Speech and Vocal Analysis from Video

1 Overview

This project extracts audio from a video file and performs speech recognition and vocal quality analysis. It includes functions to recognize speech, analyze vocal qualities, evaluate pace and pauses, and provide detailed insights based on vocal characteristics.

2 Requirements

- `moviepy`
- `speech_recognition`
- `librosa`
- `numpy`
- `wave`

3 Setup

1. Install the required libraries:

```
pip install moviepy speechrecognition librosa numpy
```

2. Place your video file in the project directory and update the `video_file` variable with the correct path:

```
video_file = '/Users/MAC/Desktop/fr.mov'
```

4 Code Overview

4.1 Import Libraries

```python
import moviepy.editor as mp
import speech_recognition as sr
import librosa
import numpy as np
import contextlib
import wave
```

4.2 Functions

4.2.1 Extract Audio from Video

```python
def extract_audio(video_file, audio_file):
    video = mp.VideoFileClip(video_file)
    video.audio.write_audiofile(audio_file)
```

4.2.2 Recognize Speech from Audio

```python
def recognize_speech(audio_file, language='fr-FR'):
    recognizer = sr.Recognizer()
    with sr.AudioFile(audio_file) as source:
        audio = recognizer.record(source)
    try:
        text = recognizer.recognize_google(audio, language=language)
        return text
    except sr.UnknownValueError:
        return "La reconnaissance vocale n'a pas pu comprendre l'audio"
    except sr.RequestError as e:
        return f"Impossible de demander les résultats du service de reconnaissance
```

4.2.3 Analyze Vocal Quality using Librosa

```python
def analyze_vocal_quality(audio_file):
    y, sr_librosa = librosa.load(audio_file)
    pitch, magnitude = librosa.core.piptrack(y=y, sr=sr_librosa)
    pitches = pitch[pitch > 0]
    mean_pitch = np.mean(pitches)
    rms = librosa.feature.rms(y=y)
    mean_intensity = np.mean(rms) * 100  # Scaling to dB

    # Vocal modulation analysis
    modulation = np.std(pitches)

    return mean_pitch / 7, mean_intensity * 200, modulation / 7
```

4.2.4 Evaluate Pace and Pauses

```python
def analyze_pace_and_pauses(audio_file, transcript):
    with contextlib.closing(wave.open(audio_file, 'r')) as f:
        frames = f.getnframes()
        rate = f.getframerate()
        duration = frames / float(rate)

    words = transcript.split()
    num_words = len(words)
    pace = num_words / duration  # words per second

    # Pauses detection using librosa
    y, sr_librosa = librosa.load(audio_file)
    onset_env = librosa.onset.onset_strength(y=y, sr=sr_librosa)
    times = librosa.times_like(onset_env, sr=sr_librosa)
    tempo, beats = librosa.beat.beat_track(onset_envelope=onset_env, sr=sr_librosa)
    pause_times = np.diff(times[beats])

    return pace, pause_times
```

4.2.5 Analyze Mean Pitch

```python
def analyze_mean_pitch(mean_pitch, gender='male'):
    if gender == 'male':
        if mean_pitch > 150:
            return "Hauteur moyenne élevée (indiquant de la nervosité ou de l'exci
        elif 100 <= mean_pitch <= 150:
            return "Hauteur moyenne modérée (indiquant du calme et de la confiance
        else:
            return "Hauteur moyenne basse (indiquant de l'autorité et de la relaxa
    elif gender == 'female':
        if mean_pitch > 225:
            return "Hauteur moyenne élevée (indiquant de la nervosité ou de l'exci
        elif 165 <= mean_pitch <= 225:
            return "Hauteur moyenne modérée (indiquant du calme et de la confiance
        else:
            return "Hauteur moyenne basse (indiquant de l'autorité et de la relaxa
    else:
        return "Genre inconnu pour l'analyse"
```

