Introduction

- Background

In recent years, chatbots have become an integral part of user interaction on a wide variety of platforms, ranging from customer support to personal assistants. Sentiment analysis, a crucial branch of natural language processing (NLP), enables systems to detect emotional tone in text, enhancing the user experience by allowing the chatbot to respond more appropriately to different emotions.

- Project Overview

This project aims to develop a chatbot that can analyse the sentiment behind user inputs, classifying them as either positive, neutral, or negative. By using machine learning and NLP techniques, the chatbot becomes more context aware and responsive to emotional states. The key element of this chatbot is the integration of a custom trained **Roberta** model for sentiment analysis, which helps in accurately identifying the sentiment of the user's input.

Objectives

- Sentiment Classification: The chatbot's primary function is to classify user input into positive, neutral, or negative sentiment. This helps the chatbot provide more contextually appropriate responses, enhancing user interaction. By understanding the emotional tone of the message, the system can tailor its reactions, such as offering support for negative inputs or acknowledging positive ones.
- Model Integration: The project uses a finetuned Roberta model for precise sentiment detection. Roberta is known for its strong performance in language understanding, and finetuning it on specific datasets allows for accurate classification of user input. Once trained, the model is seamlessly integrated into the chatbot, enabling real time sentiment analysis.
- NLP Utilization: Natural Language Processing (NLP) techniques are applied to preprocess user input before sentiment analysis. This includes tokenization (breaking the text into smaller parts) and stop word removal (eliminating unnecessary words like "and" or "the"). These preprocessing steps ensure the text is ready for the model to analyze efficiently.
- User Interface: The chatbot features an interactive user interface developed using Streamlit. Users can input text and receive sentiment analysis results instantly. The simple design ensures ease of use, allowing users to analyze their messages and view the classification (positive, neutral, or negative) in real time with just a few clicks.

System Overview

Architecture Overview

- **User Interface**: A simple web interface created using Stream-lit where users can input text for sentiment analysis.
- **NLP Pipeline:** The system preprocesses user input using NLP techniques such as tokenization and normalization.
- **Sentiment Analysis Model:** A custom Roberta model is employed to analyse the sentiment and categorize the user's input.
- **Result Display:** The system then returns the result in terms of sentiment and displays it on the user interface.

System Flow

- **Input Stage:** The user inputs text into the Stream-lit interface.
- **Text Preprocessing:** The input text undergoes tokenization and other necessary preprocessing steps using NLP techniques.
- **Sentiment Analysis:** The custom trained Roberta model analyses the processed text.
- **Output Stage:** The sentiment of the input text (positive, neutral, or negative) is displayed in the UI.

Natural Language Processing (NLP)

Overview

- Natural language processing is crucial for the chatbot's ability to understand and process human language.
- In this project, several NLP techniques are applied to prepare the text for sentiment analysis.

Key Components

- **Tokenization:** Breaking down the input text into smaller tokens (such as words or sub word) to be processed by the model.
- **Stop-word Removal:** Removing common words (like "the," "is," etc.) that do not contribute to sentiment detection.
- **Label Mapping:** The predicted labels from the model are mapped to human readable sentiment categories: Negative, Neutral, and Positive.

Machine Learning Model

Roberta Model

- The model used for sentiment analysis in this project is a finetuned version of Roberta for Sequence Classification.
- Roberta is an extension of the BERT model, optimized for sequence classification tasks. Its robust architecture allows for better performance in understanding the context of long texts.

Training and Classification

- The custom Roberta model is trained on a large dataset to classify text into three categories:

