

# Malayalam Text-to Speech system

**Guide: Jiby J P**

**Project Coordinator: Manilal D.L**

Group 16

Kurian Benoy 34

[kurianbenoy@mec.ac.in](mailto:kurianbenoy@mec.ac.in)

# Objective

- To build a Text to Speech(TTS) system in Malayalam
- Obtain the state of art result

# Contents

- Introduction
- Text to Speech
- History
- Modules
- Work Done
  - Dataset Collection
  - Text to speech system in English
  - Exploratory data analysis

# Introduction

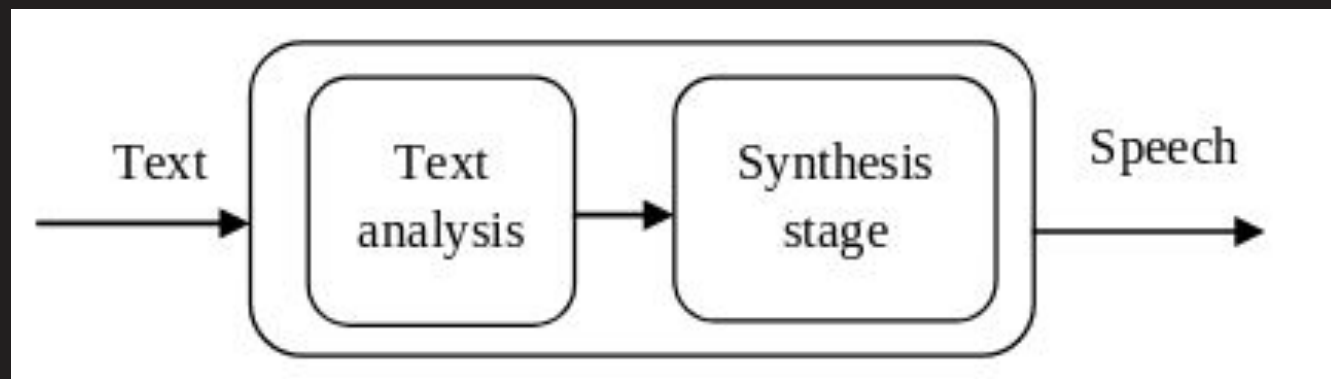
# Text to speech

Text to speech systems convert any written text into spoken speech. Text-to-speech systems is a vital step for accessibility to disabled people like blind, and deaf. It can be used in lot of educational applications as well. Most of the text-to-speech systems are currently made for English.

# Text to speech

TTS consists of two parts usually:

- The front-end consists of converting a text by text normalization, pre-processing, or tokenization and converting into graphemes.
- The back-end, referred to as the synthesizer, which converts the symbolic linguistic representation into sound.



# History

In 1779, the German-Danish scientist Christian Gottlieb Kratzenstein received the first prize in a competition declared by the Russian Imperial Academy of Sciences and Arts for the models he had designed of the human vocal tract that could generate the five long vowel sounds (International Phonetic Alphabet Notation: [ a ], [ e ], [ I ], [ o ] and [ u]). The bellows-operated "acoustic-mechanical speech machine" by Wolfgang von Kempelen of Pressburg, Hungary, described in a 1791 article[2], followed by adding models of tongues and lips. This allowed it to produce consonants as well as voices. Charles Wheatstone created a "talking machine" based on von Kempelen's design in 1837. Wheatstone's model was a bit more complicated and was capable to produce vowels and most of the consonant sounds. Some sound combinations and even full words were also possible to produce. Vowels were produced with vibrating reed and all passages were closed. Resonances were effected by the leather resonator like in von Kempelen's machine. Consonants, including nasals, were produced with turbulent flow through a suitable passage with reed-off. Joseph Faber exhibited the "Euphonia" in 1846. Paget revived Wheatstone's concept in 1923.



# History

In the 1930s, Bell Labs developed a vocoder that automatically analyzed speech in its fundamental tones and resonances.

