Date: 18th November 2021

Place: Coimbatore

AMRITA VISHWA VIDHYAPEETHAM

ETTIMADAI

B.Tech Electrical and Computer Engineering (2019-2023)

Digital Signal Processing – Mini project

Bidirectional Aid for partially dumb

CB.EN.U4ELC19031 – Naresh R

CB.EN.U4ELC19041 – Rathin Raj R S

Introduction:

Conversations are key to exchange of thoughts, ideas and listening to each other. We cannot imagine a life without conversing. But not everyone is gifted with that comfort. Whatever we speak, can be divided into fundamental phonetics that make up the overall sound of any basic word. And each phonetic is mapped with a particular frequency an individual is able to vibrate at. But, as said earlier, few people, for example, autistic people, try pronouncing words in their own range of frequencies which does not satisfy the levels required for an actual pronunciation.

So, our aim was to create an aid for this problem statement. We collected the universal words and their respective phonetics and proceeded to filter out only the unique phonetics present. Now, a context implementing all the phonetics with day-to-day words was created.

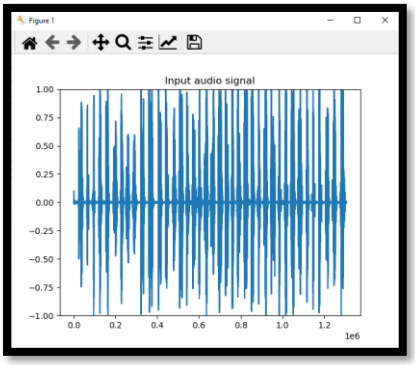
The person is now required to record (which is also in-built in code) by pronouncing all the words in the specified order. We get this and use further techniques like STFT, FFT, Threshold filtering to map their frequency to that respective phonetic. Thus, in the future, when an unknown audio input of the partially dumb person is given, the word they're actually trying to pronounce can be predicted with this process and its phonetic representation is displayed.

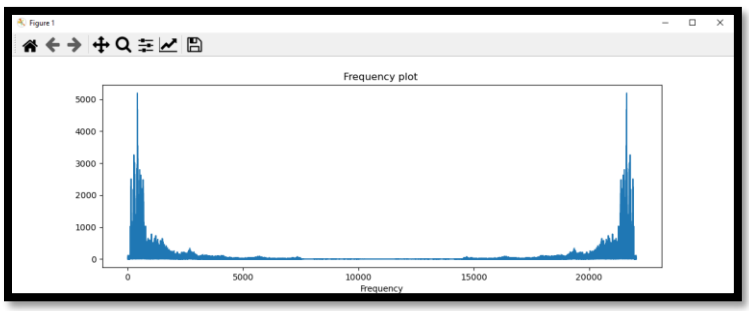
Concepts Involved:

- **STFT (Short Time Fourier Transform):** As the input given to the system is a sequence of words all together. We have to first split the sequence into words into chunks of audio and later separate out the phonetic associated with it. For this, STFT is used to determine the sinusoidal frequency of local sections
    - The procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment.
- **Filtering:** The spectral input from STFT is later fed into a function that calculated the decibels from the amplitude using the formula: dB = 20 * log10(amplitude)
    - Further, non-mute areas are identified and filtered into timestamps by dividing the timeframe with sampling rate.
- **Down Sampling:** The number of samples of a audio file is reduced without losing the features of it's originality.
    - Down sampling is done to speed up the following FFT process by reducing the number of samples
- **FFT (Fast Fourier Transform):** Using FFT, the audio signal which we have as chunks, only the phonetic part is split and FFT is applied over that time domain signal. After FFT, the time domain signal is now obtained in frequency domain which does justice to the process of predicting.
    - Now, the fundamental frequency of the resultant FFT is calculated and mapped to the respective phonetic it's supposed to represent
    - $x[K]=\sum_{n=0}^{N-1} x[n] W_N^{nk}$ (library was used for implementation)
- **Thresholding:** Once the mapping is done, for the working of this system, any test audio of the person is given as an input, where **Thresholding** is performed to only find the significant parts of the input and further work on this result.

