BIRZEIT UNIVERSITY
Electrical and Computer Engineering

Spoken Language Processing - Spring 2024
Submission Deadline: Monday 29 April 2024 (Via Moodle only: itc.birzeit.edu)

Instructions: Students should work in groups of two! Each group should submit the outcomes of their work in one PDF file and the required sound files as .wav files. Write down the names and IDs of both students in the group in the top of the PDF file. All files [PDF + wav files] should be submitted as one compressed file on the ITC.

Getting started
Most of the assignment tasks require the Speech Filing Systems (SFS) software, which you can download from this link**:**
https://www.phon.ucl.ac.uk/resource/sfs/download.php

With SFS, you may need a microphone and a headphone to complete this assignment, you need to make sure that they are working, which can be done with the help of SFS. From the start menu select [*Start* | *Speech Filing System* | *SFSWin*].

Several of the most useful menu options can be activated by buttons on the toolbar (as with most Windows applications), e.g., the button with the red circle corresponds to [*Item* | *Record*] on the pull-down menus. Click on the button to set about checking the audio hardware peripherals. First, check the mike by seeing that the recording level gives a response when tested. You want to make sure that the level is not so loud that the signal becomes distorted through clipping, nor too quiet. This can be adjusted by going through the Control Panel to Sound Devices and Volume Control and then to [*Options* | *Properties*] and volume for sound recording. Second, replay the sound signal, check that the headphones are working and adjust the volume by going to [*Options* | *Properties*] and volume for sound playback as required.

Part 1 – Basic sound analysis, Spectrograms
(a) **Bandwidth**
In this warming-up exercise you experience the bandwidth of sound signals. Download the sound files 'sound1.wav' and 'sound2.wav' from the ITC and store them on your own disk space, e.g. C:/. Load each file into SFS and examine its bandwidth. To do this, select from the menu [*Item* | *Import* | *Speech (copy)*] and then browse C:*.wav and read the files in.

Deliverable 1(a): [5]
For each of these files, use the wide-band spectrogram, [*Tools* | *1.Speech* | *Display* | *Wide Spectrogram*], to estimate the lower and upper frequency limits (+/-500Hz) of these sounds.

(b) **Spectrogram**
The aim of this exercise is to gain familiarity with spectrogram representation of a sound signal.
Spectrogram is obtained by splitting the signal into short-time segments and taking the Fourier transform of each segment. There are in general two types of spectrogram: i) wide-band – obtained using a short signal segment and providing (for stationary sounds) a good temporal resolution but poor frequency resolution; ii) narrow-band – obtained using a long signal segment and providing (for stationary sounds) a good frequency resolution but poor temporal resolution.

First, you will analyze spectrogram representation of audio signals. Download the sound files 'sound3.wav' and 'sound4.wav' and store them on your own disk space. Then use the tool [*Tools* | *1.Speech* | *Display* | *Wide and Narrow Spectrogram*]. In the obtained display, the button with a sine-wave displays the waveform and buttons next to it display a wide-band and narrow-band spectrogram.

Deliverable 1(b): [10]
Display the waveform and both the narrow-band and wide-band spectrogram of each sound (make sure you label axes). Comment on the frequency resolution in narrow-band and wide-band spectrograms of each sound. Justify your answers.

Now, you will analyse spectrogram representation of speech signals. In the context of speech signals, wideband spectrogram is convenient for investigating the characteristics of the vocal-tract filter as it provides only crude spectrum and thus a sort of spectral envelope, while narrow-band spectrogram is convenient for investigating the characteristics of the source as it provides good spectral details and thus may shows, for instance, the harmonics.

Record your speech uttering the syllable 'afa' at the sampling rate 16kHz, and export them as a WAV-file to your own disk space (on the C:\ drive). Then use the spectrogram tool in SFS and plot the waveform and both the narrow-band and wide-band spectrogram.

Deliverable 1(c): [10]
Display the waveform and both the narrow-band and wide-band spectrogram of your speech. Describe the main difference between the vocalic (voiced) and consonantal (unvoiced) parts in your syllables in the waveform and in the narrow-band spectrogram.