Homer Dudley developed a keyboard-operated voice-synthesizer called The Voder (Voice Demonstrator), which he exhibited at the 1939 New York World Fair. Dr. Franklin S. Cooper and his colleagues at the Haskins Laboratories designed the Pattern Playback in the late 1940s and completed it in 1950. There have been several different versions of this hardware device; only one currently survives. It reconverted recorded spectrogram patterns into sounds, either in original or modified form. The spectrogram patterns were recorded optically on the transparent belt.

# History

The first formant synthesizer, PAT (Parametric Artificial Talker), was introduced by Walter Lawrence in 1953 (Klatt 1987). PAT consisted of three electronic formant resonators connected in parallel. The input signal was either a buzz or noise. A moving glass slide was used to convert painted patterns into six time functions to control the three formant frequencies, voicing amplitude, fundamental frequency, and noise amplitude (track 03). At about the same time Gunnar Fant introduced the first cascade formant synthesizer OVE I (Orator Verbis Electris) which consisted of formant resonators connected in cascade (track 04). Ten years later, in 1962, Fant and Martony introduced an improved OVE II synthesizer, which consisted of separate parts to model the transfer function of the vocal tract for vowels, nasals, and obstruent consonants. Possible excitations were voicing, aspiration noise, and frication noise. The OVE projects were followed by OVE III and GLOVE at the Kungliga Tekniska Högskolan (KTH), Swede. (as mentioned in [1])

# Modules

- Module1 : EDA, dataset collection
- Module2: Train first TTS system in Malayalam
- Module3: Fine tune TTS system
- Module4: User Interface

**Work Done**

# Dataset collection

1. Malayalam Speech Corpora, which was initiated to create high quality dataset under SMC. The recording platform can be found at <https://msc.smc.org> and dataset can be downloaded from: <https://gitlab.com/smc/msc>
2. Crowdsourced high-quality Malayalam multi-speaker speech data set by openslr.org  
Dataset can be found: <http://openslr.org/63/> and is licensed under Attribution-ShareAlike 4.0 International

# Dataset collection

3. The corpus contains 10 words in Malayalam corresponding to 10 digits (0-9) in English. These words are uttered by 10 speakers include 6 females and 4 males of age ranging from 15 to 40. Every speaker gives 10 trials of each word and thus have 100 samples per speaker. Signals are recorded with a sampling frequency of 8 KHz. This dataset was Mini P.P etc. and licensed under CC.4.0

<https://data.mendeley.com/datasets/5kg453tsjw>

# Text to Speech system in English

Using Tacotron2 architecture made a TTS system in English using pretrained models from Mozilla/TTS. TTS used Tacotron2 architecture made a TTS system in English using pretrained models from Mozilla/TTS. TTS aims a deep learning based Text2Speech engine, low in cost and high in quality.

TTS includes two different model implementations which are based on [Tacotron](#) and [Tacotron2](#). Tacotron is smaller, efficient and easier to train but Tacotron2 provides better results, especially when it is combined with a Neural vocoder. Therefore, choose depending on your project requirements.