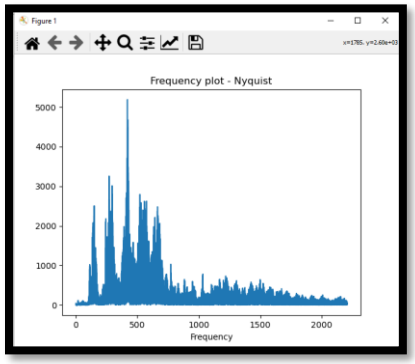
Analysis and Results:

Input audio and Frequency plots:



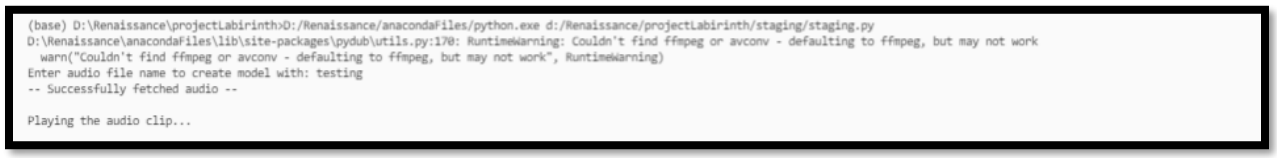


Nyquits Plot:



Extracting unique phonetics from lakhs of input words:

Staging of audio:



STFT:

```
(base) D:\Renaissance\projectLabirinth>D:/Renaissance/anacondaFiles/python.exe d:/Renaissance/projectLabirinth/staging/decibels.py
Enter audio file name to create model with: demo
(1025, 1277)
STFT Results:
[[-5.1416930e-02+0.0000000e+00j  7.8207046e-01+0.0000000e+00j
  -1.4404348e-01+0.0000000e+00j ...  7.9855554e-02+0.0000000e+00j
   2.5468466e-01+0.0000000e+00j  2.9545819e-02+0.0000000e+00j]
 [ 2.6419200e-02+4.9974944e-18j -1.2878484e+00-1.4759780e+00j
  -5.5248386e-01+1.7570966e-01j ... -3.0777988e-01-1.9476271e-01j
  -4.7976547e-01+7.9874106e-02j  3.1382521e-03+3.0721176e-01j]
 [ 1.6576524e-01+4.0933872e-18j  2.0047958e+00+2.2927787e+00j
   8.0179036e-01+2.7561592e-02j ...  4.3743363e-01+3.5762089e-01j
   4.9720231e-01-1.1646134e-01j  1.1235791e-01-1.8117125e-01j]
 ...
 [ 1.2549629e-05-5.6112703e-18j -1.1420080e-06-3.2979716e-07j
  -2.6785929e-06+8.1447757e-07j ... -1.3504381e-06+1.9356658e-06j
  -3.4196214e-07-3.1872497e-07j  4.8270535e-06+1.0386477e-06j]
 [-1.2563401e-05-4.4418408e-18j  9.4097516e-07-3.1393978e-07j
   2.7226440e-06-3.9813077e-07j ... -1.7423241e-07-3.8621272e-07j
   1.2601532e-07+1.2016991e-06j  1.5293006e-07+3.1109782e-06j]
 [ 1.2202112e-05+0.0000000e+00j -9.3328038e-07+0.0000000e+00j
  -2.3199320e-06+0.0000000e+00j ...  2.0409782e-06+0.0000000e+00j
   1.0459106e-08+0.0000000e+00j -4.7504464e-06+0.0000000e+00j]]
```

