Sentiment Analysis Service

Ahsan Nisar

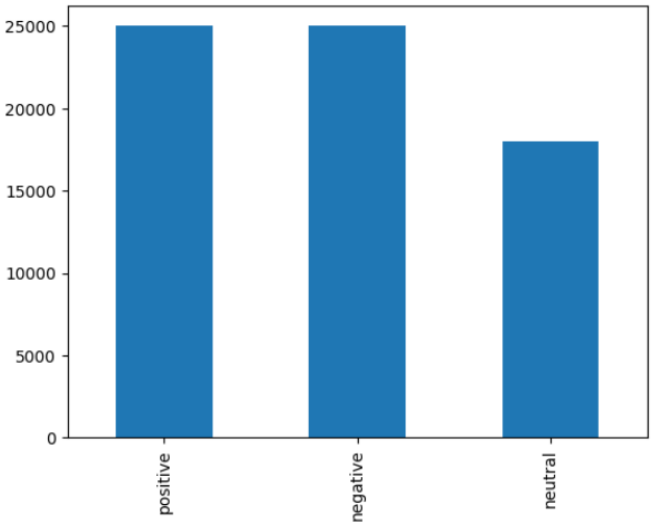
Introduction

The Sentiment Analysis API is designed to analyze the sentiment of text, providing a quick and reliable assessment of the emotional tone conveyed in the input. Built using the FastAPI framework, the API leverages a trained deep learning model to predict sentiment with high accuracy.

Model training

Data visualization

	text	sentiment
0	I love this product!	positive
1	pen	neutral
2	This movie is terrible.	negative
3	The food at this restaurant was amazing!	positive
4	I was really disappointed with the service.	negative
...	...	...
58902	i could totally do without but i feel like it ...	positive
58903	i eat i do feel more selfish like i am not thi...	negative
58904	i didn t feel groggy necessarily but still fel...	negative
58905	i paint feels like a spark of the divine meeti...	positive



Data Preprocessing Steps:

Set Maximum Words:

Define max_words to limit the number of unique words based on word frequency. In this case, max_words = 10000 is chosen.

Load Data:

Load the sentiment dataset from a CSV file (dataset.csv in this case).

Resampling:

Separate the data into positive, negative, and neutral examples.

Resample the examples to balance the dataset. For instance, resample positive and negative examples to 25,000 each and neutral examples to 18,000.

Concatenate Resampled Examples:

Concatenate the resampled examples back into a single DataFrame (df_resampled).

Text Preprocessing:

Define a preprocess_text function to clean and preprocess each text entry.

1. Remove special characters and convert to lowercase.
2. Tokenize the text.
3. Remove stopwords.
4. Join tokens back into a string.

Remove Null Values:

Remove rows with null values in the 'text' column.

Split Data:

Split the data into text (x) and labels (y).

Label Encoding:

Use LabelEncoder to convert categorical labels into numerical format.

Train-Test Split:

Split the data into training and test sets using train_test_split.

One-Hot Encoding:

Convert categorical labels to one-hot encoded vectors using to_categorical.

Tokenization:

Tokenize the text using Tokenizer and fit it on the training set.

Convert text sequences to numerical sequences.

Padding Sequences:

Pad sequences to have the same length (maxlen).

Print the maximum sequence length (maxlen).

