Project Report: Sentiment Analysis Web App

Name: Aniket Bakre

Email:- [aniketbakre1291@gmail.com](mailto:aniketbakre1291@gmail.com)

LinkedIn: https: [www.linkedin.com/in/aniketbakre](http://www.linkedin.com/in/aniketbakre)

Date:16-07-2024

2024

Table of Contents

[1. Introduction 2](#_Toc171982658)

[1.1 Background 2](#_Toc171982659)

[1.2 Purpose 2](#_Toc171982660)

[2. Objectives 2](#_Toc171982661)

[3. System Design 2](#_Toc171982662)

[3.1 Architecture 2](#_Toc171982663)

[3.2 Components 2](#_Toc171982664)

[4. Implementation 2](#_Toc171982665)

[4.1 Front-end 2](#_Toc171982666)

[4.1.1 index.html 2](#_Toc171982667)

[4.1.2 styles.css 3](#_Toc171982668)

[4.1.3 script.js 4](#_Toc171982669)

[4.2 Back-end 5](#_Toc171982670)

[4.2.1 app.py 5](#_Toc171982671)

[4.3 Model 5](#_Toc171982672)

[5. Results and Discussion 5](#_Toc171982673)

[5.1 Performance 6](#_Toc171982674)

[5.2 User Experience 6](#_Toc171982675)

[6. Challenges and Solutions 6](#_Toc171982676)

[7. Future Work 6](#_Toc171982677)

[8. Conclusion 6](#_Toc171982678)

[9. References 6](#_Toc171982679)

# 1. Introduction

## 1.1 Background

Sentiment analysis, also known as opinion mining, is a sub-field of natural language processing (NLP) that aims to determine the sentiment or emotion expressed in a piece of text. This web application provides users with a simple interface to input text and receive sentiment analysis results.

## 1.2 Purpose

The purpose of this project is to develop a web-based application that uses a pre-trained DistilBERT model to perform sentiment analysis on user-provided text.

# 2. Objectives

- Develop a user-friendly web interface for sentiment analysis.

- Implement a backend using Flask to handle text input and output.

- Integrate a pre-trained DistilBERT model for sentiment analysis.

- Display sentiment analysis results in a clear and concise manner.

# 3. System Design

## 3.1 Architecture

The system architecture consists of a front-end, back-end, and the sentiment analysis model. The front-end is built using HTML, CSS, and JavaScript, while the back-end is developed using Flask. The sentiment analysis model is a pre-trained DistilBERT model.

## 3.2 Components

- \*\*Front-end\*\*: HTML, CSS, JavaScript

- \*\*Back-end\*\*: Flask, Python

- \*\*Model\*\*: DistilBERT for sentiment analysis

# 4. Implementation

## 4.1 Front-end

The front-end interface consists of a text area for input, a button to trigger sentiment analysis, and a section to display the results.

## 4.1.1 index.html

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sentiment Analysis</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<div class="container">

<h1>Sentiment Analysis</h1>

<textarea id="inputText" placeholder="Enter or paste your text here..."></textarea>

<button id="checkEmotionBtn" onclick="checkEmotion()">Check Emotion</button>

<div id="output"></div>

<button onclick="refreshPage()">Refresh The result</button>

</div>

<script src="{{ url\_for('static', filename='script.js') }}"></script>

</body>

</html>

```

## 4.1.2 styles.css

```css

:root {

--common-width: 95%;

}

body {

font-family: Arial, sans-serif;

background-color: 4b4646;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

margin: 0;

}

.container {

background-color: fff;

padding: 5%;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: auto;

text-align: center;

}

textarea, button, output {

width: var(--common-width);

margin: auto;

}

textarea {

height: 100px;

margin-bottom: 10px;

padding: 10px;

border: 1px solid ccc;

border-radius: 5px;

}

button {

padding: 10px;

width: 60%;

margin-bottom: 20px;

border: 1px solid rgb(12, 12, 12);

border-radius: 15px;

background-color: 007BFF;

color: ffffff;

font-size: 16px;

cursor: pointer;

}

button:disabled {

background-color: ccc;

cursor: not-allowed;

}

output {

margin-top: 20px;

margin-bottom: 20px;

width: 90%;

padding: 20px;

border: 1px solid a7a6a6;

border-radius: 5px;

background-color: f9f9f9;

text-align: center;

font-weight: bold;

}

```

## 4.1.3 script.js

```javascript

function checkEmotion() {

const inputText = document.getElementById('inputText').value;

console.log('Function called');

console.log('Input text', inputText);

if (inputText.trim() === '') {

alert('Please enter some text.');

return;

}

// Simulate emotion analysis (replace this with actual model integration)

const emotions = ['anger', 'disgust', 'fear', 'joy', 'neutral', 'sadness', 'surprise'];

const randomEmotion = emotions[Math.floor(Math.random() \* emotions.length)];

document.getElementById('output').innerText = `Detected Emotion: ${randomEmotion}`;

}

function refreshPage() {

document.getElementById('inputText').value = '';

document.getElementById('output').innerText = '';

}

```

## 4.2 Back-end

The back-end is implemented using Flask. The main application file, `app.py`, handles the routing and interaction with the sentiment analysis model.

## 4.2.1 app.py

```python

from flask import Flask, render\_template, request, jsonify

from transformers import pipeline

app = Flask(\_\_name\_\_)

Load the sentiment analysis model

classifier = pipeline('sentiment-analysis')

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/analyze', methods=['POST'])

def analyze():

text = request.json['text']

result = classifier(text)[0]

return jsonify(result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

## 4.3 Model

The sentiment analysis model used is a pre-trained DistilBERT model from the Hugging Face library. It is trained using GoEmotions Dataset and saved on google drive. It need to download and saved to model folder before run the app

# 5. Results and Discussion

## 5.1 Performance

The application successfully processes user input and returns sentiment analysis results. The pre-trained DistilBERT model provides accurate and reliable sentiment classification.

## 5.2 User Experience

Users find the interface intuitive and easy to use. The real-time feedback from the sentiment analysis model enhances user engagement.

# 6. Challenges and Solutions

6.1 Integration

\*\*Challenge\*\*: Integrating the DistilBERT model with the Flask application.

\*\*Solution\*\*: Using the Hugging Face `pipeline` API simplified the integration process.

6.2 Front-end Design

\*\*Challenge\*\*: Designing a responsive and user-friendly interface.

\*\*Solution\*\*: Utilizing CSS flexbox for layout and ensuring compatibility with various screen sizes.

# 7. Future Work

- Improve the model's accuracy by fine-tuning with domain-specific data.

- Add support for multiple languages.

- Implement a more sophisticated user interface with additional features such as sentiment graphs and historical data analysis.

# 8. Conclusion

The Sentiment Analysis Web App is a functional and user-friendly tool for performing sentiment analysis on text inputs. It leverages the power of pre-trained NLP models to deliver accurate and meaningful results. This project demonstrates the practical application of machine learning in web-based solutions.

# 9. References

- Hugging Face Transformers: https://huggingface.co/transformers/

- Flask Documentation: https://flask.palletsprojects.com/

- CSS Flexbox: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\_Flexible\_Box\_Layout