Supporting Information
of
Methodology for Audio-to-Text Processing with Speaker Diarization

Table of Contents

Step 1. Preprocessing of Audio Data 3
Step 2. Speaker Diarization 5
Step 3. Audio Refinement 8
Step 4. Transcription Using WPS AI 9
Step 5. Integration of Outputs 10
Additional Implementation Details 10
Reference 10

This study leverages a hybrid approach for audio-to-text conversion, combining MATLAB's Pretrained Speaker Diarization System and WPS AI transcription to enhance transcription accuracy and generate speaker-specific timestamps. The methodology for converting audio into speaker-specific transcriptions and timestamps involves several steps: audio preprocessing, speaker diarization using a pretrained x-vector system, voice activity detection, transcription via WPS AI, and timestamp alignment. The detailed methodology and its application to an example are presented below.

Step 1. Preprocessing of Audio Data

An audio file named "new record 4.m4a" is processed using MATLAB for speaker diarization and transcription. The audio file, containing a multi-speaker conversation, is analyzed to extract speaker-specific transcriptions and timestamps. To accommodate diverse input audio formats and ensure uniformity, the audio signals are standardized:

Audio Loading and Normalization: The audio is read using MATLAB's "audioread" function and normalized by scaling the amplitude to a range of [-1, 1]:

```
[audioIn, fs] = audioread(fullFilePath);
audioIn = audioIn ./ max(abs(audioIn));
t = (0:size(audioIn,1)-1) / fs;
```

Visualization: The normalized waveform is plotted:

```
figure;
plot(t, audioIn);
xlabel("Time (s)");
ylabel("Amplitude");
axis tight;
```

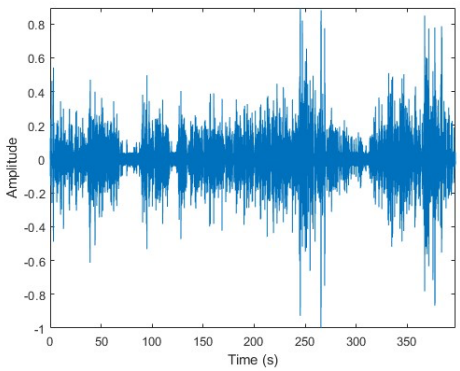


Figure S1. Normalized Waveform of "new record 4.m4a" After Audio Loading and Preprocessing

Dimensionality Reduction: For audio data with high-dimensional feature vectors, principal component analysis (PCA) is applied to reduce feature dimensions to 30, maintaining essential characteristics while improving computational efficiency and handling various audio formats:

```
for sample = 1:size(featuresSegmented, 3)
    [~, featuresPCA] = pca(featuresSegmented(:, :, sample), 'NumComponents', 30);
    featuresSegmentedPCA(:, :, sample) = featuresPCA;
end
featuresSegmented = featuresSegmentedPCA;
```

Step 2. Speaker Diarization

Speaker diarization is performed using MATLAB's pretrained x-vector system, designed for identifying "who spoke when" without prior knowledge of speakers or their number. The Pretrained Speaker Diarization System in MATLAB is employed to segment audio into speaker-specific clusters:

Feature Extraction: Acoustic features are extracted using an audioFeatureExtractor object. These features are standardized using mean and standard deviation values derived from a reference dataset. Features are extracted using the x-vector framework, with Mel-Frequency Cepstral Coefficients (MFCCs) standardized for input to the neural network.

```
xvecsys = load('xvectorSystem.mat');
features = single((extract(xvecsys.afe, audioIn) - xvecsys.factors.Mean') ./ xvecsys.factors.STD');
```

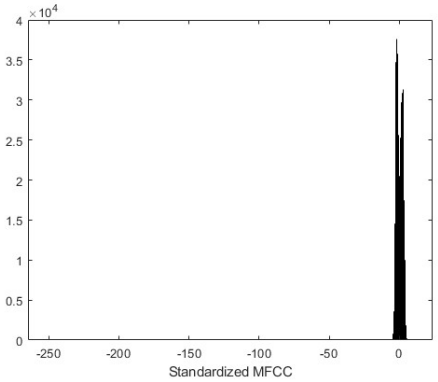


Figure S2. Standardized Mel-Frequency Cepstral Coefficients (MFCCs) of "new record 4.m4a"

