**Setup**

The text-analyzer service written in FastApi can be run locally using uvicorn. Steps to follow are

* Once inside the root folder of project, create a virtual environment using the command  
  python -m venv <venv-name>
* Activate the virtual environment using  
  source <venv-name>/bin/activate
* Install the requirements for the project using  
  pip install -r requirements.txt
* Then run the app using   
  uvicorn src.main:app –reload

We can access the api docs and redocs page at  
http://127.0.0.1:8000/docs  
http://127.0.0.1:8000/redoc

**Project Structure and choices**

**├── README.md**

**├── requirements.txt**

**├── Setup.docx**

**├── src**

**│   ├── \_\_init\_\_.py**

**│   ├── keywords.py**

**│   ├── main.py**

**│   ├── models.py**

**│   ├── router.py**

**│   └── service.py**

**└── tests**

**├── \_\_init\_\_.py**

**├── test\_routes.py**

**└── test\_service.py**

For this project instead of adding all the logic and endpoint to a single file main.py, I choose to separate the implementation.

* Add router details to router.py, which is then included into main app. This gives better control over the router and make the main file simple and small.
* Since the input text is of free-text format there is no pydantic model specified for input. But for output I have defined a **FeedbackResponse** pydantic class. Its defined in the model.py
* The actual logic of sentiment analysis and word count is separated from router as well. This logic is added to service.py
* In the service there are few constants used like SENTIMENTS, NEGATION\_WORDS etc.. These are added to a separate file called keywords.py, so that all such constants are in one place and service layer stays clean.
* The sentiment analysis logic I have used **nltk.corpus.stopwords** instead of using static list of stop words. Because nltk provides a comprehensive list which can easily be used to remove stop words from sentence.
* Also I have used colletions.Counter for counting words in sentence instead of manual counting. Because Counter class provider counter.most\_common() function which returns top N frequent elements

All the tests for service and router layer are created under /tests folder

**Example request/response**

Once the service is up and running we can use CURL command or use Postman for hitting the endpoint

* Positive sentiment

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Content-Type: text/plain' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw 'the service was very good and fabulous'  {  "word\_count": 7,  "most\_common\_words": [  "service",  "good",  "fabulous"  ],  "sentiment": "positive"  } |

* Negative sentiment

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Content-Type: text/plain' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw 'the service was very bad'  {  "word\_count": 5,  "most\_common\_words": [  "service",  "bad"  ],  "sentiment": "negative"  } |

* Neutral sentiment

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Content-Type: text/plain' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw 'She went to the store to buy groceries.'  {  "word\_count": 8,  "most\_common\_words": [  "went",  "store",  "buy"  ],  "sentiment": "neutral"  } |

* Negation logic

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Content-Type: text/plain' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw 'It wasn'\''t a bad presentation.'  {  "word\_count": 5,  "most\_common\_words": [  "bad",  "presentation"  ],  "sentiment": "positive"  } |

* Empty String

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw ''  {  "detail": "Input text cannot be empty"  } |

* Different content type

|  |
| --- |
| curl --location --request POST 'http://127.0.0.1:8000/analyze' \  --header 'User-Agent: Apidog/1.0.0 (https://apidog.com)' \  --header 'Content-Type: application/json' \  --header 'Accept: \*/\*' \  --header 'Host: 127.0.0.1:8000' \  --header 'Connection: keep-alive' \  --data-raw 'It wasn'\''t a bad presentation.'  {  "detail": "Only 'text/plain' content is supported."  } |