4.2.6 Analyze Mean Intensity

```python
def analyze_mean_intensity(mean_intensity):
    if mean_intensity < 60:
        return "Intensité moyenne basse (indiquant du calme, de l'introspection, d
    elif 60 <= mean_intensity <= 75:
        return "Intensité moyenne modérée (courante pour la conversation quotidien
    else:
        return "Intensité moyenne élevée (indiquant de l'enthousiasme, de la confi
```

4.2.7 Analyze Pitch Variability (Modulation)

```python
def analyze_modulation(modulation, gender='male'):
    if modulation < 20:
        return "Faible variabilité (suggère une livraison monotone, calme ou sérieu
    elif 20 <= modulation <= 40:
        return "Variabilité modérée (convoie du professionnalisme et de l'autorité)
    else:
        return "Forte variabilité (indique de l'expressivité émotionnelle et de l'e
```

4.2.8 Analyze Pace

```python
def analyze_pace(pace):
    if pace < 2.3:
        return "Rythme lent (indiquant de la réflexion ou du sérieux)"
    elif 2.3 <= pace <= 2.8:
        return "Rythme modéré (indiquant de la confiance et de la clarté)"
    else:
        return "Rythme rapide (indiquant de l'excitation ou de l'urgence)"
```

4.2.9 Analyze Pauses

```python
def analyze_pauses(pause_times):
    mean_pause = np.mean(pause_times)
    median_pause = np.median(pause_times)
    std_pause = np.std(pause_times)
    min_pause = np.min(pause_times)
    max_pause = np.max(pause_times)

    short_pauses = sum(0.2 <= pause <= 0.5 for pause in pause_times)
    moderate_pauses = sum(0.5 < pause <= 1.5 for pause in pause_times)
    long_pauses = sum(pause > 1.5 for pause in pause_times)

    analysis = {
        "Durée moyenne des pauses": mean_pause,
        "Durée médiane des pauses": median_pause,
        "Écart-type des pauses": std_pause,
        "Durée minimale des pauses": min_pause,
        "Durée maximale des pauses": max_pause,
        "Pauses courtes (0.2-0.5s)": short_pauses,
        "Pauses modérées (0.5-1.5s)": moderate_pauses,
        "Pauses longues (>1.5s)": long_pauses
    }

    return analysis
```

4.3 Main Function

```python
def analyze_video(video_file, gender='male'):
    audio_file = "extracted_audio.wav"
    extract_audio(video_file, audio_file)

    transcript = recognize_speech(audio_file, language='fr-FR')
    print("Transcription:", transcript)

    mean_pitch, mean_intensity, modulation = analyze_vocal_quality(audio_file)
    print(f"Qualité vocale - Hauteur moyenne: {mean_pitch:.2f} Hz, Intensité moyenn

    pace, pause_times = analyze_pace_and_pauses(audio_file, transcript)
    print(f"Rythme: {pace:.2f} mots par seconde")

    mean_pitch_analysis = analyze_mean_pitch(mean_pitch, gender)
    print(f"Analyse de la hauteur moyenne: {mean_pitch_analysis} (valeur typique: 1

    mean_intensity_analysis = analyze_mean_intensity(mean_intensity)
    print(f"Analyse de l'intensité moyenne: {mean_intensity_analysis} (valeur typic

    modulation_analysis = analyze_modulation(modulation, gender)
    print(f"Analyse de la modulation: {modulation_analysis} (valeur typique: 20-40

    pace_analysis = analyze_pace(pace)
    print(f"Analyse du rythme: {pace_analysis} (valeur typique: 2.3-2.8 mots par se

    pause_analysis = analyze_pauses(pause_times)
    print(f"Analyse des pauses: {pau_analysis}")
```

4.4 Usage

1. Ensure your video file path is correctly specified in the `video_file` variable.

2. Run the script:

```
python analyze_video.py
```

3. The script will extract audio from the video, perform speech recognition, and analyze vocal qualities, pace, and pauses.

5 Conclusion

This project demonstrates how to extract audio from a video and analyze various vocal qualities, including pitch, intensity, modulation, pace, and pauses. The results provide detailed insights into the speech characteristics, which can be useful for various applications such as speech therapy, communication analysis, and more.