Sentiment Analysis on Call Transcripts

1. Introduction

1.1 Project Overview

This project focuses on performing sentiment analysis on call transcripts between sales agents and customers. The goal is to evaluate the overall sentiment of the conversation by analyzing individual statements from both parties and computing the overall sentiment score.

1.2 Problem Statement

Customer service and sales calls are crucial for businesses. Analyzing the tone and sentiment of these conversations can provide valuable insights into customer satisfaction and the effectiveness of sales strategies. The challenge is to process call transcripts automatically and classify sentiments as **positive**, **neutral**, or **negative**, while also computing an overall sentiment score for the conversation.

1.3 Objectives

- Build a **Streamlit-based user interface** to upload call transcripts as `.txt` files.
- Develop a **Flask backend** to handle the uploaded files and process sentiment analysis.
- Perform sentiment analysis using pre-trained NLP models from Hugging Face.
- Provide detailed sentiment analysis results for each speaker and compute the overall sentiment score of the conversation.
- Deploy the application for easy accessibility and usability.

2. Methodology

2.1 Tools and Technologies

- **Frontend:** Streamlit (Python-based UI framework)
- **Backend:** Flask (Lightweight Python web framework)
- **NLP Model:** Hugging Face `cardiffnlp/twitter-roberta-base-sentiment-latest` (for sentiment analysis)
- **Deployment:** Local environment, with the potential for deployment on platforms like Vercel.

- **Other Libraries:**
  - `transformers` for loading pre-trained NLP models.
  - `torch` for PyTorch backend support.
  - `requests` for sending data between **Streamlit** and **Flask**.

2.2 Project Workflow

1. **Frontend (Streamlit App):**
   - Users can upload `.txt` files containing call transcripts.
   - Display the uploaded file content for preview.
   - Send the uploaded file to the Flask backend for processing when the "**Analyze Sentiment**" button is clicked.
2. **Backend (Flask App):**
   - Receive the uploaded file via a POST request.
   - Save the file to an `uploads` directory for processing.
   - Read and parse the file to extract individual statements for each speaker.
   - Perform sentiment analysis on each statement using a pre-trained NLP model.
   - Compute an overall sentiment score for the conversation.
   - Return a JSON response containing:
     - Sentiments for individual statements by the sales agent and customer.
     - Sentiment lists for both speakers.
     - Overall sentiment score and classification.
3. **Output:**
   - Display sentiment analysis results **overall sentiment** score, in the **Streamlit** app.

3. Implementation

3.1 Preprocessing

- **Transcript Parsing:**
  - The call transcript is expected in a specific format:

```
[Sales Agent 00:03]
Hello, good afternoon. I hope I am speaking to Mr. Anmol Singh.

[Customer 00:26]
You are calling from which institute?
```

  - The `preprocess_transcript` function processes the transcript to extract statements for each speaker (`Sales Agent` and `Customer`).

3.2 Sentiment Analysis

- **Model Used:**
  - `cardiffnlp/twitter-roberta-base-sentiment-latest`, a pre-trained model from Hugging Face.
  - The model classifies text into three sentiment labels: `positive`, `neutral`, and `negative`.
- **Pipeline Integration:**
  - Used Hugging Face's `pipeline` to load the sentiment analysis model.
  - Each statement is analyzed individually, and the sentiment label and score are recorded.

3.3 Computing Overall Sentiment

- **Sentiment Weights:**
  - Positive: `+1`
  - Neutral: `0`
  - Negative: `-1`
- **Scoring:**
  - For each speaker, calculate the sum of the weights of their sentiments.
  - Compute the total score by combining the scores of both speakers.
  - Normalize the score by dividing it by the total number of statements.
- **Classification:**
  - If the normalized score > 0 → Overall sentiment is **`positive`**.
  - If the normalized score < 0 → Overall sentiment is **`negative`**.
  - If the normalized score == 0 → Overall sentiment is **`neutral`**.

3.4 System Design

- **Frontend (Streamlit):**
  - `st.file_uploader`: Allows users to upload `.txt` files.
  - `requests.post`: Sends the uploaded file to the Flask backend.
  - Results are displayed in JSON format, along with a preview of the transcript.
- **Backend (Flask):**
  - Endpoint: `/analyze`
  - Handles file upload, saves the file, and processes it for sentiment analysis.
  - Returns detailed sentiment analysis results as a JSON response.

4. Challenges and Solutions

1. **Handling Empty or Incorrect Files:**
    - Added validation to check if the uploaded file is empty or not in the expected format.
2. **File Upload Issues in Streamlit:**
    - Resolved the issue of empty files by resetting the file pointer using `.seek(0)` before sending it to the backend.
3. **Parsing Multiline Statements:**
    - Updated the parsing logic to handle multiline statements for better accuracy.
4. **Deployment Challenges:**
    - Ensured the app works locally first. Deployment on platforms like Vercel or Heroku can be done in future iterations.

5. Conclusion

5.1 Summary

The project successfully implements a system for analyzing call transcripts, providing:

- Individual sentiment analysis for statements by the sales agent and customer.
- A normalized overall sentiment score for the entire conversation.

5.2 Key Features

- User-friendly interface for uploading transcripts.
- Detailed and accurate sentiment analysis using a state-of-the-art pre-trained model.
- Robust handling of input files and comprehensive error handling.

5.3 Future Enhancements

- **Additional Sentiment Labels:**
    - Incorporate granular sentiment labels (e.g., "very positive," "slightly negative").
- **Support for Other Formats:**
    - Extend functionality to accept `.csv` or `.pdf` files.
- **Real-Time Processing:**
    - Analyze live conversations or streamed data.
- **Visualization:**
    - Add charts and graphs to visualize sentiment trends.

6. References

1. Hugging Face Model: `cardiffnlp/twitter-roberta-base-sentiment-latest`
2. Streamlit Documentation: Streamlit File Uploader
3. Flask Documentation: Flask File Uploads