**Lab Exercise 8- Building an Agent using Azure OpenAI with requests**

**Objective:**

**Lab Exercise: Building an Agent using Azure OpenAI with requests**

Since we cannot use azure-ai-openai, we'll make HTTP requests directly to Azure OpenAI API using requests.

**Step 1: Set up Google Colab Environment**

You need to install the requests library in Colab, which will help us make API calls to Azure OpenAI.

!pip install requests

**Step 2: Define the Azure OpenAI API Connection**

Now, you need to use your **Azure OpenAI API key** and **Endpoint** from your Azure portal. This information will be used to authenticate the API requests.

Replace the placeholders '<Your\_Azure\_API\_Key>' and '<Your\_Azure\_Endpoint>' with your actual credentials.

import os

import requests

# Set up Azure OpenAI credentials

api\_key = "<Your\_Azure\_API\_Key>"

endpoint = "<Your\_Azure\_Endpoint>"

deployment\_name = "text-davinci-003" # Replace with your specific deployment name

# Create the full endpoint URL

url = f"{endpoint}/openai/deployments/{deployment\_name}/completions?api-version=2023-05-15"

# Set up headers

headers = {

"Content-Type": "application/json",

"api-key": api\_key

}

**Step 3: Define the Agent’s Functionality**

Next, define the function that sends a user input to Azure OpenAI and gets a response. We'll use a POST request with the appropriate payload.

def run\_agent(user\_input):

# Create the payload with the user's prompt

payload = {

"prompt": user\_input,

"max\_tokens": 100,

"temperature": 0.7

}

# Make the request to the Azure OpenAI API

response = requests.post(url, headers=headers, json=payload)

# Check for any errors

if response.status\_code == 200:

result = response.json()

return result['choices'][0]['text'].strip()

else:

return f"Error: {response.status\_code}, {response.text}"

# Test the agent with a sample input

response = run\_agent("What is the capital of France?")

print(response)

**Step 4: Running the Agent in Google Colab**

Now that the agent is set up, you can continuously ask questions, and the model will respond.

while True:

user\_input = input("Enter your query (or type 'exit' to stop): ")

if user\_input.lower() == 'exit':

break

print("Response:", run\_agent(user\_input))

**Explanation of Code:**

1. **requests Library**: Used to make HTTP POST requests to the Azure OpenAI API endpoint.
2. **API Key & Endpoint**: The API key and endpoint are used to authenticate with Azure OpenAI.
3. **Payload**: The user’s prompt and parameters (like max\_tokens and temperature) are sent in a JSON payload.
4. **Response Handling**: The response is parsed to extract the text generated by the model.

**Step 5: Example Queries**

You can now try asking different questions, for example:

* **Technical Query**: "How do I set up an Azure Virtual Machine?"
* **General Knowledge**: "Who is the president of the United States?"
* **Math Problem**: "What is the sum of 123 and 456?"

**Conclusion**

In this updated lab exercise, we bypassed the limitation of not finding the azure-ai-openai package by directly calling the Azure OpenAI API with the requests library. You successfully built an agent that responds to user queries using the Azure OpenAI service.