TEXT EMOTION DETECTION USING NLP AND STREAMLIT

NLP FINAL PROJECT REPORT

MILKESA KUMERA

GSE/1533/16

1. Introduction

In today's digital world, understanding the emotional context of written communication has become increasingly important. Emotional analysis from text is widely used in customer service, marketing, mental health, and social media monitoring.

This project, titled **Text based Emotion Detection**, leverages Natural Language Processing (NLP) techniques to detect emotions in user-inputted text. It is implemented in Python and deployed using **Streamlit**, an interactive web app framework. A machine learning pipeline trained on labeled emotion datasets is used in the backend to perform text classification into emotions such as

- ⇨ Joy
- ⇨ Anger
- ⇨ Sadness
- ⇨ Disgust
- ⇨ Fear
- ⇨ Love
- ⇨ Neutral
- ⇨ Shame
- ⇨ Surprise

2. Project Objective

The main objective of this project is to accurately classify text input into one of the predefined emotional categories and visualize the prediction probabilities. It serves as a demonstration of practical NLP applications and real-time emotion analysis through a user-friendly web interface.

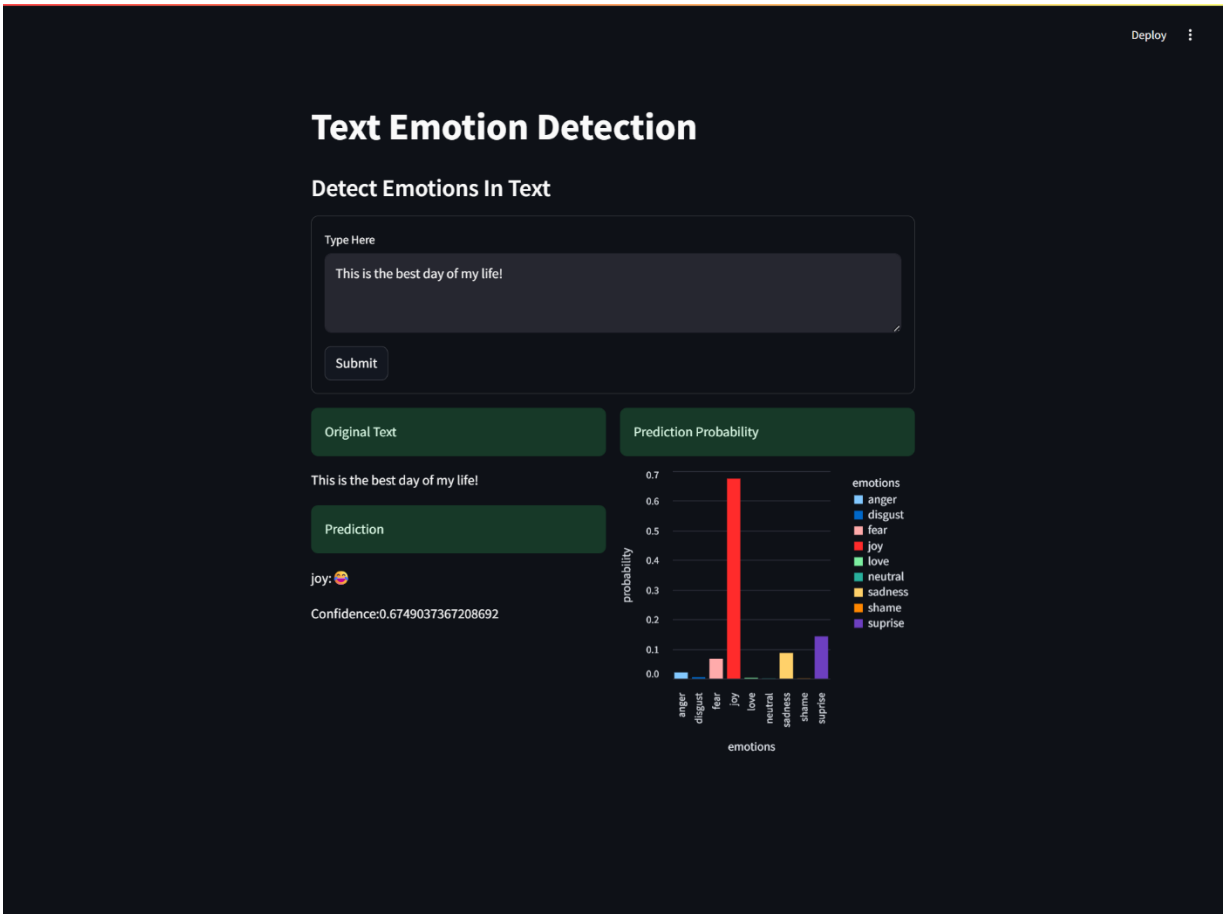
This model is beneficial in:

