CSAI 422:Laboratory Assignment 1

Exploring Temperature Parameter in Large Language Model Responses

Objectives

By the end of this lab, students will: 1. Understand how to set up and use the Groq API via OpenAI interface 2. Learn to make API calls using the OpenAI format 3. Analyze how temperature settings affect model outputs 4. Practice working with API responses and data collection 5. Develop analytical skills in evaluating language model behavior

Prerequisites

- Python 3.10 or higher
- Basic understanding of API calls
- Familiarity with JSON formatting
- Understanding of virtual environments

Part 1: Setup (20 minutes)

1. Visit the Groq website (https://groq.com)
2. Create an account and generate an API key
3. Set up your Python virtual environment:

```
# Create a new virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows:
.\venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

4. Install the required Python package:

```
pip install openai
```

5. Create a new Python file and set up your environment:

```
import os
import openai
import json
from datetime import datetime

# Initialize the client with your API key
client = openai.OpenAI(
    base_url="https://api.groq.com/openai/v1",
```

```
    api_key=os.environ.get("GROQ_API_KEY")
)

# Alternative setup if not using environment variables:
# client = openai.OpenAI(
#     base_url="https://api.groq.com/openai/v1",
#     api_key="your-api-key-here"
# )
```

Part 2: Creating the Test Framework (20 minutes)

Create a function to test different temperatures with the same prompt:

```
def generate_responses(prompt, temperature, num_responses=5):
    responses = []

    for i in range(num_responses):
        response = client.chat.completions.create(
            model="gemma-7b-it",
            messages=[
                {
                    "role": "user",
                    "content": prompt
                }
            ],
            temperature=temperature
        )

        responses.append(response.choices[0].message.content)

    return responses

def save_results(prompt, temperature, responses):
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"results_{temperature}_{timestamp}.json"

    data = {
        "prompt": prompt,
        "temperature": temperature,
        "responses": responses
    }

    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
```

Part 3: Experiment Design (45 minutes)

Test the following prompts at different temperature settings (0.0, 0.3, 0.7, and 1.0):

Example Prompts:

1. "Describe a world where apples are used as currency. What would the economy look like?"
2. "Write a short story about a golden apple that grants infinite wealth to its owner."
3. "Create a business plan for a luxury apple orchard that caters to billionaires."

For each temperature setting: 1. Generate 5 responses for each prompt 2. Save the results using the provided save_results() function 3. Document your observations about how the responses differ

Example code for running the experiment:

```
temperatures = [0.0, 0.3, 0.7, 1.0]
prompts = [
    "Describe a world where apples are used as currency. What would the economy look like?",
    "Write a short story about a golden apple that grants infinite wealth to its owner.",
    "Create a business plan for a luxury apple orchard that caters to billionaires."
]

for temp in temperatures:
    for prompt in prompts:
        responses = generate_responses(prompt, temp)
        save_results(prompt, temp, responses)
```

Part 4: Analysis and Report (60 minutes)

Write a detailed report (1000-1500 words) that includes:

1. Introduction
   - Brief overview of the experiment
   - Explanation of the temperature parameter and its theoretical impact
2. Methodology
   - Description of the prompts used and why they were chosen
   - Documentation of the temperature values tested
   - Explanation of the data collection process
3. Results
   - Comparison of responses across different temperatures
   - Analysis of:
     - Response variety
     - Creativity level
     - Coherence

    - Adherence to the prompt
  - Include specific examples from your generated responses
4. Discussion
  - Patterns observed at different temperature settings
  - Impact of temperature on:
    - Response predictability
    - Creative elements
    - Response length
    - Consistency across multiple generations
  - Potential applications for different temperature settings
5. Conclusion
  - Summary of key findings
  - Recommendations for temperature settings based on use case
  - Suggestions for future experiments

Submission Requirements

Submit: 1. Your complete Python code 2. All generated JSON files containing responses 3. Your analysis report in PDF format 4. A brief reflection (250 words) on what you learned about language model behavior

Grading Criteria

- Setup and Code Implementation (20%)
- Experiment Execution (20%)
- Data Collection and Organization (20%)
- Analysis Quality (25%)
- Report Writing and Presentation (15%)

Tips for Success

- Pay attention to patterns in response variety at different temperatures
- Take detailed notes during the experiment
- Use specific examples to support your observations
- Consider creating visualizations to support your analysis
- Document any unexpected behaviors or interesting findings
- Remember to deactivate your virtual environment when done:

  ```
  deactivate
  ```

Extra Credit Opportunities (10 points each)

1. Implement visualization of response similarities across temperatures
2. Analyze response length distributions at different temperatures
3. Create a user-friendly interface for running the experiments

Due Date

One week from lab date

Notes

- Make sure to remove your API key before submitting your code
- Keep your generated responses for potential class discussion
- Document any technical issues encountered and how they were resolved
- If using environment variables, remember to set your GROQ_API_KEY:

```
# On Windows:
set GROQ_API_KEY=your-api-key-here
# On macOS/Linux:
export GROQ_API_KEY=your-api-key-here
```