Deep Learning Applications

TITLE: Sentiment Analysis of Audio files using BERT (Bidirectional Encoder Representations from Transformers)

NAME: ANJU MULLAKKARAMALIL SUKUMARAN NAIR

STUDENT ID: MUL23618197

ABSTRACT

This project gives everything from audio processing and speech-to-text transcription to the sentiment analysis of texts all in one coherent and working application, which advanced deep learning model's power. Using OpenAI's Whisper for accurate speech recognition and BERT for reliable sentiment analysis, the system will enable users to record audio, transcribe this into English, and further assess the sentiment of the text. This application, with an intuitively designed Streamlit interface, simplifies the workflow of audio-based sentiment analysis and makes it user-friendly and effective. It will be an ideal solution for any application that requires customer feedback analysis, voice assistants, and real-time tracking of sentiment that will offer actionable insight from audio data.

INTRODUCTION

It makes a real-time audio-based sentiment analysis application using deep learning techniques that integrates speech to text transcription and sentiment classification. The application enables a user to record an audio, transcribe it to the English text, and find sentiment associated with the transcription using a combination of OpenAI Whisper model for speech recognition, and BERT for the sentiment analysis. These deep learning models for advanced speech help automate spoken language into actionable insights, such as determining the emotional tone of customer feedback, social media conversations, or voice assistant interactions.

The main objective of the project is to make the availability of sentiment analysis easier and faster by automating the whole process of transcription from audio to text and then doing sentiment classification within one smooth workflow. The importance of this project is that it connects the bridge of deep learning to audio data meaningful analysis and, simultaneously provides real-time sentiment insights through speech in customer service, content moderation, market research, amongst many others. This whole system is developed using Streamlit that allows users to have an instinctive user-friendly graphical interface. It easily interacts with the tool and records voices while showing instantaneous results without prior advanced technical skill.

This project will showcase how deep learning models, such as Whisper and BERT combined, can solve real-world problems. The model shows the power of such models in transforming audio input into relevant insights. It thereby provides an efficient and effective solution to businesses and industries seeking to understand and respond to spoken language in real time through automated transcription and sentiment analysis.

METHODOLOGY

This project will integrate two powerful deep learning models, Whisper and BERT, in developing an end-to-end audio-based sentiment analysis system. A detailed explanation of the algorithms and techniques used in this model is given below.

1. Whisper Model for Speech Recognition

This approach shall apply the Whisper model designed by OpenAI for converting speech to text: a deep neural network structure trained on a large, multilingual, multitask dataset.. The Whisper model executes robust transcription, therefore remaining ideal for speech of various natures: multiple languages and speaking with background noise, amongst others. This qualifies the approach for generalized speech recognition challenges.

Algorithm: Whisper is a single neural network model, end-to-end, that takes raw audio input; it does not have any pre-processing steps, such as feature extraction of the input. It was trained on vast multilingual data and ensures the highest possible accuracy in speech recognition for diverse languages and accents.

Language Detection: Automatic detection of the language spoken in the audio by the model ensures that transcriptions are accurate even when users speak in different languages. In our use case, the model is configured to transcribe audio to English.

Noise Resilience: The architecture of Whisper is resilient to noise; hence, it works well even in poor recording environments.
In the context of this project, Whisper has to transcribe the audio input into text, which then gets passed to the sentiment analysis component for further processing.

2. BERT Model for Sentiment Analysis

The text obtained from audio transcription will then be processed for sentiment analysis. For the purpose of sentiment analysis in this project, the pre-trained transformer-based model BERT will be used; it has already been pre-trained for many NLP tasks, including the classification of sentiment.

Algorithm: Transformer Architecture: BERT uses a transformer architecture. The self-attention mechanism in the model learns the relationships among words in a sentence for context. It provides the ability for BERT to capture such nuances of language-words having meaning based on context, which is one very crucial point in tasks related to sentiment analysis.

Pretrained Models: BERT learns common structures of language in general by pre-training on the largest corpus of text such as Wikipedia and BooksCorpus, and fine-tunes itself on a specific dataset containing for instance sentiment-labeled data so that it can gain sentiment classification capabilities.

Bidirectional Context: While most models conventionally read the text from left to right, BERT does it in a bidirectional way; it reads the whole sentence both ways. That helps it understand the full context of a word concerning the entire sentence and can therefore interpret the sentiment even better.

Sentiment Analysis Fine-tuning: In the presented project, fine-tuning of BERT has been carried out on a labeled dataset of text samples, together with their corresponding sentiment labels. Fine-tuning changes the model weights to perform optimally on the sentiment classification task.

Process: Transcription is tokenized; it is broken down into smaller units known as tokens. A BERT tokenizer will split the text into tokens understandable by the model. It adds special tokens for sentence beginning and end identification. Encoding of Input: Further, tokenized text is encoded into numerical input embeddings which BERT can read. These embeddings represent words in a vector space and capture semantic relationships between words.

Model Inference: BERT takes this encoded input and runs it through the layers of attention mechanisms, which yield the contextualized embeddings for every word. These are used as input to make a prediction about the sentiment of the text.

Sentiment Classification: The output from BERT's layers is fed to a classification head, usually a dense layer followed by softmax. Softmax gives a probability distribution over all sentiment labels, and the one with the highest probability is considered the class of the sentiment prediction.

3. Label Encoding

For conversion to human-readable labels, such as positive, negative, or neutral, the system uses Scikit-learn's LabelEncoder. It maps numerical output from BERT's classification head back into the corresponding sentiment labels.

Label training: The LabelEncoder trains from a labeled dataset; an example could be labels.csv with text examples and the respective sentiment labels.

Prediction: After BERT has produced the probabilities of each sentiment class, LabelEncoder maps the predicted numerical class to its corresponding label, such as 0 to "positive", 1 to "negative", and 2 to "neutral".

4. Integrating into Streamlit for Real-Time Interaction

This project encompasses the entire process chain from recording an audio and transcription up to sentiment analysis within one Streamlit interface.This GUI-driven approach allows users to interact in real time with the system. Users can record their audio using this interface; the system then processes the audio as described here.

Audio Recording: A user records audio with the help of the sounddevice library. Audio, which is to be recorded, comes out as a.wav file.

Transcription of Audio: The audio stored gets taken by the Whisper Model for transcription. Sentiment Analysis: This transcribed text provides input for the BERT Model to make a prediction as to sentiment, which would be presented on the UI. This whole end-to-

end workflow provides an efficient experience for the users in order to perform sentiment analysis right from the spoken language.

5. Challenges and Considerations

Noisy and ambiguous speech is one of the major barriers in speech recognition: noisy environment, accents, and unclear talking. Whisper has a robust noise immunity that will be of great help in regards to this, but transcription quality may still depend on other external factors.

Model Latency: This is the total time it takes to transcribe audio and process sentiment, and it will be a bottleneck for longer audio inputs. Optimizations in Whisper's processing or leveraging hardware acceleration could help speed up the process of inference.

Sentiment Classification Accuracy: While BERT is a very strong model, sentiment classification will always be somewhat subjective. Fine-tuning with a diverse dataset that covers all eventualities is crucial to improving the accuracy.

CODE DOCUMENTATION

The project integrates a variety of state-of-the-art machine learning techniques and user interface technologies in order to perform sentiment analysis from audio input. It is described in detail below by key functions and components.

1. Audio Recording and Processing

Functionality: The application captures real-time audio from the user's microphone and processes it for transcription.

Key Components:

SoundDevice Library: Used for audio recording, ensuring compatibility across devices.

Audio Processing: The recorded audio is normalized and saved as a WAV file to ensure compatibility with the Whisper model.

Code Highlights:

record_audio(duration, sample_rate): Captures audio for a specified duration.

save_audio(audio, filename, sample_rate): Saves the audio to a file in a format ready for transcription.

2. Speech-to-Text Conversion

Functionality: Converts the recorded audio into text using OpenAI's Whisper model. This enables the downstream task of sentiment analysis.

Key Components:

Whisper Model: A pre-trained speech-to-text model that supports multiple languages and achieves high accuracy.

Preprocessing: Ensures audio is in the correct format for Whisper to process efficiently.

Code Highlights:

transcribe_audio(filename):

Loads the Whisper model (whisper.load_model).

Processes the audio file to extract text using the transcribe function of the model.

3. Sentiment Analysis

Functionality:

Determines the sentiment (Positive, Negative, or Neutral) of the transcribed text using a fine-tuned BERT model.

Key Components:

BERT Model: Pre-trained on general language tasks and fine-tuned for sentiment classification.

Tokenizer: Converts text into tokens that the BERT model can process.

Label Encoder: Maps predicted labels back to human-readable sentiment categories.

Code Highlights:

load_model(model_name):

Loads the BERT model and tokenizer from the specified directory.

predict_sentiment(text):

Tokenizes the text using the BERT tokenizer.

Passes the tokens to the BERT model for sentiment prediction.

Maps the predicted label to a sentiment using the label encoder.

User Interface

Functionality:

Provides a user-friendly interface to interact with the application, record audio, view transcription results, and analyze sentiments.

Key Components:

Streamlit: A Python-based framework for building web applications.

Interactive Widgets: Enables users to select recording duration, start recording, and view results interactively.

Code Highlights:

st.slider: Lets the user choose the duration of the audio recording.

st.button: Triggers the recording and analysis workflow.

st.audio: Allows the user to play back the recorded audio.

st.success, st.info, and st.write: Provide real-time updates and display results.

Model Deployment and Optimization

Functionality:

Ensures the application runs efficiently, even on resource-constrained devices.

Key Components:

Whisper on CPU: The model is optimized for CPU use, but future versions may incorporate GPU acceleration.

Caching and Pre-loading Models: Reduces latency by loading the Whisper and BERT models once and reusing them for multiple inferences.

DEPLOYMENT

Setting Up the Environment

Install Python: Ensure Python (version 3.8 or above) is installed on your system. You can download it from python.org.

Create a Virtual Environment:

python -m venv sentiment_env

source sentiment_env/bin/activate # On Windows: sentiment_env\Scripts\activate

Install Required Libraries: Install the necessary dependencies by running:

pip install -r requirements.txt

```
import streamlit as st
import sounddevice as sd
import numpy as np
import whisper
from scipy.io.wavfile import write
from bert import predict_sentiment
import sys
from io import StringIO
```

```
from sklearn.preprocessing import LabelEncoder
from transformers import BertTokenizer, BertForSequenceClassification
import pandas as pd
```

Download Pretrained Models:

Ensure that the sentiment_model directory (containing the fine-tuned BERT model) is present in the project folder.

The Whisper model will automatically download during the first run.

Prepare the Labels File:

Ensure the labels.csv file (containing sentiment classes) is in the project directory.

The file must have a column named Class listing all sentiment categories.

2. Running the Application

Start the Streamlit App: Run the following command in your terminal:

streamlit run main.py

Access the Application:

After running the command, Streamlit will provide a local URL (e.g., http://localhost:8501).

Open the URL in web browser to access the app interface.

3. Interacting with the Application

Step 1: Select Recording Duration

Use the slider on the Streamlit interface to select the duration of the audio recording (1 to 10 seconds).

Step 2: Start Recording

Click the "Start Recording" button to capture audio through device's microphone.

The app will display a message indicating the recording duration and then process the recorded audio.

Step 3: Playback Recorded Audio

After recording, the audio is saved as a file (recorded_audio.wav) and made available for playback in the app.

Click the play button to review the recording.

Step 4: Transcription

The Whisper model processes the audio file and generates a text transcription.

The transcription is displayed on the interface as a success message.

Step 5: Sentiment Analysis

Once the transcription is complete, the BERT model analyzes the sentiment of the transcribed text.

The sentiment result (e.g., Positive, Negative, Neutral) is displayed on the interface.

Sentiment analysis of audio using BERT

Select recording duration (seconds)

5

1 10

Start Recording

Recording for 5 seconds...

▶ 0:00 / 0:05

Transcribing audio...

Transcription completed. Transcribed text is `The flower is very beautiful.`

Analysing sentiment....

Sentiment from the recorded audio is

value
Positive

sentiment analysis is completed

RESULTS AND ANALYSIS

The initial goal of this project was to design a program that could analyze sentiment from live speech recordings by incorporating Whisper into text with BERT for the analyses of sentiment. Testing

In the testing process of the application, it was pushed under strenuous testing in different various environments to check the ability, accuracy, and efficiency with which it would work.

Test Results

Speech to Text Accuracy:

The Whisper model, on the other hand, proved to be quite promising for the transcription of audio into text. For clear and high-quality audio, the transcription was quite accurate, having a very high word recognition rate. In cases with noisy backgrounds or unclear speech, the transcription accuracy was compromised. The model could still provide a coherent transcription but with some errors, especially for words of similar phonetics or in cases where there was too much background noise.

Sentiment Analysis Performance: The BERT-driven sentiment analysis component performed admirably regarding the categorization of texts into one of three classes of sentiment-Positive, Negative, and Neutral. Generally, sentiment predictions were observed to follow the context in which transcriptions were given. If the transcription of the text carried a positive attitude, it was rightly predicted as "positive". While this has been the exact opposite sentiment classification that fared well, it oftentimes struggled a bit in cases with more ambiguous and sarcastic contexts. This, however, can be based on the shortcomings of BERT in catching sarcasm or subtle contexts that may, at some point, capture phrases.

The user interface was developed using Streamlit, which is pretty intuitive and friendly for usage. The feature worked perfectly in recording audio, both for recording and playing the audio. Transcription and sentiment analysis were done in pretty good time, output coming within a few seconds. Therefore, this application is more interactive and responsive, fulfilling the intention of real-time analysis.

