LABOS 7-8

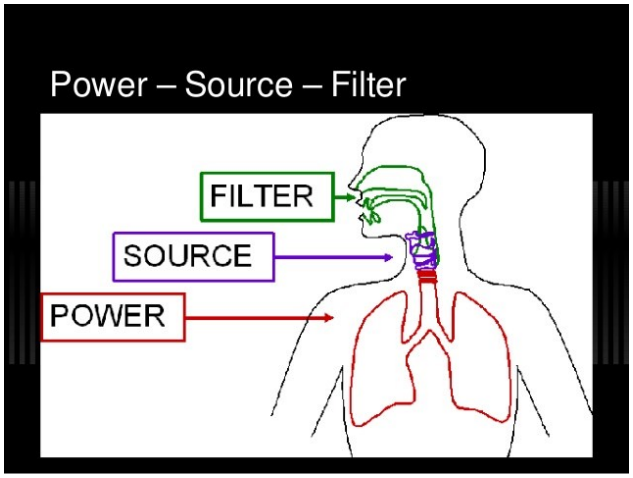
Speaker Classification: A Python Project

7.1 Introduction

In this mini-project, we are going implement rule-based systems using signal processing techniques and compare them to machine learning based techniques. In order to do so we will focus on a speaker classification application. Speaker classification is the task of recognising different speakers usually mainly based on their voices. This is not to be confused with speech recognition for which the task is to recognition what is being said by a certain speaker.

7.2 How is Speech Produced ?

Speech is composed of voiced (ex: a, o, i, e) and unvoiced (ex: f, h, s) sounds. The glottis either open or closes while air is sent from the lungs. When the glottis is opened, the air passes through the glottis and then through the vocal tract producing unvoiced sounds depending on the shape of the vocal tract. In this case the air sent from the glottis can be represented as white noise and the glottis as an all-pole filter of which the parameters depend on the shape of the vocal tract. This latter therefore filters the white noise producing one of the unvoiced sounds depending on the shape of the vocal tract. In other words, the parameters of the all-pole filter are changed every time a new unvoiced sound needs to be produced. Similarly, when the glottis is closed, it vibrates when the air, from the lungs arrives and a signal similar to a Dirac comb is produced. This signal is then filtered by the vocal tract in the same way as before. This analogy of speech production to signals filtered is called the source-filter model (very original! I know!)



7.3 Rule-Based Systems : Definition

Another original name referring to systems that rely on rules set by the developers, usually in the form of hand-coded program based on IF-ELSE conditions, to give an outcome or take a decision (IF condition1 is met then decision1, ELSEIF condition2 is met then decision2, etc...)

So, in this project, we will have to set some "rules" or conditions, that will be checked in the incoming speech to decide whether it belongs to speaker A or speaker B.
An example of a recorded sentence is given in the figure below.
In order to do so, speech parameters or "features" will be extracted from the speech signals which describe it. We will then analyse each of these features in order to define some rules based on them with the goal to recognise between different speakers. Among the mostly used features in speech processing in general and which we will consider in this mini-project are:

- The signal's energy
- The fundamental frequency (F0)
- The Mel-Frequency Cepstrum Coefficients (MFCC)
- The Formant frequencies

7.4 Database Used

For this work, we will require a database of recorded sentences. We will thus pick, from the CMU Arctic database, speakers BDL (male) and SLT (female):

http://www.festvox.org/cmu_arctic/

The data that are used to evaluate our system are usually not the same data used to build it. Indeed, the point of building a system is usually to work accurately with new previously unseen data. Evaluating it with data it was "built upon" will add a certain bias and thus give a fake evaluation of the system. In this case, a system with good evaluations has high chances of performing poorly in run-time.
In what follows, keep this thought in mind when a system's evaluation is required.

7.5 Signal Pre-Processing

Speech data in a database can be messy (for example: having different range of values or containing noise) or not organised. They need therefore some preprocessing before being used. For instance they need to put in a certain format that fits the targeted application or filtered to remove part of a constant noise present in all of them.

1. A first pre-processing to consider is called normalisation: the signal's sample values are scaled between -1 and 1 so that all the speech signals have values in the same range. To do that please write a python function that would take a mono channel (1 dimensional array) signal as input and divide it by the maximum value of this signal.

2. Another pre-processing to consider is splitting each speech segment in overlapping or nonoverlapping frames. Indeed, in order to obtain values representing signals as precisely as possible, we need to extract features not from the whole signal but from each frame extracted from a signal. So, please write another function that would take a signal and output a list of frames. This procedure is like sliding a window of a certain size, one step at a

time all along a signal. Each step will extract a frame. To do this we will therefore define two variables: the width of the signal in ms and the shifting steps also in ms.