Part 2 – Filters
The objective of this exercise is to gain a better understanding of the finite impulse response (FIR) digital filters. In this exercise, you have to generate a sinusoidal signal with 1KHz at Fs=16KHz and duration of 3 seconds, and a white noise signal at the same Fs and the same duration. To generate these signals using SFS go to Tools->Generate->Test signals. Then you need to add the two signals together into one signal. You can do this outside SFS using Matlab by exporting each signal as .wav file and then read each file into Matlab (audioread() function), then add two signals and write the output as a wave file (audiowrite() function).

Then, you have to design and implement (using Matlab) an FIR filter and then use it to filter the output wav-file (1KHz sinusoidal corrupted by a white noise). Using Matlab, apply it to an FIR filter with impulse response {1, -1}. You can listen to the input and output signal by typing in the Matlab command window 'sound(inpSigWav,Fs)' and 'sound(outSigWav,Fs)',where impSigWav and outSigWav are input and output signals of the filter, respectively.

The second task in this exercise is to perform filtering using another filter defined by the following difference equation:

y(n) = b0*x(n)+b1*x(n-1) +b2*x(n-2) +b3*x(n-3) +b4*x(n-4) +b5*x(n-5) +b6*x(n-6) +b7*x(n-7)

where all the coefficients from b0 to b7 are set to the value 0.5.

After you have performed the above filtering operations, calculate the magnitude frequency characteristic of each filter by using the fast Fourier transform. This can be obtained by typing in Matlab command window 'magFreqChar=abs(fft(impRes,256));' and then 'plot(20*log10(magFreqChar(1:128)));' where you need to define the variable 'impRes' which denotes the impulse response of the filter arranged as a 1-dimensional vector (e.g., impRes = [1, -1] in the above example) and 256 is the number of frequency points the entire frequency range from 0 to Fs is divided into (note that the impRes is automatically appended by zeros to the length of 256).

Deliverable 2: [10]
Plot the waveforms of the input and output signals. Plot the magnitude frequency characteristics of each filter. Comment, with detailed justification, on the strength of the sine-wave and noise in the output signal obtained by each filter.

Part 3 – Speech Analysis
The objective of this exercise is to gain a better understanding of the source-filter model of speech production. Based on this model, speech is produced by passing a source-signal through a filter which reflects the characteristics of the vocal-tract. The vocal-tract can be characterized by its resonant frequencies, called formants. These can be observed as peaks in the estimated filter spectrum or peaks of the spectral envelope.
Use the tool *[Tools | 1.Speech | Display | Cross Section]* to estimate the spectrum of the source-signal and vocal-tract filter in the source-filter model (this tool employs linear predictive analysis which we will briefly talk about in one of the lectures). The spectrum of the source, filter, and speech can be displayed by pressing the button (S), (F), and (O), respectively. You can select a segment of the signal the analysis is performed on by making left and right mouse clicks in the waveform (to mark the beginning and the end of the segment).

(a) Formants for vowels
In this part, you will measure the formant frequencies of vowels and investigate how formant frequencies vary for different vowels and for the same vowel across different speakers. Here, you will use the (F) display to obtain the formant frequencies. Note that the tool provides estimates of the formant frequencies at the right bottom corner, however, these

may sometimes contain an error (added or missed formant) and thus you should verify these estimates based on the displayed filter spectrum.
Use the speech file 'sample1.wav' which contains saying "She had your dark suit in greasy wash water all year" and estimate the formant frequencies F1, F2, and F3 of four vowels: / i: / in 'She'; / { / in 'had'; / u: / in 'suit'; / A: / in 'dark'. Select the middle part of the vowel (this should be at least 20ms long). Then make 2 recordings of the same sentence (spoken by you and another student) and analyze the formant frequencies.

Deliverable 3(a): [20]
Write down the frequencies of the first three formants F1, F2 and F3 for the four vowels when using the file 'sample1.wav' and your own recorded sentences. Plot F1 versus F2-F1 for these four vowels and comment on how the results of your own speech compare to the given speech file and to the vowel quadrilateral in your lecture notes.

(b) Source and Filter analysis
Perform the source-filter analysis for phoneme / s / from the word 'suit' and / A: / from the word 'dark' (use the given file 'sample1.wav').

Deliverable 3(b): [10]
Depict figure of the source (S) spectrum and filter (F) spectrum of the / A: / and / s /. What are the differences in production of these sounds? Discuss how these differences are reflected in the source (S) spectrum and filter (F) spectrum.

(c) Fundamental Frequency
Choose an appropriate section from your recorded sentence in part 3(a) and determine the fundamental frequency of your voice.

