Sentiment Analysis API Documentation

The Sentiment Analysis API allows you to analyze the sentiment of textual input using a pre-trained sentiment analysis model. It uses the Hugging Face Transformers library to perform the sentiment analysis task. This API is made using **Django REST Framework.**

Table of Contents

- Getting Started
  - Prerequisites
  - Installation
- Sentiment Analysis Model from Huggingface
  - Saving the Model Locally (Recommended)
  - Using the Model Directly from Huggingface (Not Recommended)
- Running the API
- Endpoints
  - URL: http://127.0.0.1:8000/analyze/
  - Method: POST
  - Request Body Parameters
  - Example Body Parameter (JSON content)
  - Success Response
- Handling Huggingface Input/Output JSON Format
  - Input JSON Format
  - Output JSON Format
  - Label to Sentiment Mapping
- Error Handling
  - Error Handling Example

Getting Started

Prerequisites

To use the Sentiment Analysis API, ensure you have the following prerequisites:

Python (I have used version 3.7.9)

You need to install these using pip
Django framework (version 3.2.19)
Transformers library (version 4.30.2)
Rest Framework (version 3.14.0)

(Please check requirements.txt file)

Installation

To install the required dependencies, follow these steps:
Clone the project repository from GitHub:

git clone https://github.com/FahimShahryer/Sentiment_API.git

Navigate to the project directory where requirements.txt is located:

pip install -r requirements.txt

Sentiment Analysis Model from Huggingface
The Sentiment Analysis API utilizes the twitter-roberta-base-sentiment model from Hugging Face, which is a pre-trained model fine-tuned on a large corpus of text data.

There are two ways:

- Saving the model locally and use it (download the model)

- Access the model directly from huggingface (using internet)

But I prefer saving the model locally because-
Saving the model locally allows you to access and use the pre-trained sentiment analysis model without having to download it every time you run your application. When you save the model locally, you store it in a directory on your local machine. Other advantages are offline Access, improved Performance, version Control.

(Option 1) Saving the Model Locally (**Recommended)**
To use the model locally, follow these steps:

Download the pre-trained model from **huggingface.co/cardiffnlp/twitter-roberta-base-sentiment** or you can download the model and other necessary file from my google drive link:

https://drive.google.com/drive/folders/1k6Mte5dgcBWzEy3KEaRhBvVw8_TCNPTN?usp=sharing

Save the model files in the directory **myapp/models** within the project.

(Option 2) Using directly from huggingface: (**Not Recommended)**

Then you have to modify the code of **myapp/views.py**
Replace the specific part of the code of **myapp/views.py** with this:

```
@api_view(['POST'])
def sentiment_analysis(request):
   text = request.data.get('text')
   classifier = pipeline(task="sentiment-analysis",
model="cardiffnlp/twitter-roberta-base-sentiment")
   preds = classifier(text)
   json_data = preds[0]['label']
   response_data = {'sentiment': json_data}
   return JsonResponse(data=response_data, safe=False)
```

Running the API

To run the Sentiment Analysis API locally, follow these steps:
Navigate to the project directory in the terminal and start the Django development server:

```
python manage.py runserver
```

The API will be accessible at
http://127.0.0.1:8000/

Endpoints

URL: **http://127.0.0.1:8000/analyze/**
Method: **POST**
This endpoint analyzes the sentiment of a given text.

Request Body Parameters:
text (string, required): The text to be analyzed.

Example Body parameter (JSON content):

```
{
  "text":"the product is very good"
}
```

Success Response:
sentiment (string): The predicted sentiment of the text. Possible values: "NEGATIVE", "NEUTRAL", "POSITIVE".

```
{
   "sentiment": "POSITIVE"
}
```

DONE testing the API walkthrough!

Handling Huggingface Input/Output JSON Format
The Sentiment Analysis API follows the Hugging Face Transformers library's input/output JSON format. The input to the model should be a JSON object containing the text to be analyzed, and the output is a JSON object containing the predicted sentiment.

Input JSON format:

```
{
   "text": "I am feeling great!"
}
```

Output JSON format:

```
[
  [
    {
      "label": "LABEL_2",
      "score": 0.9782516956329346
    },
    {
      "label": "LABEL_1",
      "score": 0.018881788477301598
    },
    {
      "label": "LABEL_0",
      "score": 0.0028664814308285713
    }
  ]
]
```

Here,

```
          LABEL_0: "NEGATIVE"
          LABEL_1: "NEUTRAL"
          LABEL_2: "POSITIVE"
```

So In had to manage the label to human-readable sentiment names / expected output format.

Error Handling

The Sentiment Analysis API handles errors and provides appropriate responses in case of invalid requests or errors during processing.

Error Handling Example

If the text field is missing in the request or empty, the API will respond with a 400 Bad Request status code and an error message in the response body.

Example error response for a missing 'text' field:

```
{
   "error": "Invalid input. 'text' field is required."
}
```

If the input text exceeds the maximum allowed length (1000 characters), the API will respond with a 400 Bad Request status code and an error message in the response body.

Example error response for exceeding the maximum text length:

```
{
   "error": "Invalid input. Maximum allowed text length is 1000 characters."
}
```