Approach to Identify and Redact Sensitive Data in Audio WAV Files

Overview

This approach combines traditional audio processing with generative AI to identify and redact sensitive information in WAV audio files. The solution involves speech-to-text conversion, sensitive data detection using context understanding, and precise audio redaction.

Step 1: Audio Preprocessing

1. **Load the WAV file** using libraries like librosa or pydub
2. **Normalize audio levels** to ensure consistent volume
3. **Remove background noise** (optional) using noise reduction algorithms
4. **Split into manageable chunks** if dealing with long recordings

Step 2: Speech-to-Text Conversion

1. Use **ASR (Automatic Speech Recognition)** services:
   * Cloud options: Google Speech-to-Text, AWS Transcribe, Azure Speech Services
   * Open-source: Whisper (OpenAI), Vosk, DeepSpeech
2. Generate **timed transcripts** with word-level timestamps
3. Include **speaker diarization** if multiple speakers are present

Step 3: Sensitive Data Identification

Traditional Methods:

1. **Pattern matching** for:
   * Credit card numbers
   * Social security numbers
   * Phone numbers
   * Email addresses
2. **Named Entity Recognition (NER)** for:
   * Personal names
   * Locations
   * Organizations

GenAI Enhancement:

1. Use **LLM context analysis** (GPT-4, Claude, etc.) to:
   * Understand conversation context
   * Detect implied sensitive information
   * Identify confidential business terms
   * Recognize sensitive topics based on industry/context
2. Prompt example:

text

Analyze this conversation transcript and identify all sensitive information

that should be redacted, including:

- Personal identifiable information

- Financial data

- Confidential business information

- Any other sensitive content

Return your findings with exact text matches and timestamps.

Step 4: Audio Redaction

1. **Create a beep tone**:
   * Generate a 1kHz sine wave (or other masking sound)
   * Match the duration of sensitive segments
2. **Overwrite sensitive portions**:
   * Use the timestamps from ASR to locate exact positions
   * Replace audio segments with the beep tone
3. **Smooth transitions** at redaction boundaries to avoid audio artifacts

Step 5: Validation

1. **Automatic validation**:
   * Re-run ASR on redacted audio to verify sensitive info is masked
   * Check that non-sensitive portions remain intact
2. **Human review** (for critical applications):
   * Manual spot-checking of redactions
   * Context verification

Implementation Tools

* Python libraries: librosa, pydub, whisper, transformers
* GenAI APIs: OpenAI, Anthropic, or self-hosted LLMs
* Audio processing: FFmpeg, SoX

Example Workflow Code Skeleton

python

import whisper

from pydub import AudioSegment

import numpy as np

def redact\_audio(input\_wav, output\_wav):

*# 1. Load audio*

audio = AudioSegment.from\_wav(input\_wav)

*# 2. Transcribe with timestamps*

model = whisper.load\_model("medium")

result = model.transcribe(input\_wav, word\_timestamps=True)

*# 3. Identify sensitive words (simplified)*

sensitive\_words = identify\_sensitive\_words(result["text"])

*# 4. Generate beep tone*

beep = generate\_beep()

*# 5. Redact sensitive portions*

for word in result["segments"]:

if word["text"] in sensitive\_words:

start\_ms = word["start"] \* 1000

end\_ms = word["end"] \* 1000

duration = end\_ms - start\_ms

audio = audio.overlay(beep[:duration], position=start\_ms)

*# 6. Save redacted audio*

audio.export(output\_wav, format="wav")

def identify\_sensitive\_words(text):

*# Combine pattern matching and LLM analysis*

*# This would call your GenAI service*

pass

def generate\_beep(duration=1000, freq=1000):

*# Generate beep tone*

sample\_rate = 44100

t = np.linspace(0, duration/1000, int(sample\_rate\*duration/1000), False)

beep\_wave = np.sin(2 \* np.pi \* freq \* t) \* 0.7

return AudioSegment(

(beep\_wave \* 32767).astype(np.int16).tobytes(),

frame\_rate=sample\_rate,

sample\_width=2,

channels=1

)

Advanced Considerations

1. **Speaker-specific redaction**: Different rules for different participants
2. **Context-aware redaction**: Understanding what's sensitive based on conversation context
3. **Alternative masking**: White noise, voice distortion, or complete removal instead of beeps
4. **Compliance tracking**: Logging what was redacted for audit purposes

This approach provides a balance between automated pattern matching and the nuanced understanding that GenAI can provide for contextual sensitivity detection.