EX06: Automated Multi-AI Tool Integration Using Python

Aim

The aim of this experiment is to develop and implement Python code that integrates with multiple AI tools through APIs, automates querying these tools, compares their responses, and produces actionable insights. This experiment offers a practical understanding of interacting with external AI services, handling JSON data, and applying basic natural language processing (NLP) techniques for comparative analysis.

Software and Services Required

1. **Python (Version 3.8 or higher)**: The primary programming language used for coding and integration.
2. **Python IDE**: For writing and executing Python code (e.g., Jupyter Notebook, VS Code).
3. **Python Libraries**:
   - requests: To make HTTP requests to AI APIs.
   - openai: To access OpenAI API services.
   - difflib: A standard library module for comparing sequences (used for response similarity analysis).
4. **API Services**:
   - **OpenAI API**: Provides access to GPT models for text generation.
   - **Gemini API**: Hypothetical API for a different AI model (e.g., Google's Gemini).
5. **API Keys**: Obtain API keys by creating developer accounts for OpenAI and Gemini.

Key Concepts

1. **API Integration**: Establishing a connection with external AI tools via HTTP requests and handling JSON responses.
2. **Text Generation**: Using language models (e.g., ChatGPT, Gemini) to generate responses based on input prompts.
3. **Response Comparison**: Analyzing text responses using similarity measures to identify differences or similarities between outputs.
4. **Actionable Insights**: Deriving meaningful conclusions from comparing AI-generated responses.

Experiment Code

Python Code 1: (Using ChatGPT)

```
python
Copy code
import requests
import openai
from difflib import SequenceMatcher

# API Keys (Replace with your actual API keys)
CHATGPT_API_KEY = "your_chatgpt_api_key"

# Initialize OpenAI API for ChatGPT
openai.api_key = CHATGPT_API_KEY

# Function to query ChatGPT
def query_chatgpt(prompt):
    response = openai.Completion.create(
        model="text-davinci-003",
        prompt=prompt,
        max_tokens=150
    )
    return response.choices[0].text.strip()

# Function to compare responses using similarity ratio
def compare_responses(response1, response2):
    similarity = SequenceMatcher(None, response1, response2).ratio()
    return similarity

# Function to generate insights based on the comparison
def generate_insights(response1, response2, similarity):
    if similarity > 0.8:
        return "The responses from both AI tools are highly similar."
    elif len(response1) > len(response2):
        return "ChatGPT provided a more detailed response."
    else:
        return "The other tool provided a more concise response."

# Main Experiment Code
if __name__ == "__main__":
    prompt = "What is the impact of AI on healthcare?"
    chatgpt_response = query_chatgpt(prompt)

    # For demonstration purposes, assume a placeholder response from Gemini
    gemini_response = "AI has revolutionized healthcare by improving diagnostic accuracy,
streamlining patient management, and enhancing predictive analytics."

    # Display the responses
    print("ChatGPT Response:", chatgpt_response)
    print("Gemini Response:", gemini_response)

    # Calculate similarity and generate insights
    similarity = compare_responses(chatgpt_response, gemini_response)
    print(f"Similarity Ratio: {similarity:.2f}")
    insights = generate_insights(chatgpt_response, gemini_response, similarity)
    print("Insights:", insights)
```

Python Code 2: (Using Gemini)

python
Copy code

```
import requests
import openai
from difflib import SequenceMatcher

# API Keys (Replace with your actual API keys)
GEMINI_API_KEY = "your_gemini_api_key"

# Function to query Gemini (Hypothetical implementation)
def query_gemini(prompt):
    url = "https://api.google.com/gemini/v1" # Hypothetical API endpoint
    headers = {"Authorization": f"Bearer {GEMINI_API_KEY}"}
    payload = {"prompt": prompt}
    response = requests.post(url, headers=headers, json=payload)
    if response.status_code == 200:
        result = response.json()
        return result.get('response_text', "No response available")
    return "Error in Gemini API request."

# Function to compare responses using similarity ratio
def compare_responses(response1, response2):
    similarity = SequenceMatcher(None, response1, response2).ratio()
    return similarity

# Function to generate insights based on the comparison
def generate_insights(response1, response2, similarity):
    if similarity > 0.8:
        return "The responses from both AI tools are highly similar."
    elif len(response1) > len(response2):
        return "The Gemini tool provided a more detailed response."
    else:
        return "The other tool provided a more concise response."

# Main Experiment Code
if __name__ == "__main__":
    prompt = "What is the impact of AI on healthcare?"

    # For demonstration purposes, assume a placeholder response from ChatGPT
    chatgpt_response = "AI is transforming healthcare by enabling personalized treatment,
improving diagnostics, and optimizing hospital operations."

    gemini_response = query_gemini(prompt)
```

```
# Display the responses
print("ChatGPT Response:", chatgpt_response)
print("Gemini Response:", gemini_response)

# Calculate similarity and generate insights
similarity = compare_responses(chatgpt_response, gemini_response)
print(f"Similarity Ratio: {similarity:.2f}")
insights = generate_insights(chatgpt_response, gemini_response, similarity)
print("Insights:", insights)
```

Output (Example)

When running the above two code, the output might look like this:

vbnet
Copy code

```
ChatGPT Response: The impact of AI on healthcare has been transformative, enabling faster
diagnosis, personalized treatments, and improved patient care through advanced analytics and
machine learning models.
Gemini Response: AI has revolutionized healthcare by improving diagnostic accuracy,
streamlining patient management, and enhancing predictive analytics.
Similarity Ratio: 0.85
Insights: The responses from both AI tools are highly similar.
```

Conclusion

Both ChatGPT and Gemini AI models provide coherent and relevant responses to the prompt. The comparison shows a high degree of similarity, indicating that both models have a solid understanding of the topic. The experiment demonstrates that:

1. **API Integration**: Both APIs can be integrated effectively using Python.
2. **Response Analysis**: The similarity ratio helps understand how closely aligned the responses are.
3. **Actionable Insights**: Depending on the use case, one might prefer a more detailed or a more concise response.