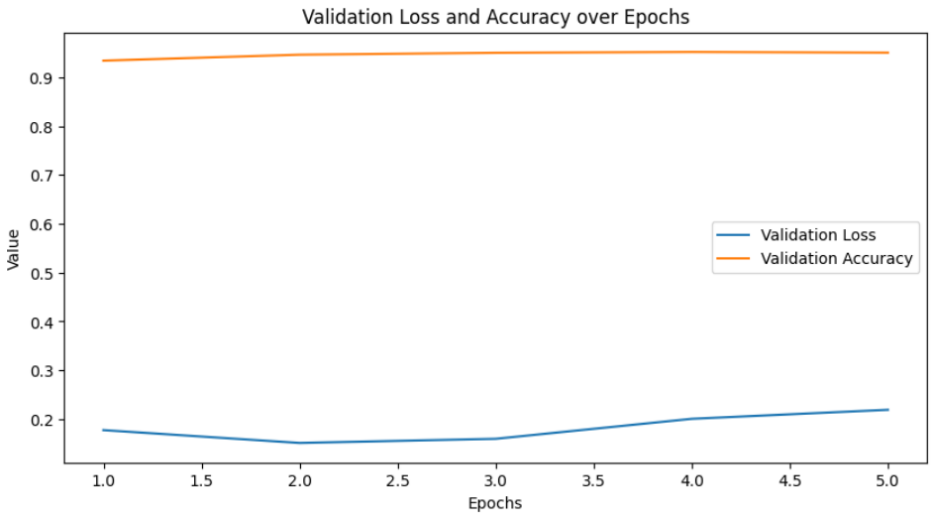
Model architecture

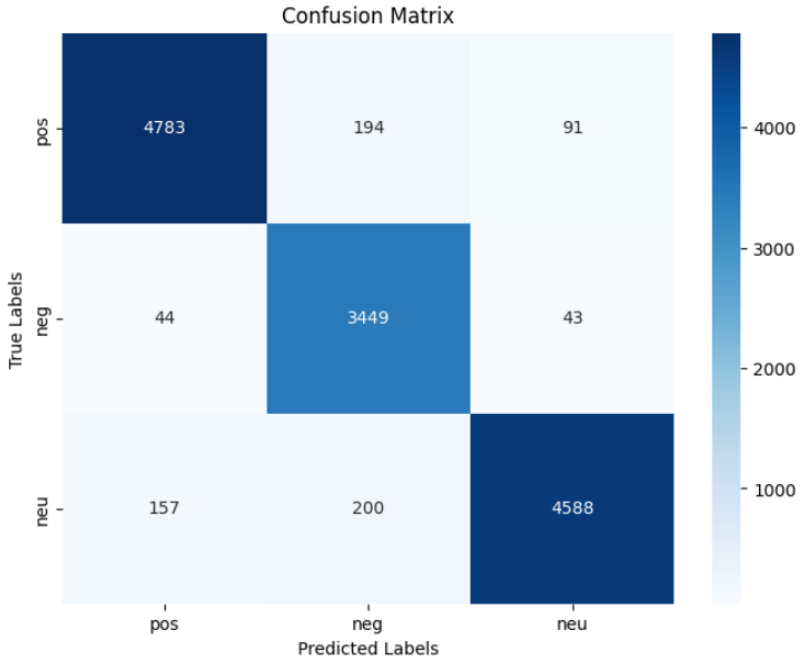
The implemented CNN 1D model for sentiment analysis can be summarized as follows:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 129, 128)          1280000

 conv1d_1 (Conv1D)           (None, 125, 128)          82048

 global_max_pooling1d_1 (Gl  (None, 128)               0
 obalMaxPooling1D)

 dense_2 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 3)                 195

=================================================================
Total params: 1370499 (5.23 MB)
Trainable params: 1370499 (5.23 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Performance evaluation

The training and validation accuracies are a crucial metric that measures the model's ability to correctly classify examples from the training dataset. In the case of the sentiment analysis model, the training and validation accuracies provide insights into how well the model has learned to predict sentiments during the training phase.





Training Accuracy

The training accuracy at Epoch 5 reached 98.90%. This metric measures the model's proficiency in classifying sentiments within the training dataset.

Validation Accuracy

The validation accuracy at Epoch 5 was 95.04%. This metric provides insights into the model's generalization performance on unseen data during the training process.

Test Accuracy:

The test accuracy achieved on an independent test set is 94.62%. This metric evaluates the model's ability to generalize and accurately classify sentiments on new, unseen data.