Deliverable 3(c): [10]
Print the waveform of the selected section of the speech signal. Write down an estimate of the fundamental frequency (+ explain your workings).

Part 4 – Speech analysis

Download from ITC the sound file AS1.wav and store it on your hard drive. View the waveform of the speech using SFS. Select (using the left and right mouse clicks) the word "capacity", which lies in the range 3290 – 3800 ms and copy the selected section using [Segment | Copy]. The word "capacity" can be phonetically transcribed using the SAMPA convention as: / k @ p { s I t i /.
Your task is to identify the start and end times of each phoneme. Note that the start time for the phoneme /k/ is considered as 0 ms. You will use the estimated start/end times in Part 6 to perform phoneme-level waveform concatenation.

Deliverable 4: [15]
Identify precisely the start and end times of each of the eight phonemes in this word (+/- 10 ms).

Part 5 – Sub-word level concatenation

Record the words "life" and "night", and the vowel sound "ah", using the carrier phrase "Say .. again" at a sampling rate of 16 kHz. In other words, record the utterance "Say life again", then view the waveform, select the part corresponding to the word "life" /laIf/ and using [Segment | Copy] obtain just this section, and export it to .wav file on your hard drive in the main SFS window using the tool [Tools | 1.Speech | Export | Make WAV file]. Then, repeat the recording of the sentence with word "night" /naIt/, and with isolated vowel "ah" /a/. The carrier phrase helps to ensure that your breathing and emphasis are balanced for the recording of each word. The idea is to generate a waveform for the new word "light" /laIt/ using parts of the signal "life" and "night" that you just recorded (note that the recording of "ah" will be used later in Part 6 of the lab). You will have to select parts to copy from your existing recordings, and be careful about how you join the two together, to avoid nasty glitches in the final sound file. This means of synthesising words is referred to as concatenative synthesis, because you concatenate, or glue together, parts of speech signals. It will take a bit of trial and error, and some fine adjustment of the waveform to get something that sounds satisfactory, and you may find it helpful to use the spectrograms in SFS to decide the best place to make the join.
The concatenation of the waveforms can be performed using Matlab or any special software.

You can also experiment and try to make another word from a different combination, e.g., "knife", "tight", or mixing recordings from two speakers.

Deliverable 5: [15]

Store and submit the sound files containing the original words "life", "night" and "ah" and the synthesised word "light", made by splicing recorded waveforms together. Comment on any difficulties you encounter or factors which you have to take care of in order to produce a good quality speech by waveform concatenation method.

Part 6 – Phone-level concatenation

Since phonemes can be considered as the basic units of any speech utterance, for a general purpose concatenative speech synthesiser we might like to use recordings of the individual sounds as the building blocks. These phones ought to give us the flexibility to generate any word of the language. In Part 4, you were asked to identify in the word "capacity" the start and end times of each of the eight phonemes: /k/, /@/, /p/, /{/, /s/, /I/, /t/, and /i/. Now, you are going to use these parts of the signal to try to make a completely different word.

You can find the phonetic transcription of any word using CMU dictionary (http://www.speech.cs.cmu.edu/cgi-bin/cmudict) or any other tool/dictionary. Choose one word you want to synthesise. To do the synthesis, select the appropriate sections of the original waveform, and save them into individual .wav files (export as .wav tool as in Part 5 above).

Deliverable 6: [15]

Store and submit the sound file containing the synthesised word. Comment on the quality of the resulting speech, and how it could be improved.

Part 7 – Generation of vowel sounds using the source-filter model (parallel formant synthesiser)

The source-filter model of speech production expresses that a speech signal can be generated by passing a source-signal through a filter (corresponding to the vocal-tract). In this exercise you will experience the use of the source-filter model to generate three vowel-sounds: /A:/ as in "hard"; /u:/ as in "who'd" and /ə/ as in "heard".

First, use the tool [Tools | Generate | Test Signals] to generate the source-signal: tick the option 'Signal Type' and choose 'pulse-train', tick the option 'Frequency' and set to 110Hz, and tick the option 'Sampling Frequency' and set to 8000. This will generate a periodic pulse-train signal of frequency 110Hz which is somewhat similar to the source-signal produced by the vibrating vocal-cords when producing voiced sounds.

