Chatbot with Google Generative AI and Sentiment Analysis: Documentation

by: Deepesh N , B.Tech CSE(3rd year) 15-sep-2024

PROJECT OVERVIEW

Task: Build a Negotiation Chatbot Using a Pre-trained AI Model

Objective: Create a chatbot that simulates a negotiation process between a customer and a supplier, leveraging pre-trained models like Gemini, ChatGPT, or other AI language models.

This project implements a negotiation chatbot using Google Generative AI (Gemini) for product price negotiation. It simulates a conversation between a chatbot (representing a seller) and a user (acting as the buyer). The chatbot offers dynamic responses based on the user's sentiment and logs API interactions for further analysis.

Key Components:

- **Google Generative AI for Chat Responses:**
  The chatbot uses Google's Generative AI (`**google-generativeai**` **SDK**) to provide dynamic and context-aware responses to the user's input.

- **Sentiment Analysis:**
  The sentiment of the user's input is analyzed using `**TextBlob**`. Depending on whether the sentiment is positive, negative, or neutral, the chatbot adapts its responses, offering better deals to polite users

- **API Endpoint Logging**
  Every API interaction is logged to track endpoints and facilitate debugging or analysis of API usage.

1. Installing Required Libraries - The following Python libraries are used in this project:

google-generativeai : To interact with the Google Generative AI model.

textblob: For sentiment analysis to determine the user's tone (positive, negative, or neutral).

dotenv: To securely load the API key from a .env file.

logging: To log API interactions and track errors.

2. API Key Setup:

To securely store and access your API key, ".env" file is used. The file should contain the following entry:

GEMINI_API_KEY=your_api_key_here

this file is stored securely and not included in public repositories.

3. Model Configuration:

Google Generative AI model is initialized using the "**google-generativeai**" Library. The model is set up with specific generation parameters such as temperature, top-p, top-k, and the maximum number of tokens in the response:

```python
genai.configure(api_key=os.getenv('GEMINI_API_KEY'))
generation_config = {
   'temperature': 1,
   'top_p': 0.95,
   'top_k': 64,
   'max_output_tokens': 8192,
   'response_mime_type': 'text/plain',
}

model = genai.GenerativeModel(
   model_name='gemini-1.5-flash',
   generation_config=generation_config,
   system_instruction=(
      'You are a chatbot that simulates a negotiation process for a product price. '
      'The user can accept, reject, or propose a counteroffer, and your task is to simulate this
negotiation.'
   ),
)
```

4. Sentiment Analysis:

To adjust the chatbot's responses based on the user's tone, sentiment analysis is performed using the **`TextBlob`** library. The polarity of the sentiment ranges from -1 (most negative) to 1 (most positive). Based on this sentiment score, the chatbot adapts its negotiation strategy:

```python
def analyze_sentiment(user_input):
    analysis = TextBlob(user_input)
    return analysis.sentiment.polarity  # Returns polarity (-1 to 1)
```

5. Chat Flow

The chatbot starts by offering a product and waiting for the user's input. It processes user input in a loop, adjusts the response based on sentiment, and sends the processed user input to the Generative AI model for dynamic responses.

```python
while True:
    user_input = input('You: ').strip()
    if user_input.lower() == 'exit':
        print('Bot: Thank you for chatting! Goodbye!')
        break
    if not user_input:
        print('Bot: I didn't catch that. Could you say that again?')
        continue
    sentiment_score = analyze_sentiment(user_input)
    if sentiment_score > 0.1:
        print('Bot: Let me offer you a special deal.')
    elif sentiment_score < -0.1:
        print('Bot: $1,200 is our best offer.')
    else:
        response = model.start_chat(history=history).send_message(user_input)
        print(f'Bot: {response.text}')
```

6. Logging API Interactions:

Every time an API call is made, the relevant endpoint and the request and response details are logged using Python's **`logging` module**. This is important for debugging, monitoring, and reviewing interactions.

```python
logging.basicConfig(
    filename='chatbot_api_log.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

def log_api_interaction(endpoint, request, response):
    logging.info(f'API Endpoint: {endpoint}')
    logging.info(f'Request Data: {request}')
    logging.info(f'Response Data: {response}')
```

7. Example Conversation

A sample conversation would look like this:

```
Bot: Hello! Welcome to our negotiation bot. I would like to show you our latest product: a high-
performance laptop...
You: That seems expensive.
Bot: I understand you might be upset, but $1,200 is our best offer at the moment.
```

Conclusion

This chatbot demonstrates how Google Generative AI can be combined with sentiment analysis to create a dynamic, interactive negotiation process. By integrating sentiment analysis and API endpoint logging, the chatbot can provide personalized responses and track interactions, making it easier to analyze and improve its performance.