7.6 Features Extraction Algorithms

In this section a description of the algorithms used to extract features will be given and you will be asked to implement each of them in Python

Signal Energy

The signal energy is computed by applying:

$$E = \sum |x(i)|^2$$

With i being the indices of the elements of the signal x

1. Write a function that would output the energy of a given signal.

Pitch

The fundamental frequency (f0) is the lowest frequency that constitute a signal. It represents the source signal of a source-filter model. Its values ranges from 60Hz for severe to 500Hz for acute voices. The term pitch is commonly used in the literature to mean fundamental frequency, but the pitch is related to subjective psychoacoustic perception while the fundamental frequency is the lowest frequency in the spectral domain.
Voiced sounds contain pitch and therefore have a non-zero f0 value while unvoiced sounds don't and therefore have a null f0.

Autocorrelation-Based Pitch Estimation System:

In this section we will implement the autocorrelation-based algorithm to estimate the pitch:
Since f0 is the lowest frequency contained in a voiced signal, the distance between the highest peaks in the autocorrelation of this signal in the temporal domain should be an estimation of the fundamental period, i.e. 1/f0. So by finding this distance, we can estimate f0.

To do this, the algorithm will be split in several points. Please implement each of them:

1. normalise the signal

2. split the signal into frames

3. compute the energy of the each frame

4. for each frame compare its energy to a threshold to be determined. If it is greater than the threshold consider it as voiced if smaller, considers it as unvoiced. (voiced/unvoiced classification is a very well studied domain and such much more accurate solutions can be found but for the sake of pitch estimation and of this lab particularly, not a lot of precision is required and so using only the energy will be enough for here)

5. on the voiced segments, compute the autocorrelation of the signal using the xcorr function of the pyplot library and a maxlag = the lowest f0 possible (50 Hz). Finally find the distance between the two peaks and estimate f0.

6. for the unvoiced ones f0 = 0.

Cepstrum-Based Pitch Estimation System

In this section, we will implement the cepstrum-based algorithm to estimate the pitch. This algorithm is defined as follows

- normalise the signal
- split the signal into frames
- compute the energy of the each frame
- for each frame compare its energy to a threshold to be determined. If it is greater than the threshold consider it as voiced if smaller, considers it as unvoiced.
- on the voiced segments, compute the cepstrum and from these determine f0
- for the unvoiced ones f0 = 0

By definition, a cepstrum (plural cepstra) is the result of taking the inverse Fourier transform (IFT) of the logarithm of the estimated spectrum of a signal. In other words you will first need to compute the frequency response of a signal. Then calculate its logarithmic value. Finally the latter's inverse Fourier transform will give you the signal's cepstra. The intuition behind this and the reason why it is related to the pitch is the following: The harmonics are multiple of the fundamental frequency f0. They are therefore separated from one another by f0 Hz in the spectral domain. The inverse FFT of the log amplitude spectrum (or log frequency response) of a signal will thus show a peak at the value corresponding the spacing between these harmonics, i.e. the fundamental frequency. So after the cepstrum vector for a certain frame is obtained, your algorithm should search for a peak value (maximum value) in the part of the cepstrum vector where the fundamental frequency should be (usually between 60 and 500 Hz)

1. We will analyse the energy of the voiced and unvoiced sounds. For this, first, write a short program that would randomly select 5 utterances from each speaker.

2. For each sentence you will need to visualize the temporal representation of the signal as well as the corresponding energy signal. So write a function that would take as input a speech signal and plot its temporal representation and energy on two subplots on the same figure. The energy is calculated on each frame of a signal. So you will need to normalize then split your signal in frames before calculating the energy on each frame.

3. Use the visualization to choose a convenient threshold to separate voiced sounds from unvoiced ones.

To implement the algorithm:

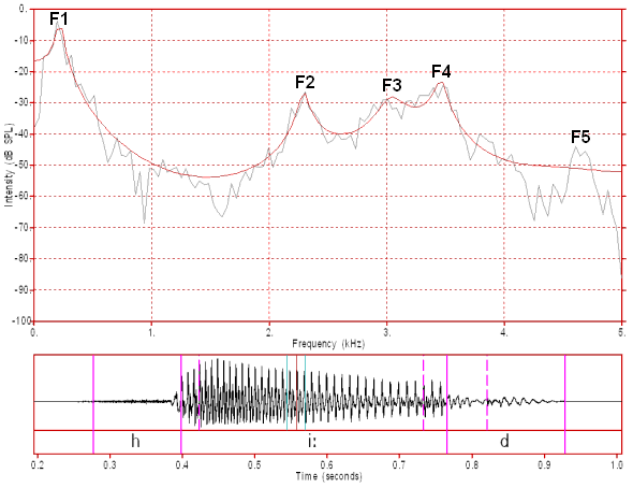
4. Please write a function that implements the steps described above. Make also sure to add plots of the temporal representation and the estimated pitch signal on two subplots on the same figure.

