Final Project: Speaker Recognition System
EEC 201

1. Introduction

Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity for authentication of access.

This project builds a simple and representative automatic speaker recognition system.

2. Principles of Speaker Recognition

Speaker recognition can be classified into identification and verification. Speaker identification is the process of determining which registered speaker provides a given word/phrase.

Figure 1 provides the basic elements of feature extraction and feature matching. Feature extraction extracts a small amount of vital data from the voice signal that can later be used to represent each speaker. Feature matching identifies the speaker by comparing extracted features with those of known speakers. We will discuss each module in detail in later sections.

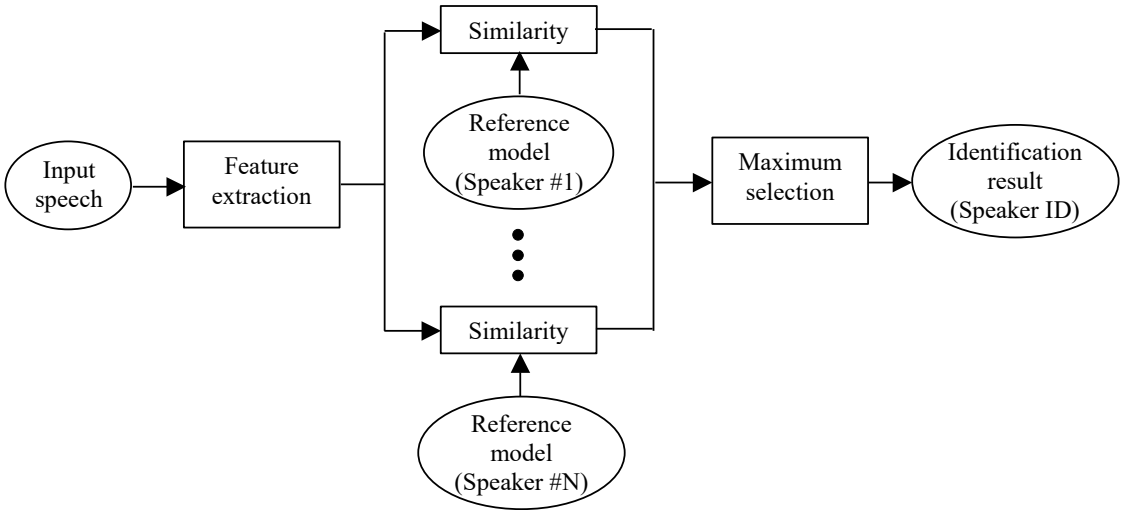

Figure 1. Basic structures of speaker identification systems

All speaker recognition systems have to serve two distinguished phases. The first one is referred to the enrolment or training phase, while the second one is referred to as the operational or testing phase. In the training phase, each registered speaker has to provide samples of their speech so that the system can build or train a reference model for that speaker. In case of speaker verification systems, in addition, a speaker-specific threshold is also computed from the training samples. In the testing phase, the input speech is matched with stored reference model(s) and a recognition decision is made.

Automatic speaker recognition works based on the premise that a person's speech exhibits characteristics that are unique to the speaker. However, this task has been complicated by the highly variant of input

speech signals. The principal source of variance is the speaker himself/herself. There are also other factors beyond speaker variability.

3. Speech Feature Extraction

3.1 Introduction

Using digital signal processing (DSP) tools can extract a set of features for further analysis.

The speech signal is a slowly time varying signal (it is called quasi-stationary). When examined over a sufficiently short period of time (between 5 and 100 msec), its characteristics are mostly stationary. Over long duration (> 1/5 seconds), their signal characteristic variation would reflect the different sounds being spoken. Therefore, short-time spectral analysis is the most common way to characterize the speech signal.

A wide range of possibilities exist for parametrically representing the speech signal for the speaker recognition task, such as Linear Prediction Coding (LPC), Mel-Frequency Cepstrum Coefficients (MFCC), and others. MFCC is perhaps the best known and most popular, and will be suggested for this project.

MFCC's are based on the known variation of the human ear's critical bandwidths with frequency, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. This is expressed in the mel-frequency scale, which is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. One nice discussion can be found at https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

3.2 Mel-frequency cepstrum coefficients processor

In an MFCC processor given in Figure 2, speech signals are typically sampled above 10000 Hz. The main purpose of the MFCC processor is to mimic the behavior of the human ears. In addition, rather than the speech waveforms themselves, MFFC's are shown to be less susceptible to variations.

