Sentiment Analysis API with LLM Integration

Author: Gaurav Rode Date: September 2024

1. Introduction

This document outlines the development of a Flask-based Sentiment Analysis API. The API processes customer reviews from

CSV or XLSX files and returns sentiment scores (positive, negative, neutral) for the reviews using basic sentiment analysis. The API can

be extended to use LLM integration for more sophisticated analysis.

2. Approach

- API Framework: Flask was used for building the API.
- File Handling: The API accepts files in CSV or XLSX format containing customer reviews.
- Sentiment Analysis: A basic keyword-based sentiment analysis was used to classify reviews into positive, negative, or neutral categories.
- JSON Response: The API returns the analysis results in a structured JSON format.

3. Implementation Details

- Route: `/analyze` is the main route for processing the file upload and sentiment analysis.
- Sentiment Analysis Logic: A mock function was used to classify reviews based on keywords (e.g., 'great', 'bad'). Future integration

  with an LLM can enhance this functionality.
- Error Handling: The API handles file validation errors, such as missing or incorrect file types.

4. Sample Input and Output

Input: The input is an XLSX or CSV file containing customer reviews in a column labeled 'Review'.

Output: JSON response containing sentiment scores:


```
{
  "positive": 5,
  "negative": 2,
  "neutral": 3
}
```


5. Challenges and Insights

During development, handling different file formats and ensuring the accuracy of mock sentiment analysis posed challenges.

Real LLM integration will significantly improve sentiment classification, especially for nuanced reviews.

6. Limitations and Future Improvements

The current implementation relies on basic keyword matching for sentiment analysis, which has limited accuracy.

Incorporating a Large Language Model (LLM) such as the Groq API will provide more accurate and context-aware analysis.

7. Conclusion

The Flask-based sentiment analysis API was successfully implemented to process customer reviews. While the current

implementation uses a simple keyword-based approach, future integration with an LLM will greatly enhance its capabilities and real-world applications.

8. Code

```python
# Task : Sentiment Analysis API with LLM Integration

from flask import Flask, request, jsonify
import pandas as pd

app = Flask(__name__)

# Function to process the uploaded file and extract reviews
def process_file(file):
    df = pd.read_excel(file)

    # Extracting the 'Review' column
    if 'Review' in df.columns:
        reviews = df['Review'].tolist()
    else:
        return None, "No 'Review' column found in the file."

    return reviews, None

# analysis
def perform_sentiment_analysis(reviews):
    positive, negative, neutral = 0, 0, 0


    for review in reviews:
        if any(word in review.lower() for word in ["great", "fantastic", "exceeded"]):
            positive += 1
        elif any(word in review.lower() for word in ["poor", "terrible", "bad"]):
            negative += 1
        else:
            neutral += 1

    return {
        "positive": positive,
        "negative": negative,
        "neutral": neutral
    }

# for handle file upload and sentiment analysis
@app.route('/analyze', methods=['POST'])
def analyze():
    if 'file' not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files['file']
    reviews, error = process_file(file)

    if error:
        return jsonify({"error": error}), 400

    # Perform sentiment analysis on the extracted reviews
    sentiment_results = perform_sentiment_analysis(reviews)
```

```
    return jsonify(sentiment_results)

if __name__ == '__main__':
    app.run(debug=True)
```

9. Result / Output

Post Man