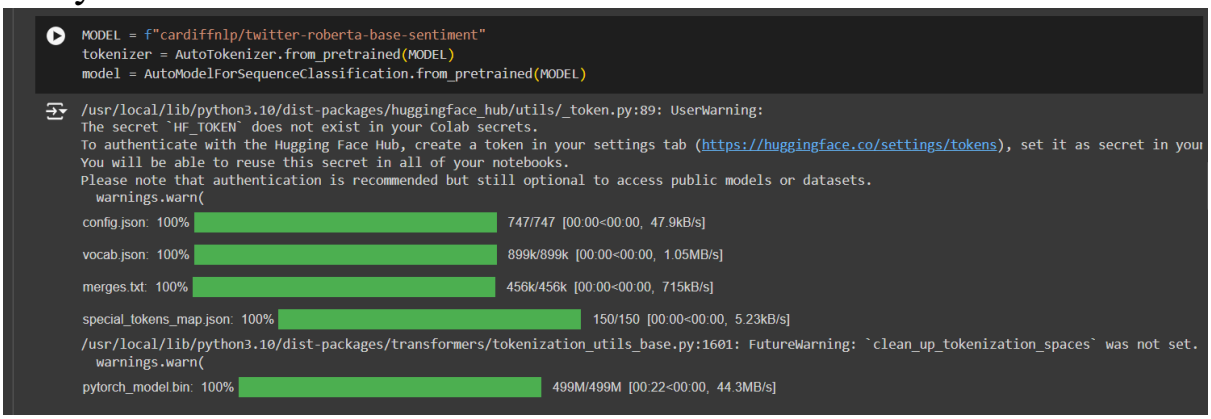
  LABEL_0: Negative Sentiment

  LABEL_1: Neutral Sentiment

  LABEL_2: Positive Sentiment

Model Pipeline

- The model and tokenizer are loaded from a pretrained directory.
- The pipeline is then established to perform the sentiment analysis.



System Design and Implementation

Technologies Used:

- Programming Language: Python
- Libraries Used:
  - ➢ Transformers (Hugging Face): For the implementation of the Roberta model.
  - ➢ Stream-lit: For developing the user interface and interactivity.
  - ➢ Pandas, NumPy, Seaborn, Matplotlib: For handling data and visualization where necessary.

Code Implementation :

```python
import streamlit as st
from transformers import RobertaForSequenceClassification, RobertaTokenizer, pipeline

# Load the saved model and tokenizer
@st.cache_resource
def load_local_model():
    # Load the tokenizer and model from your saved directory
    tokenizer = RobertaTokenizer.from_pretrained("C:\\Users\\thaku\\Desktop\\Teachnook_AI\\hit_and_trail\\roberta_saved_model")
    model = RobertaForSequenceClassification.from_pretrained("C:\\Users\\thaku\\Desktop\\Teachnook_AI\\hit_and_trail\\roberta_saved_model")
    return pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)

# Map the model's label output to human-readable sentiments
label_mapping = {
    "LABEL_0": "NEGATIVE",
    "LABEL_1": "NEUTRAL",
    "LABEL_2": "POSITIVE"
}

# Function to perform sentiment analysis and map the output labels
def analyze_sentiment(text):
    model_pipeline = load_local_model()
    raw_result = model_pipeline(text)

    # Map the label to POSITIVE, NEUTRAL, or NEGATIVE
    for result in raw_result:
        result["label"] = label_mapping.get(result["label"], result["label"])  # Map label

    return raw_result

# Streamlit UI
st.title("ROBERT Sentiment Analysis (Custom Model)")
st.write("Enter a sentence to analyze its sentiment using your custom-trained model.")

# Text input from the user
user_input = st.text_area("Input Text", "")

# Analyze button
if st.button("Analyze"):
    if user_input:
        result = analyze_sentiment(user_input)
        st.write("Sentiment Analysis Result:", result)
    else:
        st.write("Please enter some text for analysis.")
```

Challenges and Solutions

Challenges

- **Model Accuracy:** Ensuring the accuracy of the Roberta model for detecting nuanced sentiments.
- **Handling Sarcasm:** The model struggles with understanding sarcasm and complex emotional cues.
- **Latency in Real Time:** Optimizing the model to provide quick responses without much delay.

Solutions

- **Model Fine Tuning:** The model was further finetuned with additional data to improve accuracy.
- **Context Awareness:** Introducing context sensitive models can be a future solution to handle sarcasm better.
- **Optimization for Speed:** Modifications were made to streamline the real time performance and reduce latency.

Results and Discussion

- Sentiment Analysis Results: The chatbot successfully classifies user inputs into one of three sentiment categories: Positive, Neutral, or Negative. The performance was tested using a variety of sentences, and the model showed reliable accuracy in most cases. However, there is still room for improvement in cases of ambiguous or sarcastic input.

- Discussion: The integration of a custom Roberta model with an NLP pipeline and a Stream-lit user interface has proven effective in building a sentiment analysis chatbot. The system provides a user-friendly experience with relatively accurate sentiment classification. There are some limitations in more complex emotional scenarios that will be explored in future versions.

Conclusion

- This project demonstrates how natural language processing (NLP) and machine learning techniques can be applied to build a chatbot capable of analysing user sentiment.
- The custom trained Roberta model successfully classifies user input into positive, negative, or neutral sentiment.
- Future developments may include enhancements in sarcasm detection, more context awareness, and expanding the chatbot's capabilities to understand additional emotional states.