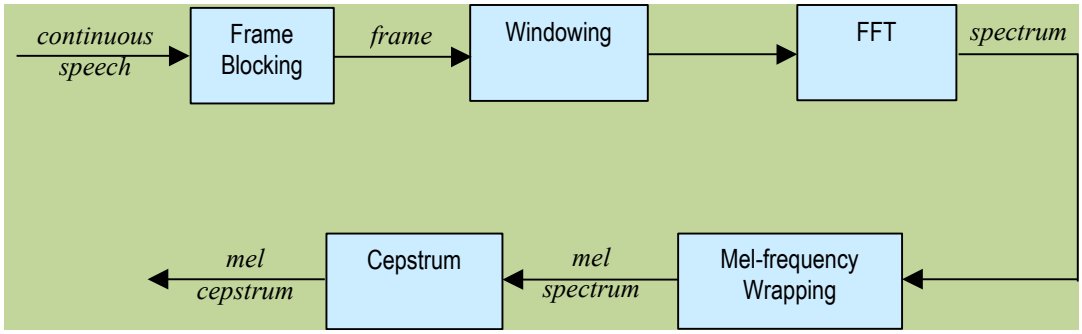

Figure 2. Block diagram of the MFCC processor

3.1.1 Frame Blocking

The speech signal is blocked into frames of N samples with overlap: The first frame consists of the first N samples. The second frame begins M samples after the first frame, and overlaps it by N - M samples, etc. N and M are N = 256 (which is equivalent to ~ 30 msec windowing) and M = 100.

3.1.2 Windowing

The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as $w(n), 0 \le n \le N-1$, where N is the number of samples in each frame, then the result of windowing is the signal

$$y_l(n) = x_l(n)w(n), \quad 0 \le n \le N-1$$

Typically the Hamming window is used, which has the form:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

The result after FFT is a periodogram.

3.1.3 Mel-frequency Wrapping

Psychophysical studies have discovered that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency, f, measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. The mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz.

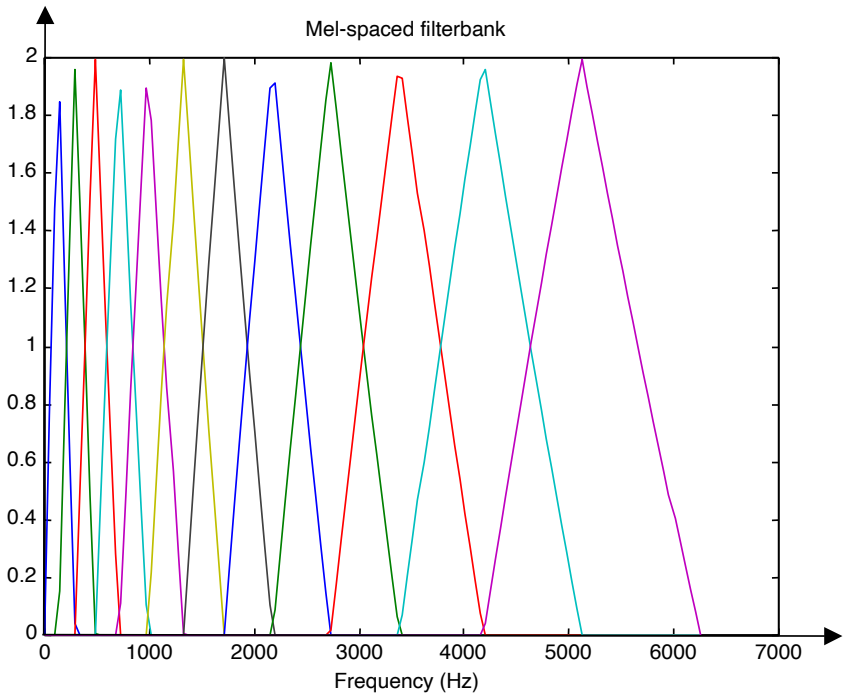

Figure 3. An example of mel-spaced filterbank

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel-scale (see Figure 3), which typically has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval. Typical number of mel spectrum coefficients, K, is chosen as 20. One can view each filter as a spectral histogram bin.

3.1.4 Cepstrum

Cepstrum converts log mel spectrum back to time. The result is called the mel frequency cepstrum coefficients (MFCC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the mel spectrum coefficients (and so their logarithm) are real numbers, they can be converted to the time domain using the Discrete Cosine Transform (DCT). Let mel power spectrum coefficients be $\tilde{S}_0, k = 0,2,..., K-1$, one can calculate the MFCC's, $\tilde{c}_n$, as

$$\tilde{c}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos \left[n \left(k - \frac{1}{2} \right) \frac{\pi}{K} \right], \qquad n = 0, 1, ..., K-1$$

One can exclude the first component, $\tilde{c}_0$, from the DCT since it represents the mean value of the input signal, which contains little speaker specific information.