Notebook Link :

https://colab.research.google.com/drive/1ARF90O_-OIhTsEIgQD9PqbVsKVsQjH_r?usp=sharing

API Development with FastAPI

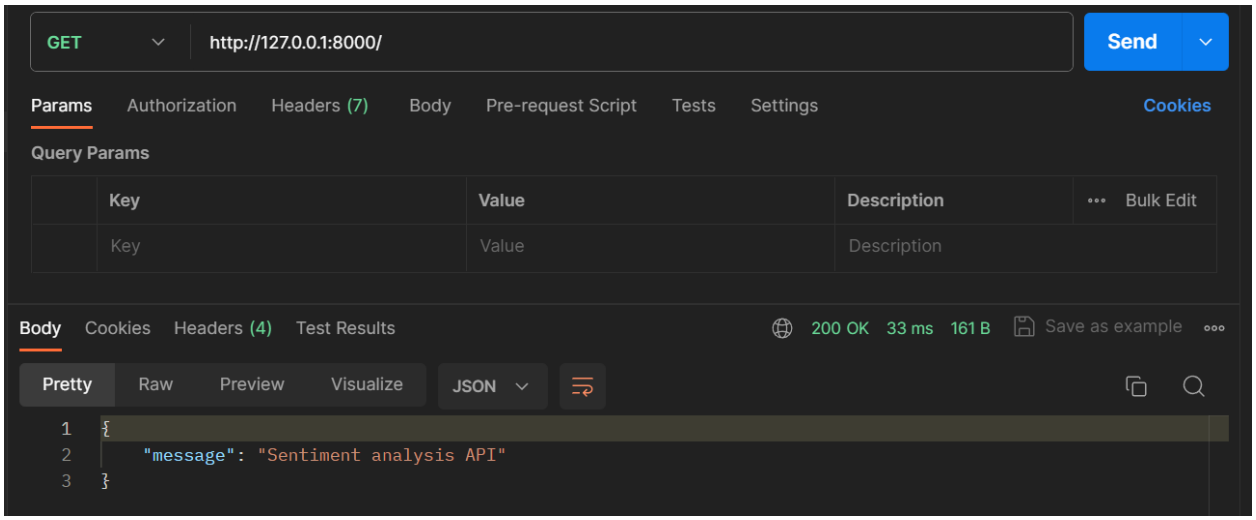
FastAPI Framework

The sentiment analysis API is implemented using FastAPI, a modern, fast, and web framework for building APIs with Python.

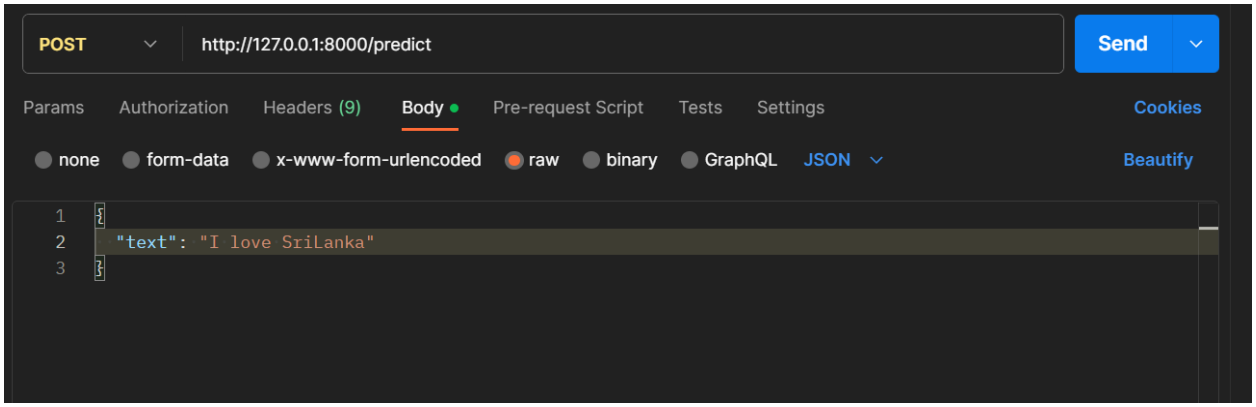
API Endpoint for Sentiment Analysis:

Created a POST endpoint /predict to accept text data for sentiment analysis.