Now, you will pass the source-signal through a filter defining the vocal-tract to generate a vowel sound. For a given vowel, the vocal-tract filter will be defined by the first two formant frequencies – these can be found out from the vowel-space diagram in lecture 5 or from your own observations in Lab1. Approximate values from lecture 5 are as follows: /A:/ F1=700Hz, F2=1100Hz; /u:/ F1=300Hz, F2=800Hz; /ə/ F1=500Hz, F2=1500Hz. The effect of the vocal-tract filter will be simulated using two band-pass filters arranged in a parallel configuration, i.e., you will filter the source-signal by two separate band-pass filters, each corresponding to one formant frequency. To filter the source-signal by a band-pass filter use the tool [Tools | 1.Speech | Process | Filtering | Band-Pass Filter]

– tick the 'Lower band-pass frequency edge' and 'Higher band-pass frequency edge' and set to a value F-200 and F+200, respectively, where F is the formant frequency value. Filter the source-signal separately by a bandpass filter corresponding to the formant frequency F1 and then F2. This gives you two signals. Export each of the signal into a wav-file by using the tool [Tools | 1.Speech | Export | Make WAV file].

To produce the final speech signal you need for each sample-index to sum the sample values of each of the signal, i.e., perform y(n)=sig1(n)+sig2(n), where 'n' is the sample-index ('n=1 to N' and 'N' is the number of samples in each of the signal) and 'sig1' and 'sig2' denotes the two signals. You will perform this operation using

Matlab. First, load the individual wav-files using the 'audioread' function (see the m-file given in Lab1 for the use of the function or type 'help audioread' for help) – the sample-values of each signal are stored in a 1D array variable.

Then perform the summation of the signals as outlined above. Finally, use the 'audiowrite' function to store the final signal in the wav-file format. Assuming the variable 'outSigWav' is a 1D array storing the summed signal samples and the variable 'fileNameOut' stores the filename (including full path) of the output wav-signal (this can be created by typing, e.g., 'fileNameOut='y:\genWovel1.wav''), the 'audiowrite' function is called by typing

'audiowrite(outSigWav, fileNameOut);'.

For your interest, you can repeat the generation of the vowels by using the white-noise as the source signal. This will generate the same vowel sounds by pronounced by a whispery voice (i.e., the vocal-folds would not vibrate).

Deliverable 7: [15]

Store and submit the wav-files of the three generated vowel sounds.

Part 8 – Formant synthesis in SFS

SFS can be used to drive a formant synthesiser, using parameters extracted from a recorded speech file. To do so, take your file with the word “life” /laIf/ and use the tool [Tools | 1.Speech | Analysis | Formants] that performs formant analysis. Use the option [Tools | 12.Formants | Export formant data] to export the output formant analysis into a text file. Look at the formant data in the file with a text editor, such as Notepad, and check that the entries correspond to what you saw displayed on the screen. The next step is to generate synthesizer control parameters from the formant data – use the option [Tools | 12.Formants | Generate synthesiser control data]. Finally, use these parameters to synthesise a speech signal – use the option [Tools | 7.Synthesis Data |
Synthesise speech]. Play the artificial speech that is produced.

To demonstrate the significance of formants for characterising different phonemes, use the formant values for the /a/ vowel (estimated from your “ah” sound recorded in Part 2 of this lab using the tool [Tools | 1.Speech | Display | Cross-section]) throughout the diphthong /aI/, i.e., for the vowel part up until voice offset (when voicing stops for the voiceless fricative /f/). Copy the exported formant data to a new filename, and modify the formant frequencies for F1 and F2 accordingly, so that the entries in the new file are constant throughout the diphthong section and equal to the first two formants for the vowel “ah”.

Then use the option [Item | Import | Formant] to import the modified formant parameters. Then repeat the above process to regenerate the synthesiser control parameters (use [Tools | 12.Formants | Generate synthesizer control data]) and the synthetic speech waveform (use [Tools | 7.Synthesis Data | Synthesise speech]). Play the artificial speech that is produced.

Deliverable 8(a): [15]

Store and submit the sound file of the synthesised speech when using the original formant data and modified formant data. Can you identify the synthesised speech produced by the modified formant data – write down what it says (give explanation)? Plot figures of waveform and wide-band spectrogram of all three (original, synthesized original and synthesised modified) speech sounds.

Deliverable 8(b): [5]

Describe briefly what needs to be done if we want to change the speaker’s voice in a formant-based speech synthesiser?