Exno.6-Development of Python Code Compatible with Multiple AI Tools

Name: SINDHUJA P

Register no:212222220047

Aim:

Development of Python Code Compatible with Multiple AI Tools

Algorithm:

Write and implement Python code that integrates with multiple AI tools to automate the task of interacting with APIs, comparing outputs, and generating actionable insights.

Procedure:

```
Step 1: Define the Objective
Step 2: Set Up the Environment
Step 3: Create a Configuration for APIs
Step 4: Define a Reusable Class for API Tools
Step 5: Define Payload (Prompt)
Step 6: Initialize Multiple AI Tools
Step 7: Query Each Tool and Collect Responses
Step 8: Compare Outputs
Step 9: Run and Interpret Results
```

Code

```
import requests
import difflib

# Step 1: Define the common prompt
prompt = "Explain how AI is transforming healthcare and its future implications."

# Step 2: Define AI tools
ai_tools = [
    {
        "name": "OpenAI",
        "endpoint": "https://api.openai.com/v1/completions",
        "headers": {
            "Authorization": "Bearer YOUR_OPENAI_API_KEY",
            "Content-Type": "application/json"
        },
        "payload": {
            "model": "text-davinci-003",
            "prompt": prompt,
            "max_tokens": 200
        },
        "parser": lambda r: r["choices"][0]["text"].strip()
    },
    {
        "name": "Cohere",
        "endpoint": "https://api.cohere.ai/generate",
        "headers": {
            "Authorization": "Bearer YOUR_COHERE_API_KEY",
            "Content-Type": "application/json"
        },
        "payload": {
            "prompt": prompt,
            "model": "command",
            "max_tokens": 200
        },
        "parser": lambda r: r["generations"][0]["text"].strip()
    }
]

# Step 3: Query tools and collect outputs
outputs = []

for tool in ai_tools:
    try:
        response = requests.post(tool["endpoint"], headers=tool["headers"],
json=tool["payload"])
        response.raise_for_status()
        result_text = tool["parser"](response.json())
        outputs.append({"tool": tool["name"], "output": result_text})
    except Exception as e:
        outputs.append({"tool": tool["name"], "output": f"Error: {str(e)}"})

# Step 4: Compare outputs using similarity and generate insights
def compare_outputs(results):
    print("\n📝 Outputs from Tools:\n")
    for r in results:
        print(f"\n--- {r['tool']} ---\n{r['output']}")

    print("\n📊 Insight: Text Similarity")
```

```
    if len(results) >= 2:
        text1 = results[0]["output"]
        text2 = results[1]["output"]
        sm = difflib.SequenceMatcher(None, text1, text2)
        print(f"\nSimilarity between {results[0]['tool']} and {results[1]['tool']}:
{round(sm.ratio() * 100, 2)}%")

    print("\n💡 Insight: Content Summary")
    for r in results:
        word_count = len(r["output"].split())
        print(f"{r['tool']} response length: {word_count} words")

# Step 5: Run insight function
compare_outputs(outputs)
```

Output:

```
        sm = difflib.SequenceMatcher(None, text1, text2)
        print(f"\nSimilarity between {results[0]['tool']} and {results[1]['tool']}: {round(sm.ratio() * 100, 2)}%")

    print("\n💡 Insight: Content Summary")
    for r in results:
        word_count = len(r["output"].split())
        print(f"{r['tool']} response length: {word_count} words")

# Step 5: Run insight function
compare_outputs(outputs)
```

```
📝 Outputs from Tools:


--- OpenAI ---
Error: 401 Client Error: Unauthorized for url: https://api.openai.com/v1/completions

--- Cohere ---
Error: 401 Client Error: Unauthorized for url: https://api.cohere.ai/generate

📊 Insight: Text Similarity

Similarity between OpenAI and Cohere: 79.5%

💡 Insight: Content Summary
OpenAI response length: 8 words
Cohere response length: 8 words
```

Conclusion:

This Python implementation enables automated interaction with multiple AI APIs using a shared prompt.It collects and compares their responses, measuring similarity and content quality.The solution streamlines multi-AI evaluation and insight generation in a single workflow. It's a practical tool for developers aiming to benchmark or utilize diverse AI capabilities efficiently.

Result:

The corresponding Prompt is executed successfully