ITCS-6177 System Integration

Sentiment Analysis

By: Ruchira Ingle (801312087)

*Email:* [*ringle7@uncc.edu*](mailto:skulka37@uncc.edu)

Course Project Report



**TABLE OF CONTENTS**

1. MOTIVATION AND CONTEXT 3
2. OVERVIEW 3
3. IMPLEMENTATION 4
4. PROJECT MILESTONES 8
5. TESTING THE API………………………………………………………………………………………………………………………………...8
6. FUTURE SCOPE 10
7. REFERENCES 10
8. **MOTIVATION AND CONTEXT**

**Sentiment Analysis:**

* + It involves identifying and categorizing the text in terms of its polarity, such as whether it is positive, negative, or neutral.
  + Sentiment Analysis is frequently used to analyze and track sentiments towards a specific category, topic, or product.
  + Finding out how consumers, customers, or the general public feel about a given product, service, or subject is made easier.
  + Its growing importance can be attributed to its wide use in elections across various countries for predicting the election result within electoral boundaries or constituencies.
  + A Sentiment Analysis API is built and presented in this report.

1. **OVERVIEW**

Sentiment Analysis has been implemented using the Azure Text Analytics services. Below steps were followed to accomplish the task:

1. Azure Account creation.
2. Text Analytics resource creation.
3. Created a node js code to call the vendor API.
4. Prediction: Predicting the sentiment of sentences.
5. **IMPLEMENTATION:**
   * + 1. **Azure Account creation**

To go ahead for this project, first we need to create a azure account. It’s the only way to access the Azure services, in our case Text Analytics service. Azure Text Analytics is a language service provided by Microsoft Azure that enables developers to extract valuable insights from text data. It offers a range of natural language processing capabilities, including sentiment analysis, key phrase extraction, named entity recognition, and language detection. With Text Analytics, developers can easily integrate these language processing features into their applications to derive meaningful information and enhance text-based analysis.

* + - 1. **Text Analytics resource creation**

For a developer to write a code to access the text analytics service of Azure, they first need to create a azure conginitive service resource which will provide them with the key and endpoint which we will need to implement the API.

To create a Text Analytics resource in Azure, follow these steps:

* + - * 1. Login to the Azure portal (portal.azure.com).
        2. Click on the "Create a resource" button (+) in the top left corner.

A screenshot of a computer

Description automatically generated

* + - * 1. Search for "Text Analytics" in the Azure Marketplace search bar.
        2. Select "Text Analytics" from the search results.

A screenshot of a computer

Description automatically generated

* + - * 1. Click on the "Create" button to start the resource creation process.
        2. Provide the required information such as subscription, resource group, and resource name.
        3. Choose the pricing tier and location for the resource.
        4. Configure additional settings if necessary.
        5. Review the summary and click on the "Create" button to create the Text Analytics resource.
        6. Wait for the deployment to complete.
        7. Once the resource is created, you can access it from the Azure portal and retrieve the endpoint URL and access keys for using the Text Analytics API.

Note: You may need appropriate permissions and a valid Azure subscription to create the Text Analytics resource.

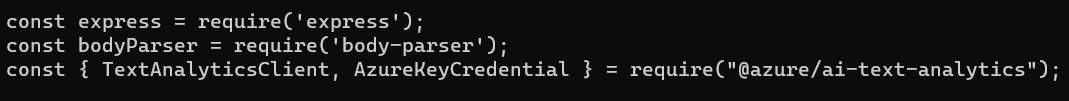
To access the key from the Text Analytics resource in Azure, follow these steps:

* + - * 1. Login to the Azure portal (portal.azure.com).
        2. Navigate to the Text Analytics resource you created.
        3. In the left-hand menu, click on "Keys and Endpoint" under the "Resource Management" section.
        4. On the "Keys and Endpoint" page, you will find two keys: "Key1" and "Key2". These keys are used for authentication when accessing the Text Analytics service.
        5. Copy either one of the keys by clicking on the copy icon next to it.
        6. You can now use the copied key in your code or application to authenticate and access the Text Analytics service.

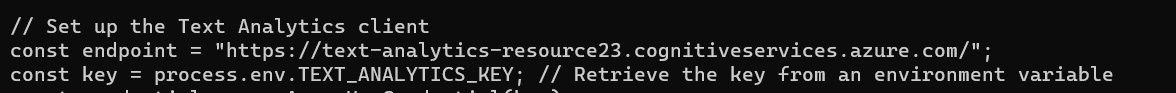
Note: Treat these keys as sensitive information and ensure they are stored securely. Avoid sharing the keys publicly or committing them to public repositories.

* + - 1. **Code implementation**

1. This is a Node.js application that uses the Express.js framework to create a server that listens for incoming HTTP requests on port 3000.
2. The application uses the body-parser middleware to parse incoming request bodies, and the @azure/ai-text-analytics package to perform sentiment analysis on text.



