)
                                              0.0/2.3 MB ? eta -:--:--
     ------------                             0.7/2.3 MB 21.8 MB/s eta 0:00:01
     ---------------------------------        1.9/2.3 MB 24.5 MB/s eta 0:00:01
     ---------------------------------------- 2.3/2.3 MB 18.5 MB/s eta 0:00:00
```

In [*]:
```
## Environment secret keys
from google.colab import userdata
sec_key=userdata.get("HF_TOKEN")
print(sec_key)
```

HuggingFaceEndpoint

How to Access HuggingFace Models with API

There are also two ways to use this class. You can specify the model with the repo_id parameter. Those endpoints use the serverless API, which is particularly beneficial to people using pro accounts or enterprise hub. Still, regular users can already have access to a fair amount of request by connecting with their HF token in the environment where they are executing the code.

In [*]:
```
from langchain_huggingface import HuggingFaceEndpoint
```

In [*]:
```
from google.colab import userdata
sec_key=userdata.get("HUGGINGFACEHUB_API_TOKEN")
print(sec_key)
```

In [*]:
```
import os
os.environ["HUGGINGFACEHUB_API_TOKEN"]=sec_key
```

In [*]:
```
repo_id="mistralai/Mistral-7B-Instruct-v0.2"
llm=HuggingFaceEndpoint(repo_id=repo_id,max_length=128,temperature=0.7,token=sec_key)
```

In [*]:
```
llm.invoke("What is machine learning")
```

In [*]:
```
repo_id="mistralai/Mistral-7B-Instruct-v0.3"
llm=HuggingFaceEndpoint(repo_id=repo_id,max_length=128,temperature=0.7,token=sec_key)
```

In [*]:
```
llm.invoke("What is Genertaive AI")
```

In [*]:
```
from langchain import PromptTemplate, LLMChain

question="Who won the Cricket World Cup in the year 2011?"
template = """Question: {question}
Answer: Let's think step by step."""
prompt = PromptTemplate(template=template, input_variables=["question"])
print(prompt)
```

In [*]:
```
llm_chain=LLMChain(llm=llm,prompt=prompt)
print(llm_chain.invoke(question))
```

HuggingFacePipeline

Among transformers, the Pipeline is the most versatile tool in the Hugging Face toolbox. LangChain being designed primarily to address RAG and Agent use cases, the scope of the pipeline here is reduced to the following text-centric tasks: “text-generation", “text2text-generation", “summarization”, “translation”. Models can be loaded directly with the from_model_id method

In [*]:
```
from langchain_huggingface import HuggingFacePipeline
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
```

In [*]:
```
model_id="gpt2"
model=AutoModelForCausalLM.from_pretrained(model_id)
tokenizer=AutoTokenizer.from_pretrained(model_id)
```

In [*]:
```
pipe=pipeline("text-generation",model=model,tokenizer=tokenizer,max_new_tokens=100)
hf=HuggingFacePipeline(pipeline=pipe)
```

In [*]:
```
hf
```

In [*]:
```
hf.invoke("What is machine learning")
```

In [*]:
```
## Use HuggingfacePipelines With Gpu
gpu_llm = HuggingFacePipeline.from_model_id(
    model_id="gpt2",
    task="text-generation",
    device=0,  # replace with device_map="auto" to use the accelerate library.
    pipeline_kwargs={"max_new_tokens": 100},
)
```

In [*]:
```
from langchain_core.prompts import PromptTemplate

template = """Question: {question}

Answer: Let's think step by step."""
prompt = PromptTemplate.from_template(template)
```

In [*]:
```
chain=prompt|gpu_llm
```

In [*]:
```
question="What is artificial intelligence?"
chain.invoke({"question":question})
```

In [ ]: