Audio Argument Detector

Automated Detection of Heated Conversations from Audio Recordings

Daniel Birsan, Digital Nation Coding Challenge - Submission

Contents

1. Introduction

1.1 Problem Statement

Modern multimedia content such as podcasts, interview recordings, and meeting archives often contain intense, argumentative segments. These can include heated debates, shouting, or emotionally charged conflicts between participants. Identifying and extracting those segments manually is time-consuming. This project aims to automate the process by developing a Python application that inputs an audio file (e.g., WAV, MP3) and outputs timestamps that indicate the location of these heated arguments.

1.2 Objectives and Goals

The main objectives of the project include:

- **Audio Signal Analysis:** Extract low-level features such as loudness and pitch to find potential signs of shouting.
- **Speech-to-Text Transcription:** Convert audio chunks into text for further sentiment or emotion analysis.
- **Emotion Detection from Text:** Classify text segments to identify high-anger content, indicative of arguments.
- **Timestamp Extraction:** Merge consecutive *heated* chunks into coherent segments and produce a final list of time ranges.

1.3 Applications

- **Podcasts and Interviews:** Quickly identify controversial or argumentative sections for editing or promotional highlights.
- **Meeting Recordings:** Detect conflicts or heated debates for subsequent review by HR or management.
- **Media Analysis:** Help journalists or researchers identify tension points in political or social discourse.
- **Customer Support Calls:** Pinpoint times when a call escalates, relevant for quality control and training.

2. Theoretical Background

2.1 Audio Signal Processing

2.1.1 Loudness (dBFS)

In digital audio, loudness can be approximated by the average power of the signal. PyDub represents this in terms of dBFS (decibels relative to full scale). Loud segments (e.g., above `-20 dBFS`) often correspond to shouting.

2.1.2 Pitch Estimation

Human speech pitch typically varies between 75 Hz and 300 Hz. Using the Librosa `piptrack` function, we estimate the fundamental frequency. Shouting tends to be at the higher end of pitch range for a speaker.

2.1.3 Voice Activity Detection (VAD)

WebRTC VAD is an effective method to determine if a 30 ms audio frame contains speech. In multi-speaker or tense debates, overlapping speech is a clue for argumentation.

2.2 Speech-to-Text Transcription

Whisper is a transformer-based model for automatic speech recognition developed by OpenAI. It can handle multiple languages and offers different sized models (tiny, base, small, medium, large). Whisper transforms short windows of audio into text, enabling subsequent text-based analysis.

2.3 Emotion Recognition from Text

Natural Language Processing (NLP) techniques can detect emotions such as anger, joy, sadness, or fear in textual data. In this project, we use a DistilBERT-based model (`bhadresh-savani/distilbert-base-uncased-emotion`) to classify each chunk's transcript into a distribution of emotions. A high anger score is taken as an indicator of heated discussion.

3. System Design and Architecture

3.1 Overview

Figure 3.1: High-level Architecture of the Audio Argument Detector

The system design, shown in Figure 3.1, begins by splitting the input audio into manageable chunks (e.g., 3-second segments). Each chunk is passed through multiple modules for feature extraction, transcription, and emotion detection. A final aggregator merges consecutive heated chunks into final timestamps.

3.2 Audio Feature Extraction

The primary audio features used include:

- **Loudness (dBFS):** From PyDub's `AudioSegment.dBFS`.
- **Pitch (Hz):** Using Librosa's `piptrack` to estimate mean pitch in each chunk.
- **Overlap Ratio:** Using WebRTC VAD, we measure how much of the chunk is active speech, thereby hinting at possible interruptions or multiple speakers.

3.3 Speech Transcription (Whisper)

- **Chunk-based Transcription:** Each chunk is temporarily saved as a WAV file and passed to the Whisper model.
- **Language Support:** The user can specify the language (e.g., English, Romanian, etc.)
- **GPU Acceleration:** If a CUDA-compatible device is detected, the model runs on GPU for faster inference.

3.4 Text-Based Emotion Detection

The transcribed and translated to english text from each chunk is analyzed by a Hugging Face pipeline, specifically configured for emotion classification. We typically obtain a probability distribution across various emotions (anger, fear, joy, etc.).

3.5 Heated Argument Scoring Algorithm

Each chunk gets a `heated_score` from 0 to 5, incremented based on:

- **Loudness Threshold:** If loudness $\geq$ -20 dBFS, add 1.
- **Pitch Threshold:** If pitch $\geq$ 200 Hz, add 1.
- **Overlap Threshold:** If the overlap ratio $\geq$ 0.3, add 1.
- **Anger Threshold:** If anger probability $\geq$ 0.8, add 1.
- **Heated Keywords:** If transcript contains words like *fight*, *shut up*, *idiot*, add 1.

Chunks with `heated_score` $\geq$ 4 are flagged as heated. Consecutive heated chunks are merged to form final segments.

4. Implementation

4.1 Environment Setup

The core libraries used include:

- **PyDub:** Audio processing, loudness measurement
- **Librosa:** Advanced audio analysis (pitch)
- **WebRTC VAD:** Voice Activity Detection
- **OpenAI Whisper:** ASR model for chunk-level transcription
- **Transformers (Hugging Face):** Emotion pipeline
- **Python Standard Libraries:** `os`, `argparse`, `json`, etc.

4.2 Code Structure and Explanation

The project is organized into the following components:

- `mp3totext.py`: Main entry point and orchestrator for detection.
- `AudioFeatureExtractor`: Extracts loudness, pitch, and VAD features.
- `WhisperTranscriber`: Chunk-based transcription using Whisper.
- `TextEmotionAnalyzer`: Analyzes transcripts for anger and other emotions.
- `HeatedArgumentDetector`: Integrates the modules and calculates final heated segments.

4.3 Main Modules

4.3.1 AudioFeatureExtractor

Responsible for calculating loudness (dBFS), pitch (Hz), and VAD-based speech overlap. In the code, we implement the pitch extraction using Librosa's `piptrack`, average the nonzero values, and use webrtcvad for speech detection frames.

4.3.2 WhisperTranscriber

Initializes a Whisper model (either CPU or GPU). It transcribes short chunks of audio by first exporting them as temporary WAV files, then passing them to the model.

4.3.3 TextEmotionAnalyzer

Loads a pretrained pipeline from Hugging Face (`text-classification` for emotion tasks). The output is a list of emotion scores, from which we extract the `anger` category.

4.3.4 HeatedArgumentDetector

Combines all the above to compute a chunk-level `heated_score`. It merges consecutive heated chunks into final timestamps. If the final segments exceed 10-15 seconds, they are considered relevant.

5. Usage and Examples

5.1 Command-Line Interface

Usage Example:

```
python mp3totext.py \
   --audio my_audio.mp3 \
   --chunk_ms 3000 \
   --loudness_thresh -20.0 \
   --pitch_thresh 200.0 \
   --overlap_thresh 0.3 \
   --anger_thresh 0.8 \
   --heated_score_cutoff 4.0 \
   --whisper_model small \
   --whisper_lang en \
   --output_detailed_json detailed_chunks.json \
   --output_segments_json heated_segments.json
```

Running this command processes the audio file in 3-second chunks, calculates the relevant features, transcribes each chunk, and writes final JSON outputs.

5.2 Sample Outputs and Analysis

5.2.1 Heated Segments JSON

An example of the final heated segments JSON could look like:

```
[
  {
    "start": "1:49",
    "end": "2:22"
  },
  {
    "start": "3:30",
    "end": "3:47"
  }
]
```

6. Discussion

6.1 Strengths and Limitations

6.1.1 Strengths

- Integrates multiple signals: loudness, pitch, overlap, text anger.
- Modular design: easy to swap or upgrade the ASR model or emotion classifier.
- Reasonable performance in real-world audio.

6.1.2 Limitations

- Single-channel overlap detection is naive; it does not truly detect multiple speaker channels.
- Transcription inaccuracies can reduce the reliability of text-based emotion detection.
- Processing speed is limited by the Whisper model size and hardware.

6.2 Potential Improvements

- **Speaker Diarization:** Distinguish speakers to detect true multi-speaker overlap.
- **Fine-tuned Language Model:** Adapt emotion detection for specific domain language (e.g., legal, political).
- **Real-time Streaming Support:** Process audio on-the-fly for live argument detection.
- **Adaptive Chunking:** Dynamically adjust chunk size based on speech activity.

7. Conclusion

7.1 Summary of Work

I developed a pipeline that merges audio signal processing, speech-to-text transcription, and emotion detection to locate argumentative or heated moments in audio files. This integrated approach offers a more robust solution than relying solely on volume or textual sentiment.

7.2 Future Work

Future enhancements may include improved diarization, domain-specific emotion analysis, and streaming capabilities for real-time conflict monitoring. Expanding the heated keyword list or training custom classifiers could also boost accuracy.

Appendices

Appendix B: Dependencies and Installation

Dependencies

- pydub
- librosa
- webrtcvad
- torch
- transformers
- openai-whisper

Installation

```
pip install pydub librosa webrtcvad torch transformers openai-
   whisper
```

Appendix C: GitHub Repository

The full project source code and README are available at:

-

Bibliography

[1] Google WebRTC VAD. *https://webrtc.org/*.

[2] OpenAI Whisper. *https://github.com/openai/whisper*

[3] Hugging Face Transformers. *https://huggingface.co/docs/transformers*

[4] PyDub Documentation. *https://github.com/jiaaro/pydub*