- Analyzing user sentiments from feedback forms.
- Understanding emotions expressed on social media.
- Supporting mental health monitoring and analysis.

3. Tools and Technologies

No	Technology	Description
1	Python	Core programming language
2	Pandas	Data analysis and manipulation
3	NumPy	Numerical operations
4	Scikit-learn	Model training, evaluation, and pipelines
5	Joblib	Model serialization for reuse
6	Altair	Data visualization (probability chart)
7	Streamlit	Web application interface

4. Application Workflow



The application operates through a streamlined workflow:

1. **User Input**: A form where the user enters a text message.
2. **Emotion Prediction**: The input is processed by a pre-trained model that classifies the emotion.
3. **Confidence Visualization**: A probability score for each emotion is displayed in a bar chart.
4. **Emoji Output**: An emoji is shown based on the predicted emotion to enhance user engagement.

5. Model Overview

The model used in this project is trained offline and saved using `joblib`. It follows this standard NLP model pipeline:

- **Text Preprocessing**: Removing noise and tokenizing text.
- **Vectorization**: Converting text to numeric format using CountVectorizer or TF-IDF.
- **Model Training**: Using a classifier such as Logistic Regression. In this experiment LR showed better result efficiency compared to SVM and random forest classifier.
- **Model Evaluation**: Validating with accuracy, and confusion matrix.
- **Serialization**: Saving the model for use in the Streamlit application.

The serialized model (`text_emotion_detection.pkl`) is then loaded in the app and used to make real-time predictions.

6. Streamlit App Functionality

The `app.py` script contains the logic for the web interface. Below is a summary of how it works:

- The model is loaded:

```
python
CopyEdit
pipe_lr = joblib.load(open("model/text_emotion.pkl", "rb"))
```

- The app includes a text box for user input and a submit button:

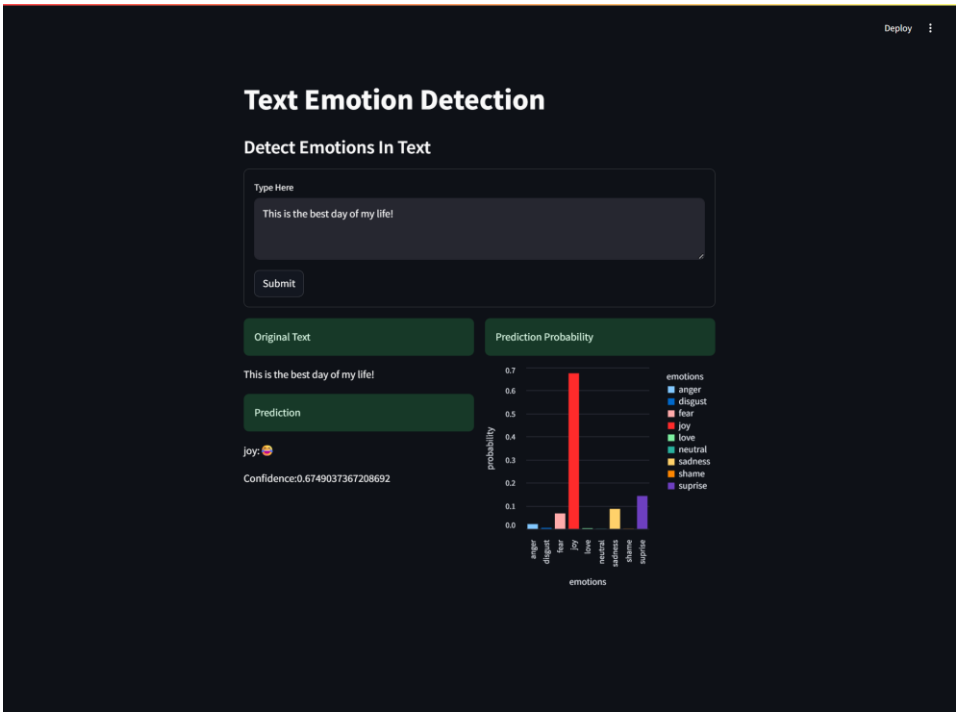
```
python
CopyEdit
raw_text = st.text_area("Type Here")
submit_text = st.form_submit_button(label='Submit')
```