With MFCC, each voice utterance is transformed into a sequence of acoustic vectors.
It is a common method to use filter-banks [Haytham Fayk].

3.2 Feature Matching

Once the features are extracted, the problem of speaker recognition belongs to a much broader topic known as pattern recognition. Since the classification procedure here is applied on extracted features, it can be also referred to as feature matching. Since the set of patterns that the individual classes of which are already known, then one has a problem in supervised learning.

All data can be divided into a training set and a test set.

The state-of-the-art techniques now include deep learning (DL), Hidden Markov Modeling (HMM), and Vector Quantization (VQ).

In this project, you are advised to consider the methods of DL and VQ, given their ease of implementation and high accuracy. DL uses neural networks and requires substantial training data. By comparison, VQ is a simpler process of mapping vectors from a large vector space to a few regions therein. Each region is called a cluster whose center is a codeword. The collection of all codewords forms a codebook.

Figure 4 shows a conceptual diagram to illustrate this recognition process using 2 speakers and also a 2-D vector feature space. In the training phase, using a clustering algorithm, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The result codewords (centroids) are shown. The distance from a vector to the closest codeword of a codebook is called a VQ-distortion. In the recognition phase, an input utterance of an unknown voice is also “vector-quantized” using each trained codebook and the total VQ distortion is computed. The speaker corresponding to the VQ codebook that generates the smallest total distortion is identified as the speaker.

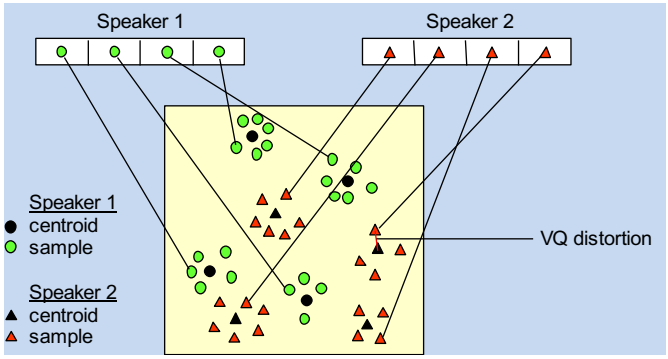

Figure 4. Conceptual diagram illustrating vector quantization codebook formation.

3.3 Clustering the Training Vectors

The acoustic vectors extracted from input speech of each speaker provide a set of training vectors for that speaker. The next step is to build a speaker-specific VQ codebook using the training vectors. A well-know LBG algorithm [Linde, Buzo and Gray, 1980], teaches the clustering a set of *L* training vectors into a set of *M* codebook vectors. The algorithm is formally implemented by the following recursive procedure:

a. Initialize with a single-vector codebook; this is the centroid of all training vectors.

b. Double the size of the codebook by splitting each current codebook yn according to the rule

$$y_n^+ = y_n(1+\varepsilon)$$

$$y_n^- = y_n(1-\varepsilon)$$

as *n* varies from 1 to the current size of the codebook, and ε is a splitting parameter (can choose ε=0.01).

3. Nearest-Neighbor Search: *for each training vector, find the closest codeword in the current codebook (in terms of similarity measurement), and assign that vector to the corresponding cell (associated with the closest codeword).*

4. *Centroid update: update the codeword in each cell using the centroid of training vectors assigned to that cell.*

Iteration 1: repeat steps 3 and 4 until the average distance falls below a preset threshold

Iteration 2: repeat steps 2, 3 and 4 until a codebook size of M is designed.

Intuitively, the LBG algorithm designs an M-vector codebook in stages. It starts first by designing a 1-vector codebook, then uses a splitting technique on the codewords to initialize the search for a 2-vector codebook, and continues the splitting process until the desired M-vector codebook is obtained.

