Experiment Title

Developing a Prompt-Based Application Using ChatGPT to Organize Daily Tasks

---

Objective

To build a simple to advanced prompt-driven application leveraging ChatGPT's natural language processing capabilities, aimed at organizing daily tasks effectively. The experiment will show how prompt design influences the quality and utility of responses, progressing from basic to more sophisticated prompts.

Tools & Technologies

- Python 3
- OpenAI ChatGPT API (or OpenAI Python SDK)
- Basic knowledge of prompt engineering
- `requests` or `openai` Python library for API calls (assuming real API integration)

Methodology

Step 1: Define the Task

Create a text-based prompt interface where the user provides task information in natural language, and ChatGPT returns a structured task list or helpful suggestions to organize the day.

Step 2: Design Prompt Variations

The experiment focuses on how varying prompt complexity improves output. Three prompt designs will be demonstrated:

- **Basic Prompt:** Simple request to list tasks.
- **Intermediate Prompt:** Request tasks with priority and timing details.
- **Advanced Prompt:** Request detailed daily plan including priority, time slots, and motivational notes.

Step 3: Implement Prompt-API Calls (Simulated)

For demonstration, API calls are mocked or represented by expected responses.

Step 4: Compare Outputs

Analyze how each prompt affects the clarity, usefulness, and detail of ChatGPT’s response.

Python Code Example (Simulated)

```
# Simulated ChatGPT responses for demonstration purposes

def chatgpt_basic(prompt):
    # Basic prompt response
    return "Tasks for today:\n1. Grocery shopping\n2. Finish assignment\n3.
Call mom"

def chatgpt_intermediate(prompt):
    # Intermediate prompt response with priority and timing
    return ("Tasks with priority and time:\n"
            "1. Finish assignment (High priority, 9 AM - 11 AM)\n"
            "2. Grocery shopping (Medium priority, 1 PM - 2 PM)\n"
            "3. Call mom (Low priority, 7 PM)")

def chatgpt_advanced(prompt):
    # Advanced prompt response with detailed plan and motivational note
    return ("Today's Schedule:\n"
            "9:00 AM - 11:00 AM: Finish assignment [High Priority]\n"
            "1:00 PM - 2:00 PM: Grocery shopping [Medium Priority]\n"
            "7:00 PM: Call mom [Low Priority]\n\n"
            "Remember: Prioritize important tasks early and take short
breaks to stay productive!")

def run_task_organizer():
    basic_prompt = "List my tasks for today."
    intermediate_prompt = ("List my tasks with priority and suggested time
slots for today.")
    advanced_prompt = ("Create a detailed daily schedule with priorities,
time blocks, and motivational tips.")

    print("=== Basic Prompt ===")
    print(chatgpt_basic(basic_prompt))
    print("\n=== Intermediate Prompt ===")
    print(chatgpt_intermediate(intermediate_prompt))
    print("\n=== Advanced Prompt ===")
    print(chatgpt_advanced(advanced_prompt))

if __name__ == "__main__":
    run_task_organizer()
```

Sample Outputs

1. Basic Prompt Output

```
Tasks for today:
1. Grocery shopping
2. Finish assignment
3. Call mom
```

2. Intermediate Prompt Output

```
Tasks with priority and time:
1. Finish assignment (High priority, 9 AM - 11 AM)
2. Grocery shopping (Medium priority, 1 PM - 2 PM)
3. Call mom (Low priority, 7 PM)
```

3. Advanced Prompt Output

```
Today's Schedule:
9:00 AM - 11:00 AM: Finish assignment [High Priority]
1:00 PM - 2:00 PM: Grocery shopping [Medium Priority]
7:00 PM: Call mom [Low Priority]

Remember: Prioritize important tasks early and take short breaks to stay
productive!
```

Analysis

- **Basic Prompt:** Provides a simple unordered list; useful for quick task recall but lacks detail.
- **Intermediate Prompt:** Adds structure with priorities and suggested time frames, helping with time management.
- **Advanced Prompt:** Delivers a comprehensive daily schedule, blending task info with motivational advice, thereby supporting productivity and emotional engagement.

The increasing sophistication of the prompts leads to more actionable and personalized responses.

Conclusion

This experiment demonstrated how ChatGPT can be employed as the core engine behind a prompt-based daily task organizer. Starting with simple commands, the progression to more complex prompt designs allows the AI to produce increasingly helpful outputs, from basic lists to full schedules with motivational guidance.

Prompt engineering is crucial in maximizing the effectiveness of AI-driven applications, especially in personal productivity domains.