CS435 - Class Assignment and Demonstration - AI Judge API

March 14, 2025

Assignment Description

Objective: You will create an **AI Judge** that can detect if a user-submitted question or statement could potentially harm a brand's reputation.. **Be ready to show this in the next class.**

Context:

- Many companies seek to protect their brand image by preventing harmful, defamatory, or highly controversial content from being associated with them.
- Your task is to write a program that analyzes an input question (or statement) to determine whether it might harm a brand's reputation.
- The analysis should flag any potential brand risk, such as:
  - Controversial or sensitive topics
  - Misinformation
  - Non-compliance with regulations
  - Possible defamation, offensive, or hateful content

LLM Options for AI Moderation:

- **Gemini**
  `https://gemini.google.com/app`
- **Hugging Face Llama 3** (this is gated so make sure to apply ahead of time - it takes 24 hours for approval)
  https://huggingface.co/meta-llama/Llama-3.1-8B
  `https://huggingface.co/settings/tokens`
- **OpenAI**
  `https://platform.openai.com`

You may select **any one** of these APIs and **generate a key** to run your brand risk analysis. You can use all if you like. If you choose another provider than these 3, ensure you explain your decision and provide necessary credentials.

In the next lecture, you will showcase your code and we will be working with Nemo Guardrails. **guardrails and safe prompts**, you can read up on it here:

- `https://github.com/NVIDIA/NeMo-Guardrails`

Requirements

1. **Select one of the provided APIs (Gemini, Llama 3, or OpenAI) and integrate it into your program.**
2. **Implement a function that takes a question (string) as input and queries the chosen API.**
3. **Analyze the API response to determine if the question is potentially damaging to a brand.**
   - Use any relevant classification or confidence scores provided by the API.
   - If no explicit classification is provided, devise your own threshold for brand risk.
4. **Output a clear, human-readable result:** either ``` ``Potentially damaging to brand'' ``` or ``` ``Safe for the brand'' ```.
5. **Handle errors gracefully** (e.g., invalid API key, network issues) and return an appropriate message or fallback.

Implementation Details and Suggestions

1. API Setup

- Create or obtain an API key for your chosen LLM provider.
- Locate the `API endpoint` and `authorization` method (likely "Bearer" tokens or keys in headers).
- If needed, store the API key securely (do not commit it directly to a public repository).

2. Basic Pseudocode

```
def check_brand_risk(question):
    # 1. Define API endpoint and headers
    api_url = "YOUR_CHOSEN_API_ENDPOINT"
    api_key = "YOUR_API_KEY"

    # 2. Prepare request payload
    payload = {"input": question}
    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    # 3. Make request
    response = requests.post(api_url, headers=headers, json=payload)

    # 4. Parse response for brand risk
    result = response.json()

    # 5. Return "Potentially damaging" or "Safe"
    if result.get("flagged"):
        return "Potentially damaging to brand."
    else:
        return "Safe for the brand."
```

3. Handling Errors and Edge Cases

- Handle **network timeouts**, invalid JSON responses, or `HTTPError` exceptions.
- If the API response is unclear, or a certain part of the question triggers suspicion, consider returning "Potentially damaging to brand" to err on the safe side.

4. Example Code Snippets

Use these as references for your final implementation.

OpenAI Example

```
import openai

openai.api_key = "YOUR_OPENAI_KEY"

response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": "Say this is a test"}],
    max_tokens=6,
    temperature=0
```

```
)

print(response.choices[0].message.content)
```

Generic Example

```
import requests

def check_brand_risk(question):
    api_url = "YOUR_API_ENDPOINT"
    api_key = "YOUR_API_KEY"
    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }
    payload = {"input": question}

    try:
        response = requests.post(api_url, headers=headers, json=payload)
        response.raise_for_status()
        result = response.json()

        if result.get("flagged"):
            return "Potentially damaging to brand."
        else:
            return "Safe for the brand."

    except Exception as e:
        return f"Error during API call: {str(e)}"
```

Deliverables

- **Program in Python:**
  - Takes an input question from the user.
  - Returns either ‘‘`Potentially damaging to brand`’’ or ‘‘`Safe for the brand`’’.
  - Run **10 distinct examples** through your program and include the results in your demonstration.
- **Submission:**
  - Be prepared to **demonstrate your program in the next class**.
    - Your source code.
    - Screenshots or console logs showing the program running with 10 example inputs.

- **Extra Credit:**
  - Implement your solution using `fastapi`.
  - Provide a simple `POST` endpoint (e.g., `/check_brand_risk`) that accepts a JSON body with the question.
  - Return the result as JSON (``damaging'' or ``safe'').

Note: If you have any questions or need clarification regarding the APIs or the requirements, please bring them up in class or post in the course discussion forum.