- Once the text is submitted, the emotion prediction and confidence levels are shown, along with a probability chart:

```
python
CopyEdit
prediction = predict_emotions(raw_text)
probability = get_prediction_proba(raw_text)
```

- An emoji is displayed alongside the predicted emotion, and a bar chart shows the prediction confidence across all emotion classes using **Altair**.

7. Screenshots of the Application



8. Example Result

Sample Input:

```
Plaintext : "This is the best day of my life"
```

Predicted Emotion:

joy 😂

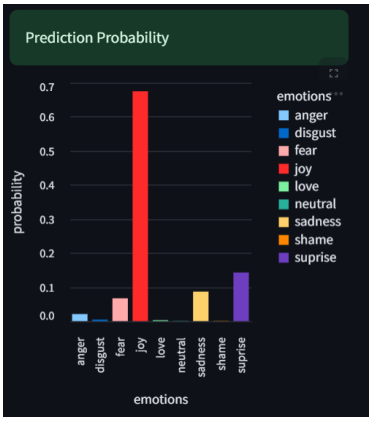
Confidence Score:

0.6749

Prediction Probability Table:

No	Emotion	Probability
1	Anger	0.02
2	Disgust	0.0044
3	fear	0.0668
4	joy	0.6749
5	Love	0.002492
6	Neutral	0.000728
7	Sadness	0.08656
8	Shame	0.000728
9	Surprise	0.1427

A bar chart is displayed visually showing the probabilities for each emotion.



9. Project Limitations

While the application works well in general use cases, it is subject to a few limitations:

- **Dataset Bias**: The model is only as good as the dataset it's trained on. The dataset used has huge imbalance
- **Nuanced Emotion Detection**: Complex emotions, sarcasm, or mixed emotions may not be accurately captured.
- **Static Model**: The current model is static and does not improve with usage unless retrained.
- **Limited Emotion Categories**: Only predefined emotion classes are available. There is no option for multi-label or compound emotions.

10. Future Work and Improvements

Potential improvements and enhancements include:

- **Upgrade to Transformer Models**: Use of models like BERT or RoBERTa for deeper contextual understanding.
- **Live Feedback Loop**: Allow users to correct predictions, thereby enabling incremental learning.
- **Multilingual Support**: Extend the app to support multiple languages.
- **Mobile Responsiveness**: Improve UI/UX for mobile devices.
- **Deployment Online**: Host the application publicly on platforms like Streamlit Cloud or Heroku.

11. Conclusion

The Text Emotion Detection project successfully demonstrates the application of machine learning and NLP techniques in detecting emotions from text. By integrating a pre-trained model with a user-friendly Streamlit interface, it bridges the gap between data science and end-user interaction.

With further development and model improvements, this application can be used in a wide range of real-world scenarios, including sentiment analysis, social media monitoring, and even healthcare applications.