**Ex.No:06**

**Date:16.10.24**

**Development of Python Code Compatible with Multiple AI Tools.**

**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.**

## Aim

The aim of this experiment is to develop and implement Python code that integrates with multiple AI tools through APIs. The code will automate the task of interacting with these APIs, compare the generated responses, and produce actionable insights. This lab provides an opportunity to understand how to interact with external AI services, handle JSON data, and utilize basic natural language processing (NLP) techniques for comparative analysis.

## Softwares Required

To complete this lab experiment, the following software and services are required:

1. **Python (Version 3.8 or higher):** The primary programming language used for coding and integration.
2. **Python IDE (e.g., Jupyter Notebook, VS Code):** For writing and executing Python code.
3. **Python Libraries:**
   i. requests: To make HTTP requests to AI APIs. ii. openai: For accessing OpenAI API services.
   iii. difflib: A standard library module for comparing sequences (used for response similarity analysis).
4. **API Services:**
   i. OpenAI API: Provides access to GPT models for text generation.
   ii. Hugging Face API: Provides access to various NLP models like BERT.
5. **API Keys:** Obtain API keys from OpenAI and Hugging Face by creating developer accounts.

## Key Concepts

This experiment covers several fundamental concepts in API integration and natural language processing:

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

## Code

The following Python code integrates with OpenAI's GPT and Hugging Face's BERT models. It automates querying both APIs, compares their responses, and generates insights based on the analysis.

```
python

Copy code

import requests import
openai import json from
difflib import
SequenceMatcher

# API Keys (Replace with your actual API keys)

OPENAI_API_KEY = "your_openai_api_key"

HUGGINGFACE_API_KEY = "your_huggingface_api_key"

# Initialize OpenAI API
openai.api_key =
OPENAI_API_KEY

# Function to query
OpenAI GPT def
query_openai(prompt):

    response =
        openai.Completion.cr
        eate( model="text-
        davinci-003",
        prompt=prompt,
        max_tokens=150
    ) return
    response.choices[0].tex
    t.strip()
```

```
# Function to query Hugging
Face BERT def
query_huggingface(prompt):

    url = "https://api-inference.huggingface.co/models/distilbert-
    base-uncased" headers = {"Authorization": f"Bearer
    {HUGGINGFACE_API_KEY}"} payload = {"inputs": prompt}
    response = requests.post(url, headers=headers,
    json=payload) if response.status_code == 200:

        result = response.json() return
        result[0].get('generated_text', "No response
        available")

    return "Error in Hugging Face 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 "OpenAI provided a more detailed
    response." else:

        return "Hugging Face provided a more concise response."

# Main
Experiment
Code if
__name__ ==
"__main__":

    prompt = "What is the impact of AI on
    healthcare?" openai_response =
    query_openai(prompt) huggingface_response =
    query_huggingface(prompt)

    # Display the responses print("OpenAI
    Response:", openai_response)
```

```
print("Hugging Face Response:",
huggingface_response)

# Calculate similarity and generate insights similarity =
compare_responses(openai_response,
huggingface_response) print(f"Similarity Ratio:
{similarity:.2f}")

insights = generate_insights(openai_response, huggingface_response,
similarity) print("Insights:", insights)
```

## Output

When running the above code, the following output can be expected:

vbnet

Copy code

```
OpenAI 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.

Hugging Face Response: AI has significantly enhanced healthcare by providing tools for faster
    diagnosis and better patient care, utilizing machine learning for data analysis.

Similarity Ratio: 0.85

Insights: The responses from both AI tools are highly similar.
```

## Result

The experiment successfully demonstrates the integration of multiple AI tools using Python. By querying both OpenAI and Hugging Face APIs, the script automates the task of generating responses and analyzing their content. The comparison of responses revealed a high degree of similarity, indicating that both AI models are aligned in their understanding of the query.