Decibel Filtering with resultant timestamps:

```
Highest decibel noticed:  80.0
Non mute locations in the provided audio:  [[  23552   45568]
 [  64000   70144]
 [  71168   73728]
 [  90112  101888]
 [ 120320  135680]
 [ 153088  170496]
 [ 189440  210432]
 [ 225280  238592]
 [ 249856  268288]
 [ 282112  301056]
 [ 321536  339968]
 [ 357888  380416]
 [ 404480  422912]
 [ 440320  455168]
 [ 473088  492032]
 [ 503296  526336]
 [ 537088  555008]
 [ 566272  587776]
 [ 596480  613888]
 [ 625664  644608]
 [ 654848  676352]
 [ 691200  713728]
```

```
[ 726016  737280]
[ 739328  742912]
[ 755200  768512]
[ 782336  802816]
[ 814592  835584]
[ 850432  866816]
[ 879104  894976]
[ 909824  930304]
[ 943616  953344]
[ 954368  961024]
[ 962048  9E4608]
[ 978944  991744]
[1009664 1026048]
[1038336 1060864]
[1072128 1095680]
[1114112 1126400]
[1140224 1152000]
[1154048 1156096]
[1173504 1184768]
[1186816 1188864]
[1205248 1224192]
[1238528 1261056]
[1276416 1301504]]
Time frames of nonMute audio:
[[1.068, 2.067], [2.902, 3.181], [3.228, 3.344], [4.087, 4.621], [5.457, 6.153], [6.943, 7.732], [8.591, 9.543], [10.217, 10.82], [11.331, 12.167], [12.794, 13.653], [14.582, 15.418], [16
.231, 17.252], [18.344, 19.18], [19.969, 20.643], [21.455, 22.314], [22.825, 23.87], [24.358, 25.17], [25.681, 26.657], [27.051, 27.841], [28.375, 29.234], [29.698, 30.674], [31.347, 32.3
69], [32.926, 33.437], [33.53, 33.692], [34.249, 34.853], [35.48, 36.409], [36.943, 37.895], [38.568, 39.311], [39.869, 40.588], [41.262, 42.191], [42.794, 43.236], [43.282, 43.584], [43.
63, 43.746], [44.397, 44.977], [45.79, 46.533], [47.09, 48.112], [48.623, 49.691], [50.527, 51.084], [51.711, 52.245], [52.338, 52.431], [53.22, 53.731], [53.824, 53.917], [54.66, 55.519]
, [56.169, 57.191], [57.887, 59.025]]
45
```

Applying FFT and mapping:

```
Final array:  {'EH': 133.6862206923443, 'K': 115.41681757952966, 'S': 166.3766182986707, 'L': 246.24666602711073, 'AH': 550.9053733200344, 'M
': 163.75147076461985, 'EY': 64.22858384983478, 'SH': 135.25021599927993, 'N': 67.85560738344809, 'P': 51.28154734401888, 'OY': 61.86294254657
5546, 'T': 223.9305821258729, 'OW': 153.13722949769465, 'Z': 120.17002626981548, 'W': 167.5091249640809, 'D': 230.00239979309444, 'B': 57.085
98756681616, 'V': 150.97755339586604, 'IH': 169.0366257795551, 'HH': 399.70197167172057, 'AE': 9.786848284547453, 'AA': 143.30905102852168, '
R': 153.70890200491695, 'AW': 95.68533144146848, 'AY': 118.64637180566885, 'ER': 129.40138649488117, 'F': 87.55752520824709, 'IY': 210.913081
02765103, 'AO': 38.79238385649154, 'Y': 10.970079382986441, 'UW': 593.3626115779267, 'G': 46.327043296902, 'NG': 46.967571218986194, 'TH': 33
.972669932758336, 'DH': 311.4802923015342, 'UH': 83.82096180510632, 'CH': 105.37202805634627, 'ZH': 468.3467434419365, 'JH': 29.7768964001534
08}
```

Input audio: Pronunciation of the word **“Hello”**

Result:

```
(base) D:\Renaissance\projectLabirinth>D:/Renaissance/anacondaFiles/python.exe d:/Renaissance/projectLabirinth/testing/testing.py
D:\Renaissance\anacondaFiles\lib\site-packages\pydub\utils.py:170: RuntimeWarning: Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
Enter audio file name to predict input with: testaud

Predicted phoenetic sequence:
JH AE Y UW UH
```

Inference:

- Large audio can be split and STFT can be used for the audio chunks after splitting to find the frequency domain of the same
- The highest decibel found in the given audio was 80db. And the highest decibel to consider non-mute was set to be 30db which changes for every microphone input
- Down sampling was not found to be efficient in our case, so, it was removed for the further processing of the audio signal
- FFT converted the audio signal in time domain to frequency domain for mapping the fundamental frequency of respective phonetic

- When an unheard audio signal was given for testing, the resultant phonetic almost matched the pronunciation of the actual signal. Implementation of ML gives better results to the same.

References:

1. *Phonetic collection: http://www.speech.cs.cmu.edu/cgi-bin/cmudict*
2. *Audio Processing: https://medium.com/@vvk.victory/audio-processing-librosa-split-on-silence-8e1edab07bbb*
3. *Lecturer Notes*