Here is an example:

```
function m = melfb_own(p, n, fs)
% MELFB_own        Determine matrix for a mel-spaced filterbank
%
% Inputs:      p = number of filters in filterbank
%              n  = length of fft
%              fs  = sample rate in Hz
%
% Outputs:     x = a (sparse) matrix containing the filterbank amplitudes
%              size(x) = [p, 1+floor(n/2)]
%
```

```
% Usage:  Compute the mel-scale spectrum of a colum-vector s, with length n and sample rate fs:
%            f = fft(s);
%            m = melfb(p, n, fs);
%            n2 = 1 + floor(n/2);
%            z = m * abs(f(1:n2)).^2;
%
%            z would contain p samples of the desired mel-scale spectrum
%            To plot filterbank responses:
%            plot(linspace(0, (12500/2), 129), melfb(20, 256, 12500)'),
%            title('Mel-spaced filterbank'), xlabel('Frequency (Hz)');

f0 = 700 / fs;   fn2 = floor(n/2);
Lr = log(1 + 0.5/f0) / (p+1);

% convert to fft bin numbers with 0 for DC term
Bv = n*(f0*(exp([0 1 p p+1]*Lr) - 1));

b1 = floor(Bv(1)) + 1;  b2 = ceil(Bv(2));
b3 = floor(Bv(3));   b4 = min(fn2, ceil(Bv(4))) - 1;

pf = log(1 + (b1:b4)/n/f0) / Lr;   fp = floor(pf);    pm = pf - fp;

r = [fp(b2:b4) 1+fp(1:b3)];
c = [b2:b4 1:b3] + 1;
v = 2 * [1-pm(b2:b4)   pm(1:b3)];

m = sparse(r, c, v, p, 1+fn2);
```

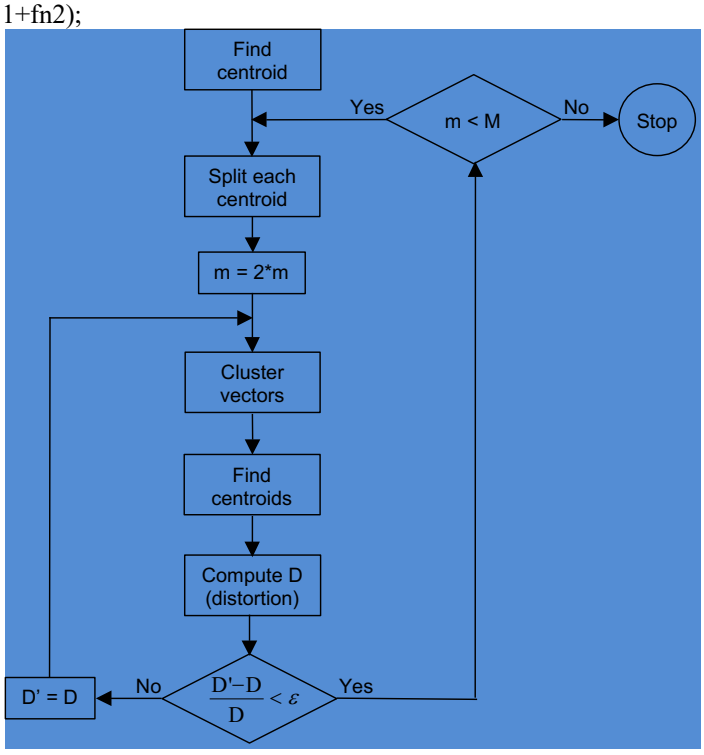

Figure 5. Flow diagram of LBG algorithm (Adopted from Rabiner and Juang, 1993)

```
% To compute Euclidean distance D between two vectors or matrices X and Y:
%       D = norm(X-Y,2)
```

Required Project Tasks

Using the convenient platform of Matlab, many of steps have already been implemented. The project Web page given at the beginning provides a test database and several helper functions to ease the development process.

A. Speech Data Files

Download the ZIP file of the speech database from canvas. After unzipping the file, you will have 11 speech files, named: S1.WAV, S2.WAV, …; each is labeled after the ID of the speaker. These files were recorded in WAV format. These files are used as training data.

Our goal is to train a voice model (e.g., a VQ codebook in the MFCC vector space) for each speaker using the corresponding sound file. After this training step, the system would have knowledge of the voice characteristic of each (known) speaker. Next, in the testing phase, you should add noises to distort the existing training signals to generate a test set. The amount of noises would vary to test the robustness of your system.

TEST 1: To begin, use the voice files of "zero" that we provided (not from the class). Play each sound file in the TRAIN folder. Can you distinguish the voices of the given 11 speakers in the database? Next play each sound in the TEST folder in a random order without looking at the groundtruth and try to identify the speaker manually. Record what is your (human performance) recognition rate. Use this result as a later benchmark.