X-Vector Generation: Extracted features are processed through a pretrained deep learning network to generate x-vectors, which are compact speaker-specific embeddings. Figure 3 shows the raw x-vectors extracted from the pretrained x-vector system neural network. Each column corresponds to a specific segment of the audio, and each row represents a feature dimension (512 features in total). It visualizes the unprojected x-vectors in their high-dimensional feature space, which is often dense and not optimized for clustering or downstream tasks. Figure 4 shows the x-vectors after applying the projection matrix, which reduces their dimensionality and aligns them for better separation. The projection optimizes the x-vectors by reducing noise and enhancing speaker-specific features, making them more suitable for clustering.

```
xvecs = zeros(512, size(featuresSegmented, 3));
for sample = 1:size(featuresSegmented, 3)
    dlX = dlarray(featuresSegmented(:, :, sample), "TCB");
```

```
    xvecs(:, sample) = predict(xvecsys.dlnet, dlX, Outputs="fc_1");
end
```

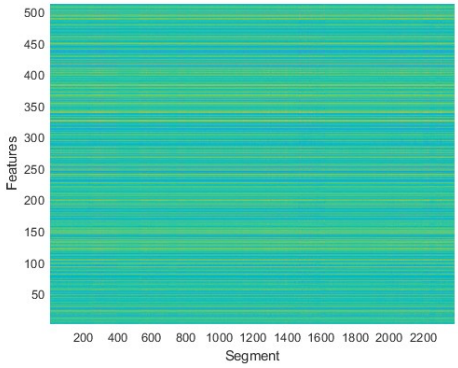


Figure S3. High-Dimensional X-Vectors of "new record 4.m4a" Before Dimensionality Projection

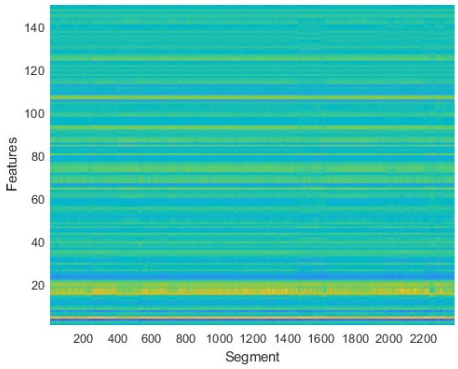


Figure S4. Projected X-Vectors of "new record 4.m4a" for Enhanced Speaker Clustering

Segmentation and Clustering: The audio is segmented into fixed durations (e.g., 1-second windows with 0.5-second hops). The x-vectors are clustered into speaker groups using agglomerative hierarchical clustering (AHC) with Probabilistic Linear Discriminant Analysis (PLDA) scoring. This assigns each segment to a specific speaker identity. The features were standardized by subtracting the mean and dividing by the standard deviation.

```
segmentDur = 1; segmentHopDur = 0.5;
segmentLength = round(segmentDur / featureVectorHopDur);
segmentHop = round(segmentHopDur / featureVectorHopDur);

T = clusterdata(x', Criterion="distance", distance=@(a,b)helperPLDAScorer(a,b,xvecsys.plda), linkage="average", maxclust=maxclusters);
```

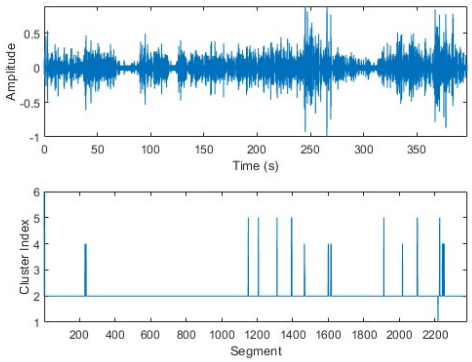


Figure S5. Agglomerative Hierarchical Clustering Results of Speaker Segments in "new record 4.m4a"

Step 3. Audio Refinement

Non-speech portions are removed by applying a voice activity detection (VAD) mask to the diarization results. The time alignment of speech is refined using the output of speaker clustering and VAD. The clustering results are mapped to the original audio timeline to generate a mask corresponding to active speaker intervals. By combining the clustering mask with the VAD mask, non-speech portions are excluded, ensuring that only speech intervals are retained. The final result is shown in Figure 5. The colored part is the dialogue part, and the colorless part is the interval or noise that will be removed. Finally, only the dialogue part will be output as a new file “filtered_new record 4.m4a”.

```
VADidx = detectSpeech(audioIn, fs, MergeDistance=fs * 0.1);
VADmask = sigroi2binmask(VADidx, numel(audioIn));
mask = mask .* VADmask;
```

