Documentation

Sentiment Analysis API with LLM Integration

Approach:

I developed a python-FastAPI based API that processes customer reviews from a CSV or XLSX file, performs sentiment analysis using a Large Language Model (LLM), and returns a structured JSON response with sentiment scores.

To achieve this:

- I used Python's FastAPI to implement this API as it is simple and fast for creating these kinds of API's.
- This application can extract reviews from the csv/xlsx file using the pandas module and then performs sentiment analysis using Groq API.
- Customer reviews are first divided into batches of 10 to optimize API calls to the Groq model and prevent exceeding token limits.
- The API calls the Groq API, which uses the LLaMA-3 model, to generate sentiment analysis scores for each batch. The responses are then aggregated, and scores for all reviews are combined.
- The API returns a well-structured JSON response containing the sentiment scores for each review.

Structured Response Implementation:

The Groq API is set up with specific instructions using a "system role" that clearly defines the response format. The response is required to follow a strict JSON schema, ensuring that each review's sentiment analysis is returned in a structured format. The JSON structure looks like:

Format:

```
{
   "scores": [
      {
         "review": "review_text1",
         "score": {
            "positive": "number (0-100)",
            "negative": "number (0-100)",
            "neutral": "number (0-100)"
         }
      },
   ]
```

}

This ensures that each review's sentiment is captured under the `scores` array with a corresponding sentiment breakdown (`positive`, `negative`, and `neutral`).

Examples of API Usage:

Let's consider that a CSV file "Reviews.csv" containing customer reviews (with a "Review" column) is uploaded to the `/uploadfile/` endpoint of the api. The contents of the Reviews.csv is shown below

Reviews.csv:

Review
"The product is excellent and arrived early."
"I had a terrible experience with customer support."

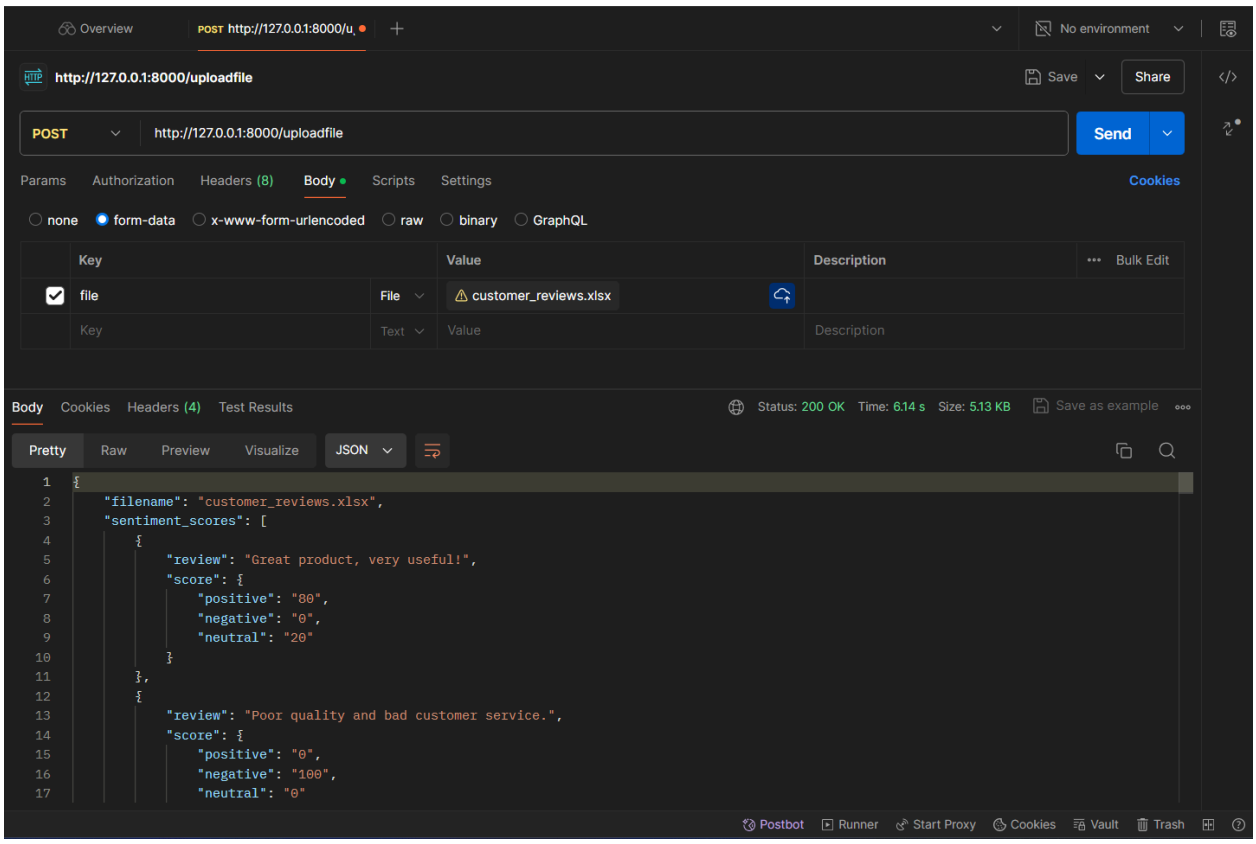
The json response for this would be as shown:

JSON Response:


```
{
   "filename": "customer_reviews.csv",
   "sentiment_scores": [
      {
         "review": "The product is excellent and arrived early.",
         "score": {
            "positive": 85,
            "negative": 5,
            "neutral": 10
         }
      },
      {
         "review": "I had a terrible experience with customer support.",
         "score": {
            "positive": 10,
            "negative": 85,
            "neutral": 5
         }
      }
   ]
}
```

Customer_reviews.xlsx:

You can check the response of the api in postman or you can just go to the `/docs/` api endpoint after running the python file to try out this api. Here is the response for the customer_reviews.xlsx file given for testing:



Analysis of results:

- **Performance:** The sentiment analysis was able to capture nuanced sentiments from customer reviews, providing balanced scores across positive, negative, and neutral emotions.
- **Consistency:** The JSON response schema remained consistent across various inputs, making it easy to parse and handle programmatically.
- **Error Handling:** I added robust exception handling for file upload issues, JSON parsing errors, and API-related problems, ensuring that the system gracefully handles errors like invalid formats or missing columns in the file.

Limitations and Potential Improvements:

- **Batch Processing Limitations:** The current implementation processes reviews in batches but may face token size issues with the Groq API if too many reviews are

concatenated. Future work could dynamically adjust batch size based on token limits to avoid overflow.

- **Processing Speed:** Since each review is processed through an LLM model, the API's performance can be affected by the speed of the model, especially when handling large volumes of reviews. Optimizations like asynchronous requests or parallel processing could improve response time.
- **Sentiment Nuance:** While the sentiment analysis is effective, more fine-tuned models could be used to provide more accurate or domain-specific sentiment scores, particularly for industry-specific reviews.

Additional Insights and Observations:

LLM Accuracy: The LLaMA-3 model worked well for general sentiment analysis but could sometimes miss specific contextual nuances, such as sarcasm or cultural variations in review text. More advanced or fine-tuned models could enhance the quality of sentiment scores.

Ease of Integration: The Groq API integration was smooth, thanks to its user-friendly design. The structured format made it easier to work with the API and extract sentiment analysis results effectively.

Model Customization: The use of the "system role" in the Groq API allowed us to precisely control the format of the responses, ensuring valid JSON responses without any extra text, which is essential for production environments.