User Experience: The user interface, developed in Streamlit, was pretty intuitive and straightforward to use. The audio recording feature was working as anticipated: it allowed the user to record and then play the audio without any issues. The transcription and sentiment

analysis were run in a decent amount of time; results were available in a few seconds. This made the application more interactive and responsive for users hence meeting the requirement of real-time analysis.

Challenges Faced

Audio Quality Issues:

Testing have raised some challenges, including consistent audio quality. This ranged from background noise to low recording volumes and unclear speech. Whisper does well with good quality audio, but its accuracy degrades in case of issues with the recording environment. For instance, transcription errors increased when either noisy conditions or overlapping speech occurred.

Solution:

Further did some testing in rather quieter environments and with different microphones. Employing noise cancellation techniques, applying audio pre-processing are other options through those could be tried out to improve transcription accuracy.

Limitations in Performance on CPU:

Both Whisper and BERT models, when run on the CPU, had faced a much slower processing time, especially for big audio files or if multiple requests were coming in real-time.

Solution:

Here also tried to use a GPU for inference to increase performance. This reduced the processing time of transcription and sentiment analysis significantly, hence making the working of the application smooth and user-friendly. In cases where a GPU is not available, it suggested optimizing the application to handle shorter audio clips or providing an option to select the quality and length of the audio recordings.

Complex Sentiment Handling:

While the BERT-based sentiment analysis model performed really well for general sentiments, it did not do as well with more subtle emotional nuances, such as sarcasm, irony, or ambiguous statements.

Solution:

One way to overcome this limitation is by fine-tuning the BERT model on a more diverse and contextually complex dataset, including examples of sarcastic or ambiguous language

Besides, integrating models specifically designed for emotion detection or sentiment analysis of nuanced language could improve the overall accuracy of the system in these areas.

Strategies Followed for Addressing the Challenges

Preprocessing of Audio Data:

Due to noisy environments and incomprehensible speakers, used some minimum processing steps like normalizing volume levels and filtering low frequency that would help in providing noise and hence improve the accuracy overall.

Optimization for Model Inference:

Also tried the system on more powerful machines and optimized the code for running on CPUs. For the end, GPU acceleration could also be provided in options if the user has appropriate hardware.

Fine-tuning of Models:

Among the challenges that came in the way of performing sentiment analysis and, most importantly complex sentences, we considered fine-tuning the BERT model using more specialized data containing sarcastic, ambiguous, and subtle nuances. This would allow model to handle diversified ways of expressing sentiments.

User Feedback Loop:

Provided the feedback loop in the UI where a user can flag those cases when transcription or sentiment analysis is wrong. That data then could be used to fine-tune both models further and improve the performance for the next iterations.

CONCLUSION

This project showed the potential in using state-of-the-art deep learning models for the real-time sentiment analysis of audio recordings. By first using the Whisper model to convert speech to text and then using the BERT model to analyze the sentiment within that text, we have done just that-constructed a spoken language sentiment analyzer. Some of the points that were noted during development and testing have not only improved the system's performance but also gave deeper insight into both the challenges and limitations imposed by applying deep learning techniques to real-time audio analysis.

The need for high-quality input audio was among the salient features learned from the project. Whisper performs well on clean speech, but background noise, incoherent speech, and ambient factors seriously affect the transcription accuracy. This was a clear indication that audio preprocessing and noise reduction would play an important role in improving overall system performance. Additionally, we learned that while BERT is very good at general sentiment analysis, it falters on complex emotional expressions such as sarcasm, irony, and ambiguous statements. This pointed to the need for models that can find these minute sentiments, which are actually designed for the task at hand.

The project will impact because it will smoothly integrate speech processing and sentiment analysis into applications ranging from customer service, monitoring mental health, to analyzing social media. Converting speech to text and performing sentiment analysis-when this is done automatically by machines, the organization would come up with useful information on what customers do or feel, thereby making informed decisions that will hopefully improve user experience..

While generating a more significant amount, the BERT model needs further tuning with an expanded and complex dataset for better emotionally conveyed sentiments. It would not be out of place to extend the functionalities of the proposed application by using techniques for determining individual speakers and supporting multi-languages. Regarding this solution, the issue of performance in real time can be optimized further. In this respect, Model Compression and GPU acceleration for faster Inference will be further looked into.

In summary, this project provides a solid starting point on which to develop real-time speech-

based sentiment analysis that can become an impactful tool in many industries with just a few more tweaks in improvement in terms of accuracy and performance, driving innovation and creating an enhanced user experience using intelligent, automated spoken language analysis.