EXP 6 : Development of Python Code Compatible with Multiple AI Tools

Experiment:

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:

To compare the responses of two open-source language models, **GPT-Neo** and **GPT-2**, to a given question, and analyze how different models generate text and handle natural language queries.

Procedure:

1. **Install Required Libraries**:

Use the command below to install the necessary Python libraries:
bash
Copy code
```
pip install transformers torch
```

2. **Load Models**:
    - Load two pre-trained language models from Hugging Face:
        - **GPT-Neo** (`EleutherAI/gpt-neo-1.3B`).
        - **GPT-2** (`gpt2`).
3. **Define Functions**:
    - Define two functions to generate text from both models.
        - **GPT-Neo Function**: Generates text from the GPT-Neo model.
        - **GPT-2 Function**: Generates text from the GPT-2 model.
4. **Generate Answers**:
    - Input the question “What are the benefits of renewable energy?” to both models and generate their responses.
5. **Compare Answers**:
    - Compare the generated answers from both models to see if they match or differ.
    - Print the responses and a summary indicating whether the answers are the same or different.
6. **Execute the Code**:
    - Run the code to generate and compare answers.

Deliverables:

1. **Python Script**: A script to compare answers from two models.

2. **Comparison Output**: The answers generated by both models and a summary of whether the answers are similar or different.

Sample Code:

```
from transformers import pipeline

# Load GPT-Neo and GPT-2 models
generator_neo = pipeline('text-generation', model='EleutherAI/gpt-neo-
1.3B')
generator_gpt2 = pipeline('text-generation', model='gpt2')

# Function to get answer from GPT-Neo
def get_gpt_neo_answer(question):
    generated_text = generator_neo(question, max_length=100,
num_return_sequences=1)
    return generated_text[0]['generated_text']

# Function to get answer from GPT-2
def get_gpt2_answer(question):
    generated_text = generator_gpt2(question, max_length=100,
num_return_sequences=1)
    return generated_text[0]['generated_text']

# Function to compare answers from both models
def compare_answers(question):
    answer_gpt_neo = get_gpt_neo_answer(question)
    answer_gpt2 = get_gpt2_answer(question)

    print("GPT-Neo Answer:", answer_gpt_neo)
    print("GPT-2 Answer:", answer_gpt2)

    if answer_gpt_neo == answer_gpt2:
        summary = "Both models provided the same answer."
    else:
        summary = "The answers are different."

    print("Summary:", summary)

    return {
```

```
        "question": question,
        "gpt_neo_answer": answer_gpt_neo,
        "gpt2_answer": answer_gpt2,
        "summary": summary
    }

# Run the comparison with a sample question
question = "ABOUT TODAY WEARTHER"
result = compare_answers(question)
print("Comparison Result:", result)
```

Result:

Sample Output:

GPT-Neo Answer: The weather today will be...
GPT-2 Answer: I can't predict the weather today...
Summary: The answers are different.
Comparison Result: {'question': 'ABOUT TODAY WEATHER', 'gpt_neo_answer': 'The weather today will be...', 'gpt2_answer': 'I can't predict the weather today...', 'summary': 'The answers are different.'}

Summary & Conclusion:

In comparing GPT-Neo and GPT-2, responses differ due to their distinct architectures and training sets. GPT-Neo typically offers more context-rich outputs, while GPT-2 tends to be simpler and more concise. Their generative nature can lead to varied results, making the choice dependent on whether depth, creativity, or efficiency is prioritized. Fine-tuning may help achieve consistency for specific applications.