The API has a root endpoint (/) providing a brief message.



The main sentiment analysis endpoint is /predict, accepting POST requests with JSON payloads containing text data.

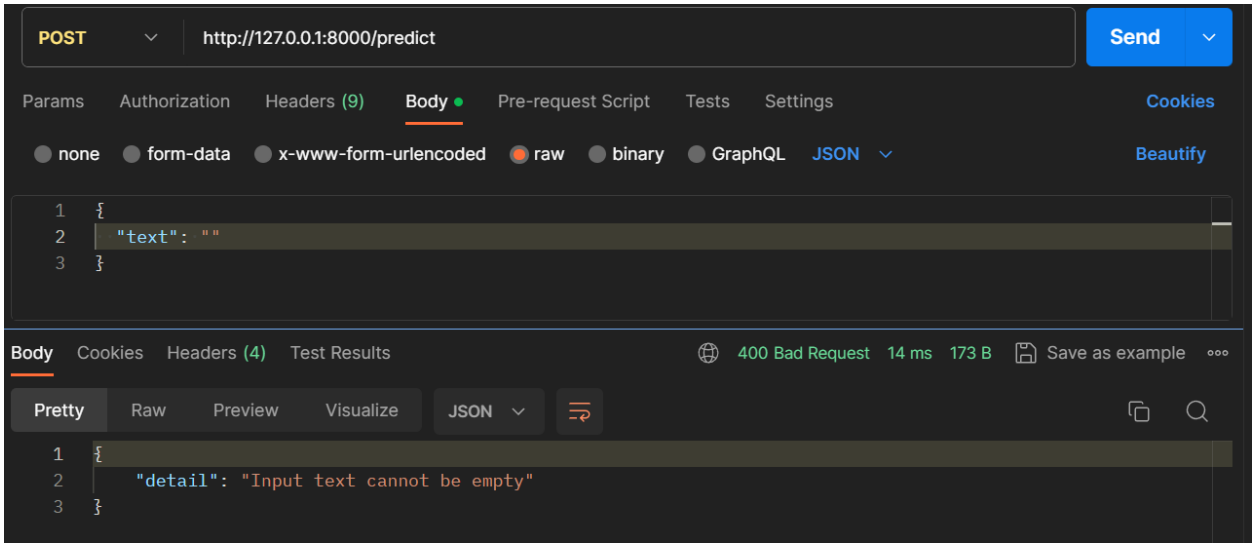


Handling Multiple Requests:

Incorporated asynchronous features using async and await to handle multiple requests simultaneously, ensuring efficient and responsive API performance.

Input Validation:

Implemented input validation to ensure that the input data adheres to the expected format and structure.



Error Handling:

Included proper error handling mechanisms for various scenarios, such as invalid inputs or server errors. This enhances the robustness of the API.

