Web API for NLP Sentiment Analysis, Topic Detection, and Keyword Cloud Generation

1. Introduction

This document outlines the requirements for developing a web API in Python that utilizes Natural Language Processing (NLP) techniques for sentiment analysis, topic detection, and keyword cloud generation.

2. Goals

The primary goals of the project are:

- To develop a RESTful web API that provides endpoints for sentiment analysis, topic detection, and keyword cloud generation.
- To integrate an NLP library into the API for processing textual data.
- To ensure the API is scalable, efficient, and easy to use for developers.

3. Functional Requirements

3.1 Sentiment Analysis

- The API should accept textual input and return the sentiment analysis results.
- Sentiment analysis should classify the input text into positive, negative, or neutral sentiment.
- The API should provide a confidence score or probability for each sentiment classification.

3.2 Topic Detection

- The API should accept textual input and return the detected topics.
- Topic detection should identify the main themes or subjects present in the input text.
- The API should support detecting multiple topics within a single text.

3.3 Keyword Cloud Generation

- The API should accept textual input and generate a keyword cloud.
- Keyword cloud generation should highlight the most significant keywords or phrases in the input text.
- The API should allow customization options such as the number of keywords and visual style of the cloud.

4. Non-functional Requirements

4.1 Performance

- The API should be capable of handling multiple concurrent requests efficiently.

- Response times for sentiment analysis, topic detection, and keyword cloud generation should be kept minimal.

4.2 Scalability

- The API should be designed to scale horizontally to accommodate increasing user load.
- It should utilize asynchronous processing where applicable to improve scalability.

4.3 Security

- The API should implement secure communication protocols (e.g., HTTPS) to protect data transmission.
- Input validation should be performed to prevent injection attacks and ensure data integrity.

4.4 Usability

- The API should have clear and well-documented endpoints, parameters, and response formats.
- Error messages should be informative and user-friendly to aid in troubleshooting.

5. Technology Stack

- Python: The API will be developed using Python programming language.
- NLP Library: An appropriate NLP library (e.g., NLTK, spaCy) will be chosen for implementing sentiment analysis, topic detection, and keyword cloud generation.
- Web Framework: Flask or Django will be used as the web framework for developing the API.
- Deployment: The API will be deployed on a cloud platform such as AWS, Azure, or Google Cloud.

6. API Endpoints

The following endpoints will be exposed by the API:

- **/sentiment-analysis**: Endpoint for performing sentiment analysis on text.
- **/topic-detection**: Endpoint for detecting topics in text.
- **/keyword-cloud**: Endpoint for generating a keyword cloud from text.

7. Data Flow

- Client applications will send HTTP requests to the API endpoints with textual data.
- The API will utilize the chosen NLP library to process the input text and generate the desired outputs.
- Processed results will be returned to the client applications as JSON responses.

8. Testing

- Unit tests and integration tests will be implemented to ensure the correctness and reliability of the API.
- Test cases will cover various scenarios for different input texts and expected outputs.
- Automated testing frameworks such as pytest will be used for testing purposes.

9. Documentation

- Comprehensive documentation will be provided for the API, including usage instructions, endpoint descriptions, parameter details, and example responses.
- Documentation will be available in both human-readable format (e.g., Markdown) and machine-readable format (e.g., Swagger/OpenAPI).

10. Future Considerations

- Support for additional NLP tasks (e.g., named entity recognition, text summarization) may be added in future iterations.
- Continuous integration and deployment pipelines may be implemented to streamline the development and deployment process.

Here are sample input and output examples for each of the endpoints defined in the web API:

1. Sentiment Analysis Endpoint (/sentiment-analysis)

Sample Input:

```
{
 "text": "I love the product! It exceeded my expectations."
}
```

Sample Output:

```
{
 "sentiment": "positive",
 "confidence": 0.85
}
```

2. Topic Detection Endpoint (/topic-detection)

Sample Input:

```
{
 "text": "The conference covered a wide range of topics including artificial intelligence, machine learning, and natural language processing."
}
```

Sample Output:

```
{
 "topics": ["artificial intelligence", "machine learning", "natural language processing"]
}
```

3. Keyword Cloud Generation Endpoint (/keyword-cloud)

Sample Input:

```
{
  "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."
}
```

Sample Output:

```
{
  "keywords": [
    {"keyword": "dolor", "count": 2},
    {"keyword": "consectetur", "count": 1},
    {"keyword": "adipiscing", "count": 1},
    {"keyword": "elit", "count": 1},
    {"keyword": "tempor", "count": 1},
    {"keyword": "incididunt", "count": 1},
    {"keyword": "labore", "count": 1},
    {"keyword": "magna", "count": 1},
    {"keyword": "aliqua", "count": 1},
    {"keyword": "veniam", "count": 1}
  ]
}
```