1. The TextAnalyticsClient is set up using an API endpoint and key, which are retrieved from environment variables.



1. The root route / responds to a GET request by serving an HTML form that allows the user to input text to analyze. The form submits the text as a POST request to /sentiment.
2. The /sentiment route handles incoming POST requests by extracting the text from the request body, calling the Text Analytics API to perform sentiment analysis on the text, and returning an HTML response that displays the sentiment score and the original text.
3. If an error occurs during the processing of the request, the server sends a 500 Internal Server Error response with a JSON object that contains an error property.
4. Finally, the server is started by calling the listen() method of the app object and passing in the port number and IP address on which to listen for incoming requests.
5. Error handling: The code has error handling for the async function called in the app.post('/sentiment', async (req, res) => {...}) route.
6. If an error occurs while executing the try block, it will catch the error and log it to the console with console.error(error) and then send an error response with res.status(500).json({ error: 'An error occurred' }).
7. This error handling ensures that if any error occurs, the server does not crash, and the user receives an appropriate error response instead of a blank page or an error message that does not explain the problem. Additionally, the console.error(error) statement helps the developer to debug the issue and identify its root cause.
   * + 1. **Prediction**

The Text Analytics service works by leveraging natural language processing (NLP) techniques to analyze and extract insights from text data. When it comes to sentiment analysis, the service aims to determine the overall sentiment or emotional tone expressed in a piece of text, whether it is positive, negative, or neutral.

Here's a high-level overview of how the Text Analytics service calculates the sentiment score:

Text Preprocessing: The service preprocesses the input text by removing any noise, such as special characters or punctuation marks, and normalizing the text to a standard format. This step helps to ensure accurate analysis.

Sentiment Classification: The core of sentiment analysis involves classifying the sentiment of the text. The service uses a combination of machine learning algorithms and linguistic rules to understand the sentiment expressed in the text.

Feature Extraction: The service identifies key features or linguistic patterns within the text that are indicative of sentiment. These features may include positive or negative words, sentiment-bearing phrases, or contextual cues.

Sentiment Scoring: Based on the identified features, the service assigns sentiment scores to the text. The sentiment score typically represents the degree of positivity or negativity in the text. For example, a higher positive sentiment score indicates a more positive sentiment expressed in the text.

Confidence Scores: The Text Analytics service also provides confidence scores for each sentiment category, such as positive, negative, or neutral. These scores indicate the level of certainty in the sentiment classification. Higher confidence scores imply a higher level of confidence in the assigned sentiment label.

It's important to note that sentiment analysis is an ongoing field of research, and different sentiment analysis models and techniques may be used by the Text Analytics service. The underlying models are trained on large datasets and are continuously improved to provide accurate sentiment analysis results.

By utilizing the Text Analytics service in your code, you can leverage its sentiment analysis capabilities to extract sentiment scores and gain insights from textual data in your applications.

1. **PROJECT MILESTONES**
   * **Definitely Accomplish:** The code sets up a server using Express.js framework to expose two endpoints: one from analyzing sentiment from POST request and another for analyzing sentiment from GET request. It integrates with the Azure Text Analytics service to extract sentiment scores based on provided text, allowing users to obtain sentiment prediction easily.
     + Completed
   * **Ideally Accomplish:** Implement sentiment analysis by building a dataset from scratch using Twitter API and build a user interface.
2. **Testing the API:**

During the prediction process, the Text Analytics service will assign a score between 0 and 1 to the input sentence. A score of 0 indicates a negative statement, while a score closer to 1 indicates a positive statement.

This API has two requests - a GET request and a POST request:

* + - GET Request:

Example: text= “I am very happy today”

Steps:

Enter the URL “http://134.209.112.193:3000/” in a browser.

Enter your sentence in the text box on the page and hit “Analyze” button.

A screenshot of a computer

Description automatically generated with medium confidence

The results will appear on the next page under “Analysis Results”.

A screenshot of a computer

Description automatically generated with medium confidence

* + - POST Request:

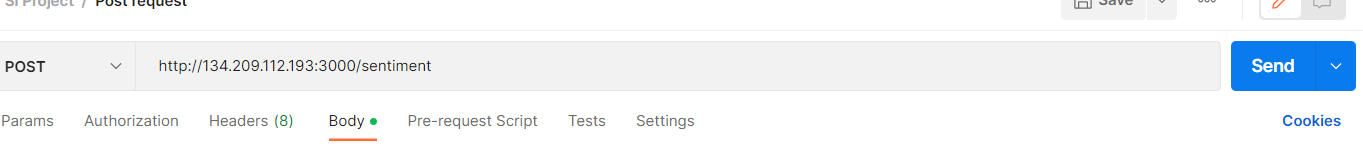
Example: text= “I am very tired today”

Steps:

Open Postman and create a new request.

Set the request type to POST.

Enter the URL for your endpoint (e.g.http://134.209.112.193:3000/sentiment).



Select the Body tab.

Select the raw option and set the format to JSON.

Enter the request body as a JSON object with a text property (e.g. {"text": "I love this product!"}).

A screenshot of a computer

Description automatically generated with low confidence

Click the Send button to send the request to the server.

1. **FUTURE SCOPE**
   * Handle double negation in words like “I am not happy today” should be classified as negative sentiment but the “happy” keyword might contribute to classify it as positive
   * The API should intake a dataset and predict the sentiment of using the given data.
   * Include more variety of words.
   * Implement information retrieval techniques like TF-IDF.
2. **REFERENCES**
   * Express.js: https://expressjs.com/
   * Body-parser middleware: https://www.npmjs.com/package/body-parser
   * Azure Text Analytics API: https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/quickstarts/nodejs-sdk
   * Azure SDK for JavaScript: <https://github.com/Azure/azure-sdk-for-js>