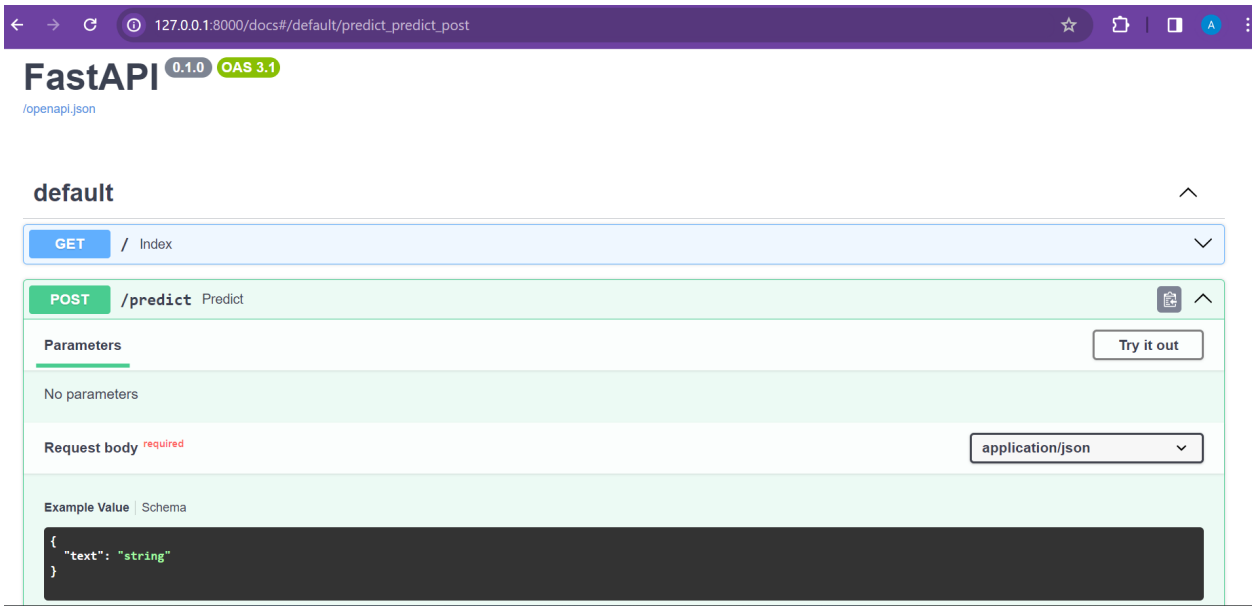
Logging:

Implemented logging to track incoming requests and corresponding responses. This aids in monitoring and debugging.

API Documentation:

Documented API endpoints using FastAPI's automatic OpenAPI and JSON Schema generation. The documentation provides a clear overview of available endpoints and expected inputs.

Link : http://127.0.0.1:8000/docs (Ensure that your API server is running)



Testing the Sentiment Analysis API

1.Index Endpoint:

The / endpoint is tested to ensure it returns a status code of 200 and the expected message.

2.Predict Endpoint - Valid Input:

A positive test case with valid input is conducted to verify that the /predict endpoint returns the expected JSON response structure.

3.Predict Endpoint - Invalid Input:

An invalid input scenario is simulated, checking if the API correctly responds with a status code of 422 (Unprocessable Entity) and provides detailed error information.

4.Predict Endpoint - Internal Server Error:

An internal server error scenario is simulated by providing invalid text (None), ensuring the API responds appropriately.

5.Predict Endpoint - Boundary Input Length:

A test case is designed to assess the API's behavior with input of the maximum allowed length. This tests the model's handling of boundary input conditions.

6.Predict Endpoint - Empty Input:

The API is tested with an empty input to verify that it responds with the correct status code (400).

7.Predict Endpoint - Long Text Input:

A test case with a very long input text is performed to evaluate the API's ability to handle lengthy inputs.

8.Predict Endpoint - Invalid Text Format:

The API is tested with an input of an invalid format (numeric), checking if it responds with a status code of 422.

9.Predict Endpoint - Invalid Input Characters:

The API is tested with input containing special characters to ensure it handles such characters appropriately.

10.Predict Endpoint - Whitespace Input:

The API is tested with whitespace-only input, assessing its response to this scenario.

