React Pattern Report

How the LLM Was Used for Reasoning

Reasoning with a Large Language Model (LLM)

The **LLM (Gemini)** was used as the core reasoning engine to generate meaningful and diverse research questions based on a given topic. Here's how the reasoning step works:

1. **Input Understanding**:
   The LLM receives a human-readable instruction prompt like:
   "Generate 5 to 6 well-structured research questions covering different aspects of the topic: 'Artificial Intelligence in Healthcare'."
2. **Knowledge Retrieval & Structure Generation**:
   The LLM uses its pretrained knowledge to identify:
   - Key subdomains within the topic (e.g., clinical efficacy, ethics, regulations).
   - Relevant question formats for academic or research contexts.
3. **Output Generation**:
   It generates diverse and logically coherent questions by reasoning through:
   - What areas researchers might investigate.
   - How those areas relate to current debates, challenges, and opportunities.

This allows the LLM to produce content that isn't just fact-recall, but **logical, relevant, and structurally sound reasoning output**.

---

ii. Code Explanation and Flow

Key Components

1. Library Setup

```
import google.generativeai as genai
from tavily import TavilyClient
```

These libraries are used to connect with:

- `Gemini` for LLM-based content generation.
- `Tavily` for real-time web search results.

2. API Configuration

```
genai.configure(api_key=GEMINI_API_KEY)
tavily = TavilyClient(api_key=TAVILY_API_KEY)
```

APIs are authenticated using your keys to allow requests to both services.

---

3. The `ResearchAgent` Class

This class wraps the entire logic for topic-based research.

a. `__init__()`

```
def __init__(self, topic):
    self.topic = topic
    self.questions = []
    self.answers = []
```

Stores the topic and prepares lists for questions and answers.

b. `generate_questions()`

```
model = genai.GenerativeModel(model_name="gemini-2.5-flash")
response = model.generate_content(prompt)
```

- A prompt is crafted dynamically using the `topic`.
- Gemini LLM generates structured questions.
- Questions are cleaned and stored.

c. `search_answers()`

```
results = tavily.search(query=question, max_results=3)
```

- For each generated question, Tavily performs a web search.
- The top 3 content results are formatted and stored as answers.

d. `generate_report()`

```
report += f"### ❓ Question {idx}: {question}\n{answer}\n\n"
```

- Combines all the questions and answers into a structured markdown report.

---

4. Execution Flow

```
agent = ResearchAgent("Artificial Intelligence in Healthcare")
agent.generate_questions()
agent.search_answers()
final_report = agent.generate_report()
```

- Initializes the agent with a topic.
- Generates questions.
- Finds answers using web search.
- Compiles a full report.

---

5. Saving the Report

```
with open ("AI_Research_Report.txt", "w") as f:
    f.write(final_report)

from google.colab import files
files.download('AI_Research_Report.txt')
```

The report is saved as a text file in the local and Colab environment.