B. Speech Preprocessing

Write a function that reads a sound file and turns it into a sequence of MFCC (acoustic vectors) using the speech processing steps described previously. Helpful matlab functions include: wavread, hamming, fft, dct, and own function melfb_own.m

TEST 2: In Matlab one can play the sound file using "sound". Record the sampling rate and compute how many milliseconds of speech are contained in a block of 256 samples? **Now plot the signal to view it in the time domain. It should be obvious that the raw data are long and may need to be normalized because of different strengths.**

Use STFT to generate periodogram. Locate the region in the plot that contains most of the energy, in time (msec) and frequency (in Hz) of the input speech signal. Try different frame size: for example N = 128, 256 and 512. In each case, set the frame increment M to be about N/3.

TEST 3: Plot the mel-spaced filter bank responses. Compare them with theoretical responses (e.g. triangle shape that you expected). Compute and plot the spectrum of a speech file before and after the mel-frequency wrapping step. Describe and explain the impact of the melfb.m or melfbown.m program.

TEST 4: Complete the "Cepstrum" step and put all pieces together into a single Matlab function, e.g., mfcc.m

C. Vector Quantization

Now apply VQ-based pattern recognition technique to build speaker reference models from those vectors in the training set before identifying any sequences of acoustic vectors from unmarked speakers in the test set.

TEST 5: To check whether the program is working, inspect the acoustic space (MFCC vectors) in any two dimensions in a 2D plane to observe the results from different speakers. Are they in clusters?

Now write a function that trains a VQ codebook using the LGB algorithm.
TEST 6: Plot the resulting VQ codewords using the same two dimensions over the plot of in TEST 5. You should get a figure like Figure 4.

D. Full Test and Demonstration

Using the programs to train and test speaker recognition on the data sets.

TEST 7: Record the results. What recognition rate can your system achieve? Compare this with human accuracy. Experiment and find the reason if high error rate persists. Record more voices of yourself and your teammates/friend. Each new speaker can provide one speech file for training and one for testing. **Record the results.**

TEST 8: Use notch filters on the voice signals to generate another test set. Test your system on the accuracy after voice signals have passed different notch filters that may have suppressed distinct features of the original voice signal. Report the robustness of your system.

From this point on, we are recording our own speeches as tests.

TEST 9: Assuming that your system has been trained with the non-students' "zero" speeches, we now augment the problem by adding 10 students' voice as follows: **(a)** Randomly select speech "zero" of 10 students from 2024 we recorded twice: one for training and one for recognition test. **(b)** Next, retrain your system by adding 2024 recorded speech to our existing speakers' samples. **(c)** Test the accuracy of your system and compare the accuracy obtained previously. NOW this system can identify more speackers that include the original set of speakers + the 10 students from 2024.

TEST 10a: Use all samples of EEC 2024 students saying "zero" and "twelve". Retrain the speech recognition system and test the accuracy in terms of the accuracy to identify the different "zero" and "twelve" sounds. Question 1: If we use "twelve" to identify speakers, what is the accuracy versus the system that uses "zero"? Question 2: If we train a whole system that tries to identify a) which speaker, and b) whether the speech is "zero" or "twelve", how accurate is your system?

TEST 10b: Use all samples of EEC 2025 students saying "five" and "eleven". Retrain the speech recognition system and test the accuracy in terms of the accuracy to identify the different "five" and "eleven" sounds. Question 3: If we use "eleven" to identify speakers, what is the accuracy versus the system that uses "five"? Question 4: How well do they compare against test in **10a** using zero/twelve?

References

[1] L.R. Rabiner and B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, N.J., 1993.

[2] L.R Rabiner and R.W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, N.J., 1978.

[3] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", IEEE Transactions on Acoustics, Speech, Signal Processing, Vol. ASSP-28, No. 4, August 1980.

[4] Y. Linde, A. Buzo & R. Gray, "An algorithm for vector quantizer design", IEEE Transactions on Communications, Vol. 28, pp.84-95, 1980.

[5] S. Furui, "Speaker independent isolated word recognition using dynamic features of speech spectrum", IEEE Transactions on Acoustic, Speech, Signal Processing, Vol. ASSP-34, No. 1, pp. 52-59, February 1986.

[6] S. Furui, "An overview of speaker recognition technology", ESCA Workshop on Automatic Speaker Recognition, Identification and Verification, pp. 1-9, 1994.

[7] F.K. Song, A.E. Rosenberg and B.H. Juang, "A vector quantisation approach to speaker recognition", AT&T Technical Journal, Vol. 66-2, pp. 14-26, March 1987.

[8] Frequently Asked Questions WWW site, http://svr-www.eng.cam.ac.uk/comp.speech/