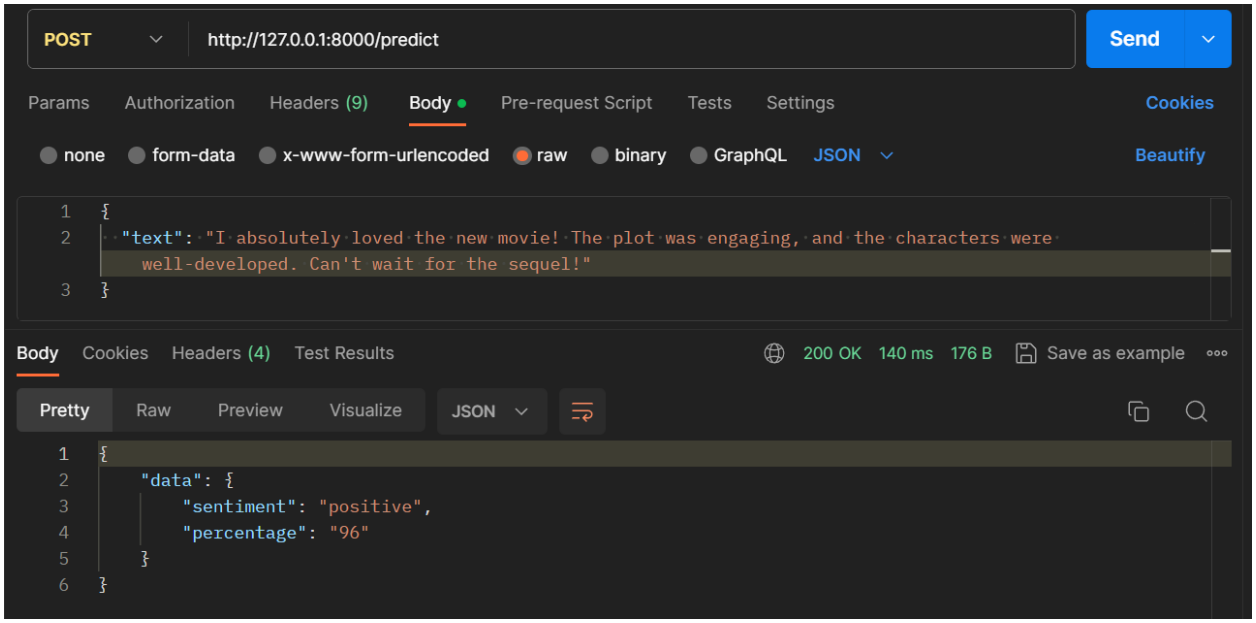
11.Multiple Requests Simultaneously:

The API is subjected to multiple simultaneous requests using concurrent.futures.ThreadPoolExecutor to evaluate its performance under concurrent usage.

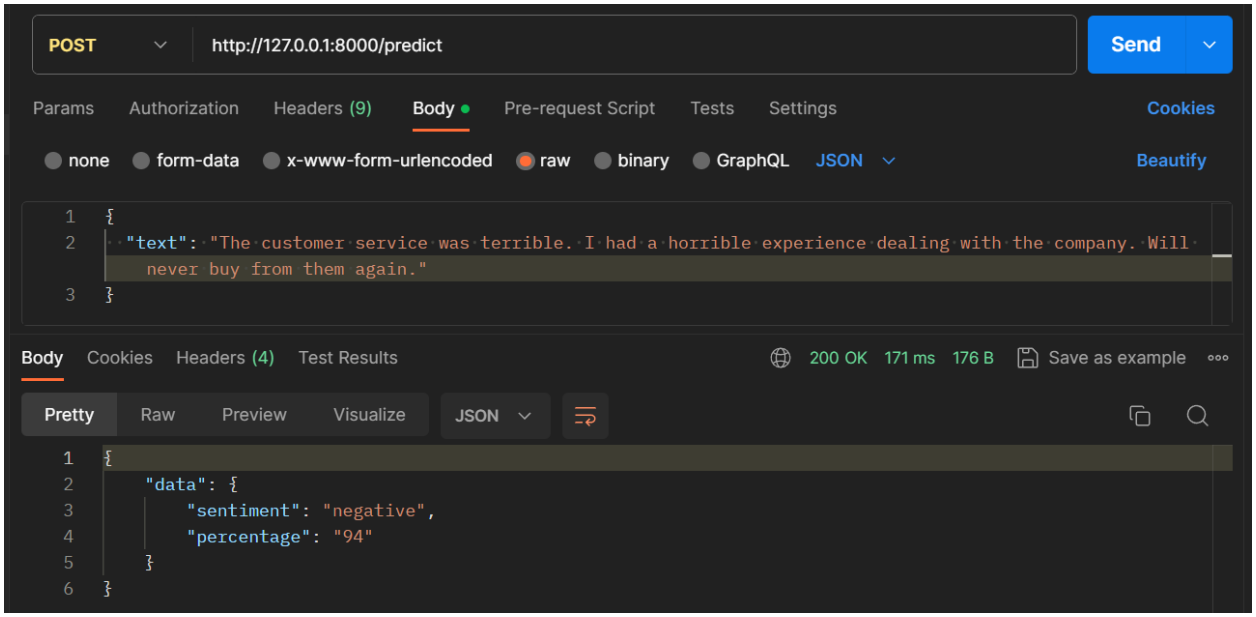
```
rootdir: D:\ML_prijects\SentimentAnalysisModelmaster
plugins: anyio-3.7.1, dvc-3.12.0, hydra-core-1.3.2
collected 11 items

tests\test_main.py ...........                                                [100%]

============================================ 11 passed in 7.18s ============================================
```