5. Add a line of code that would apply a Hamming window on the frame before extracting the cepstra and after calculating the energy from it. Compare the pitch estimation results obtained with the Hamming and with the rectangular window.

6. To obtain the estimated pitch, pick the index corresponding to the highest values

Formants

Formants are the peaks found in the frequency response envelope. They are therefore frequency values that represent the resonances of the vocal tract. They are therefore characteristic to the way it is shaped at a certain instant and thus also characteristic to the sound produced. In order to estimate the formants, we will rely on a Linear Predictive Coding (LPC)-based algorithm. The LPC are coefficients that approximate the vocal tract as the coefficients of the all-pole filter in the source-filter model. So the formants, with respect to this filter, can be seen as its poles since they are peaks at certain frequencies.



In order to implement the algorithm to estimate the formants please follow the steps enumerated below:

1. Split the signal in frames

2. pass the signal into a first order high pass filter in order to balance the high and low frequencies of the signal (usually high frequencies have lower values) and to import the signal to noise ratio since part of the noise in the recordings are in the low frequencies:

$$y(t)=x(t)-\alpha x(t-1)$$

where α is between 0.62 and 0.67. We will take $\alpha = 0.67$ here.
This is called the preamplasis step or preamphasising.

3. Apply a Hamming window onto the signal

4. Extract the LPC coefficients on each windowed frame by using the LPC function of the scikits.talkbox library. To determine the order of the LPC coefficients

(remember that they are also the coefficients of the all pole filter) it is usually common to use the rule of thumb ncoeff=2+Fs/1000 but you can pick a number between 8 and 13 here.

5. Find the roots of the LPC (which should represent the formants)

6. Since the LPC are real values, the roots will be complex conjugate, keep only the roots with one of the conjugate complex numbers (either the positives or the negatives). Indeed the goal is to find the angles of these number that represent the frequency values. So 2 complex conjugate will give the same angle and so the same frequencies.

7. Deduce the corresponding Hz values from these complex conjugate (make sure to also sort them at the end in order to obtain increasing values so it is easier to associate them to formants: F1 smallest, F2 > F1 etc…)

MFCC

The MFCCs are coefficients representing the filter of a source filter model. They therefore represent the vocal tract activities during speech production.
We will also implement an algorithm to compute the MFCCs. Please follow the steps enumerated below:

1. The first step is to preamphasise the signal (same as above) but with an α between 0.95 and 0.97. We will take $\alpha = 0.97$ here.

2. The signal should be split into frames

3. A hamming window should be applied to every frame
4. We can now compute the power spectrum of the signal periodogram):
$$P=(\mid FFT(x(t)) \mid^2) / (2N_{TFD})$$
we will take $N_{TFD} = 512$ to calculate P and the FFT.

5. The power spectrum's scale will then be transformed by passing it through a Mel-Filter Bank. The idea behind this is to mimic the ear perception and adjust the scales calculated by the FFT. For this, you will find, on moodle, a function which takes as input power spectrum frames and outputs the corresponding filter bank values.

6. The filter bank coefficients begin very correlated, we apply a Discrete Cosine Transform (dct function from the fftpack sublibrary of the scipy library) on the filter bank coefficients giving the MFCCs vectors. The parameters of the dct should be as follows:
dct(filter_banks, type=2, axis=1, norm='ortho').

7. Of the obtained vector, in general, only the first 13 are kept (the first being equivalent to the energy and the 12 others being the MFCCs).

7.7 Building a Rule-Based System

We will now analyse each of the features extracted from sentences of both of the speakers and define rules to build a speaker recognition system.

From each speaker randomly pic 15 sentences on which the features analysis and extraction will be made

1. Use the algorithms you have written in the previous section to extract features for the 30 sentences. Then study visualize the discriminatory power of these features (considering also the pitch extracted from each algorithm as different features) by comparing their values for both speakers.
2. Based on what you have seen, propose a rule-based system by:
    a. choosing features on which the system will rely
    b. for each feature, state rules that might discriminate between the two speakers.
    ex (if pitch >300 Hz and F1 > 566 Hz: this is SLT)

3. Implement the rule based system you have proposed in 3, in python and test it on the database.

4. To evaluate how well your system works you will rely on the "accuracy" metric. This means that you will compare each output your systems gives to the real class (speaker A or B). The accuracy is then the percentage of utterance well classified. For instance, if the total of test sentences is 10, 5 from speaker A and 5 from speaker B, and if your systems classifies 3 and 4 utterances from speaker A and B respectively correctly then the accuracy is 7/10 = 70%.

5. A next step would be to try to change our system or "fine tune" it in order to obtain higher accuracy results. You will quickly notice that between the amount of possibilities to try by changing the combinations rules/features is infinite. So for the sake of this exercise we will define 3 systems (at least you are free to add more if you desire so) and compare them. Finally pick the one with the highest accuracy value.