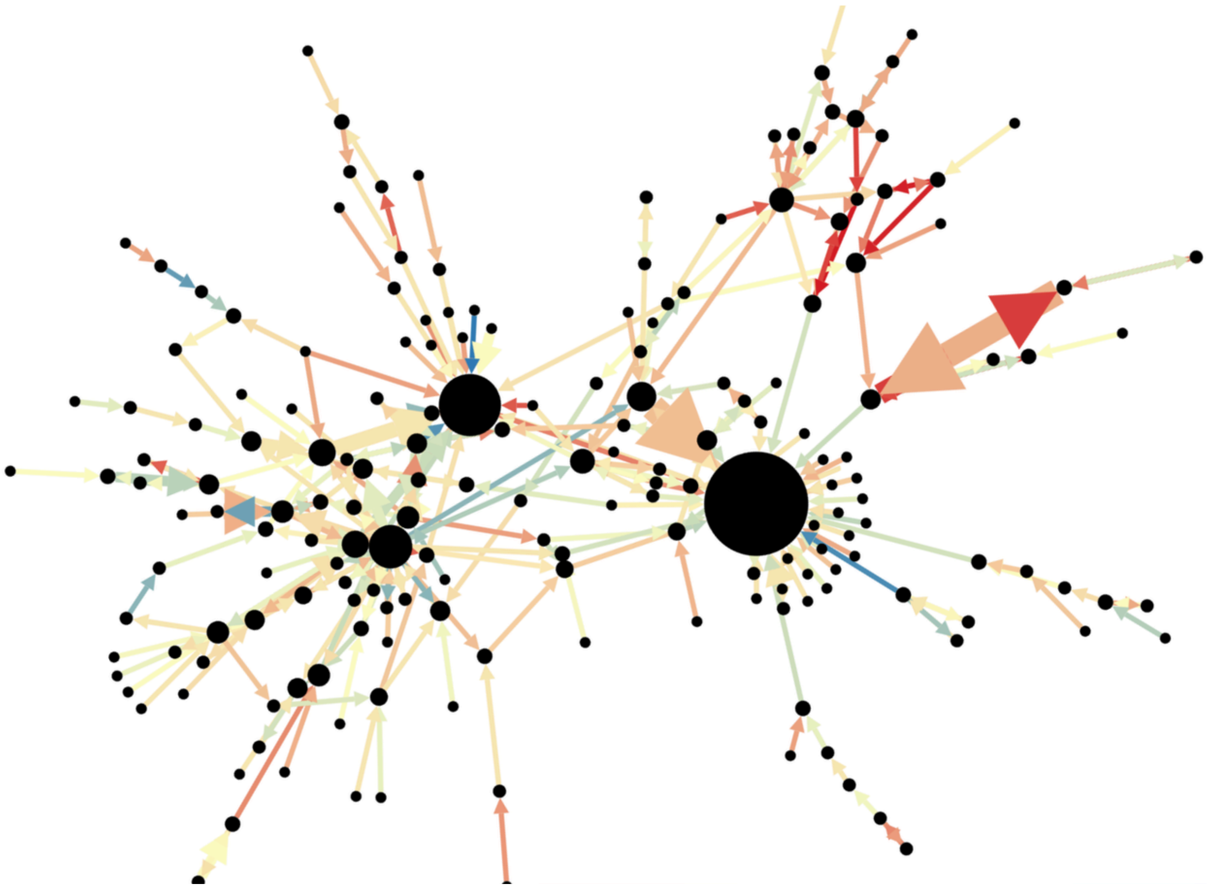
Sentiment Analysis Groq API

Assignment Report



Anshu Kumar Singh

Mob: 9896843511
anshukumar9728@gmail.com

Overview

Develop a Python-based API that processes customer reviews, performs sentiment analysis using a Large Language Model, and returns structured results. For this assignment task below is the following approach is used

The approach Involved:

1. Setting the Fastapi Application for handle the request and create the api Endpoints
2. Using the pandas for data manipulation to extract review text from the upload file
3. Now For LLM Integration Used the Groq API For performing the sentiment analysis on the reviews
4. For Error Handling we used the try except block of python and fastapi for request related problems

Directory Structure

```
/Project
 ├── app
 │   ├── route        # All the routing related task
 │   │   └── __init__.py
 │   │   └── sentiment_analysis.py
 │   ├── service      # Handle file validation and etc
 │   │   └── __init__.py
 │   │   └── sentiment_service.py
 │   └── utils        # In which module all the deep analysis like generate response
 │      └── __init__.py
 │   │   └── sentiment_utils.py
 ├── requirements.txt
 ├── .env
 ├── Dockerfile
 └── main.py
```

API USAGE

Once the application is running, you can access the API at:

1. /analyze - POST

This endpoint accepts a CSV or XLSX file containing customer reviews and processes them to perform sentiment analysis.

- **Method:** POST
- **Description:** Accepts an uploaded file (CSV or XLSX) and returns a sentiment analysis score.
- **Request Body:**
    - The request body should include a file with customer reviews. The file should have a column labeled Review containing the review text.

Solution Overview

- **Data Input:** Accept Excel files containing customer reviews.
- **Sentiment Analysis:** Utilize a pre-trained model to analyze the sentiments of reviews.
- **Response Format:** Provide structured responses detailing each review's sentiment.

Implementation Steps

1. **File Upload Handling:** Use `multipart/form-data` to handle file uploads.
2. **Data Processing:** Read the Excel file and extract customer reviews.
3. **Sentiment Analysis:** Process the reviews using a sentiment analysis model.
4. **Response Structuring:** Format the output as a JSON array with sentiment scores for each review.

2. Structured Response Implementation

Response Structure

The API returns a JSON array containing objects for each review with the following fields:

- `review`: The original customer review text.
- `sentiment`: A JSON string detailing the sentiment analysis, including:
  - `positive`: Score indicating positive sentiment.
  - `negative`: Score indicating negative sentiment.
  - `neutral`: Score indicating neutral sentiment.
  - `confidence_score`: Confidence level of the sentiment analysis.

Example Response

json

Copy code

```
[
  {
    "review": "Great product, very useful!",
    "sentiment": "{ \"sentiment_analysis\": { \"sentiment\": {
\"positive\": 0.8, \"negative\": 0.1, \"neutral\": 0.1 },
\"confidence_score\": 0.9 } }"
  }
]
```

Sample Inputs/Outputs

Input:

- Excel file (`customer_reviews.xlsx`) containing reviews like:

    - "Fantastic product!"
    - "Worst purchase ever."

Output:

json

Copy code

```
[
  {
    "review": "Fantastic product!",
    "sentiment": "{ \"sentiment_analysis\": { \"sentiment\": {
\"positive\": 0.9, \"negative\": 0.05, \"neutral\": 0.05 },
\"confidence_score\": 0.95 } }"
  },
  {
    "review": "Worst purchase ever.",
    "sentiment": "{ \"sentiment_analysis\": { \"sentiment\": {
\"positive\": 0.1, \"negative\": 0.9, \"neutral\": 0.0 },
\"confidence_score\": 0.92 } }"
  }
]
```

Limitations

- **Model Accuracy:** Results depend heavily on the quality of the sentiment analysis model.
- **Review Context:** The model may struggle with sarcasm or context-specific phrases.
- **File Format:** Currently limited to `.xlsx, .csv` files.