Output for each cases

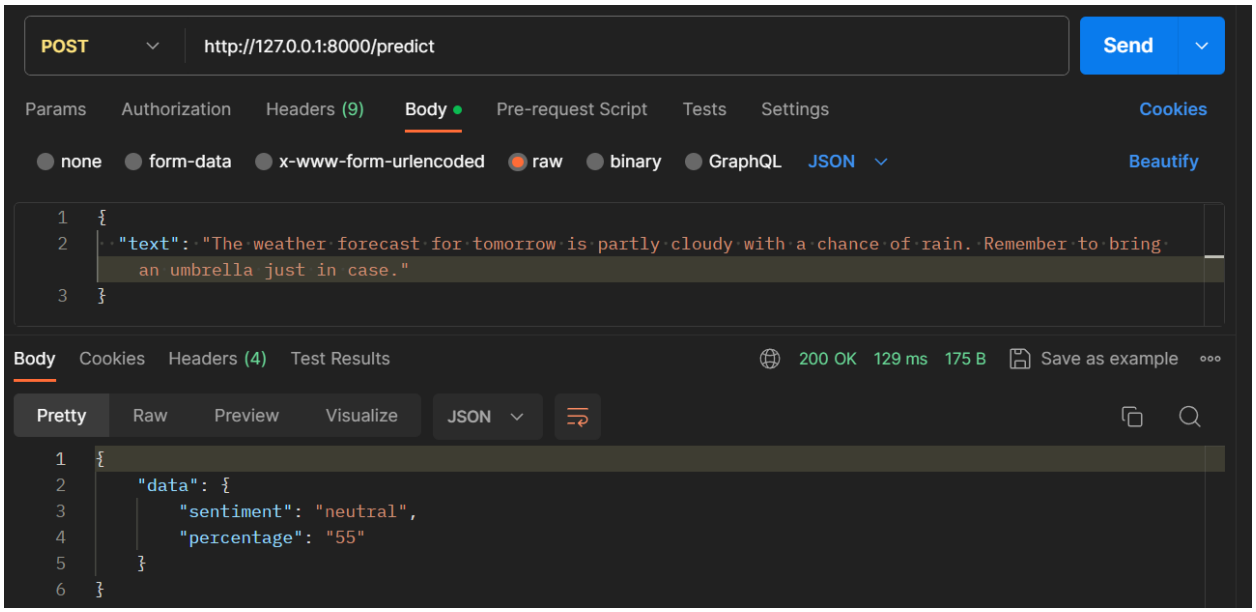
Positive



Negative



Neutral



You can find the code on GitHub:

https://github.com/Ahsanmnm/Sentiment-Analysis.git