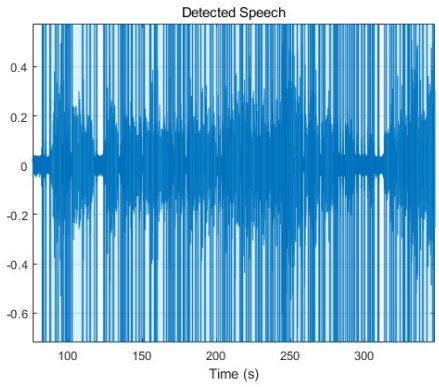
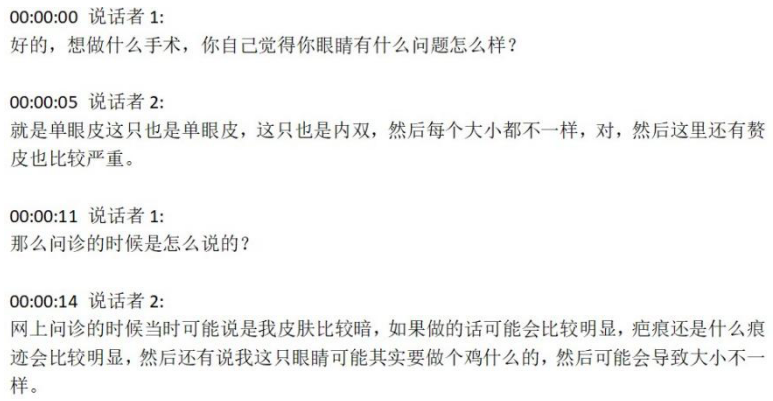


Figure S6. Detected Speech Segments in “new record 4.m4a” After Applying VAD Mask

Step 4. Transcription Using WPS AI

Filtered audio files containing only speech segments are processed with WPS AI's transcription service:

- WPS AI provides functionality to distinguish speakers and perform speech-to-text conversion. However, it only generates the start timestamp for each speaker without indicating the end timestamp.
- To calculate the speaking durations for each speaker, the diarization results from the previous steps are utilized. Specifically, since the intervals in the conversation have been removed, the duration of the speaker's speech is inferred by subtracting the start timestamp of the current speaker from the start timestamp of the next speaker. This ensures accurate alignment and duration calculation for each speaker.



00:00:00 说话者 1:
好的，想做什么手术，你自己觉得你眼睛有什么问题怎么样？

00:00:05 说话者 2:
就是单眼皮这只也是单眼皮，这只也是内双，然后每个大小都不一样，对，然后这里还有赘皮也比较严重。

00:00:11 说话者 1:
那么问诊的时候是怎么说的？

00:00:14 说话者 2:
网上问诊的时候当时可能说是我皮肤比较暗，如果做的话可能会比较明显，疤痕还是什么痕迹会比较明显，然后还有说我这只眼睛可能其实要做个鸡什么的，然后可能会导致大小不一样。

Figure S7. Final Speaker-Labeled Transcription Results of "filtered_new record 4.m4a"

Step 5. Integration of Outputs

The final output consists of:

1.Speaker-Labeled Transcriptions: The text segments are associated with respective speakers.

2.Speaking Durations: The timestamp for each speaker is calculated and appended to the transcription.

Additional Implementation Details

This methodology adapts concepts from the “Speaker Diarization Using x-vectors” example provided in MATLAB documentation:

X-Vector System Components: The x-vector system comprises a pretrained neural network (dlnet), a projection matrix for dimensionality reduction, and a PLDA model for scoring speaker clusters.

By integrating advanced speaker diarization techniques with WPS AI transcription capabilities, this method ensures precise speaker segmentation and accurate transcription, meeting the demands of diverse audio processing applications.

Reference

[1] MathWorks. (n.d.). Speaker Diarization Using x-vectors. MathWorks China. Retrieved January 24, 2025, from https://ww2.mathworks.cn/help/audio/ug/speaker-diarization-using-x-vectors.html