# Text to Speech system in English

## Training Notebook -

<https://colab.research.google.com/drive/1Raaiags-RkFako1HJ0tXSW-EnWkvX84x>

```

LJSpeech_training.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[ ] # pull the trigger
!python train.py --config_path config.json | tee training.log

|> symmetric norm:True
|> mel_fmin:0
|> mel_fmax:8000.0
|> max_norm:4.0
|> clip_norm:True
|> do_trim_silence:True
|> sound_norm:False
|> n_fft:2048
|> hop_length:275
|> win_length:1100
|> Using model: Tacotron2
|> Num output units : 1025

> Model has 28921234 parameters
> Number of outputs per iteration: 7

|> DataLoader initialization
|> Use phonemes: True
|> |> phoneme language: en-us
|> |> Number of instances: 12090
|> |> Max length sequence: 187
|> |> Min length sequence: 5
|> |> Avg length sequence: 98.20591666666667
|> |> Num. instances discarded by max-min (max=150, min=6) seq limits: 572
|> |> Batch group size: 0.

> Epoch 0/2
|> Step:24/178 GlobalStep:25 PostNetLoss:3.87938 DecoderLoss:3.97822 StopLoss:3.45789 AlignScore:0.0689 GradNorm:2.43576 GradNormST:4.84735 AvgTextLen:56.6 AvgSpecLen:312.3 StepTime:1.38 LoaderTime:61.23
|> Step:49/178 GlobalStep:50 PostNetLoss:3.83943 DecoderLoss:3.65129 StopLoss:3.22181 AlignScore:0.0108 GradNorm:4.59911 GradNormST:4.54263 AvgTextLen:73.5 AvgSpecLen:415.0 StepTime:1.39 LoaderTime:4.86
|> Step:74/178 GlobalStep:75 PostNetLoss:2.60457 DecoderLoss:2.62243 StopLoss:2.52114 AlignScore:0.0092 GradNorm:2.95845 GradNormST:0.94370 AvgTextLen:86.3 AvgSpecLen:497.7 StepTime:1.25 LoaderTime:0.05
|> Step:99/178 GlobalStep:100 PostNetLoss:1.97698 DecoderLoss:1.71418 StopLoss:2.35038 AlignScore:0.0100 GradNorm:0.70443 GradNormST:15.07312 AvgTextLen:97.4 AvgSpecLen:555.3 StepTime:1.52 LoaderTime:0.07
|> Step:124/178 GlobalStep:125 PostNetLoss:1.77552 DecoderLoss:1.60470 StopLoss:1.94877 AlignScore:0.0085 GradNorm:0.85603 GradNormST:8.08014 AvgTextLen:111.1 AvgSpecLen:627.3 StepTime:1.51 LoaderTime:71.3
|> Step:149/178 GlobalStep:150 PostNetLoss:1.66202 DecoderLoss:1.59207 StopLoss:1.75909 AlignScore:0.0082 GradNorm:0.56687 GradNormST:8.36113 AvgTextLen:122.6 AvgSpecLen:691.6 StepTime:1.07 LoaderTime:23.0
|> Step:174/178 GlobalStep:175 PostNetLoss:1.60542 DecoderLoss:1.49736 StopLoss:1.32455 AlignScore:0.0077 GradNorm:0.55848 GradNormST:6.35953 AvgTextLen:138.6 AvgSpecLen:741.4 StepTime:1.70 LoaderTime:0.08
|> EPOCH END -- GlobalStep:179 AvgPostNetLoss:3.06231 AvgDecoderLoss:2.52719 AvgStopLoss:2.58456 AvgAlignScore:0.009511 EpochTime:267.31 AvgStepTime:1.49 AvgLoaderTime:19.76

> Validation
|> |> TotalLoss: 6.98393 PostNetLoss: 2.00195 - 2.00195 DecoderLoss:1.58287 - 1.58287 StopLoss: 3.39912 - 3.39912 AlignScore: 0.0250 : 0.0250
|> |> TotalLoss: 9.34040 PostNetLoss: 2.12476 - 2.06420 DecoderLoss:1.34097 - 1.59993 StopLoss: 1.67467 - 2.39601 AlignScore: 0.0106 : 0.0134
|> |> TotalLoss: 5.24663 PostNetLoss: 2.05901 - 2.06008 DecoderLoss:1.47117 - 1.50731 StopLoss: 1.71645 - 2.24702 AlignScore: 0.0093 : 0.0113
warning: audio amplitude out of range, auto clipped
|> |> Training Loss: 3.06231 Validation Loss: 2.07321

> BEST MODEL (2.07321) < ./.ljspeech-graves-January-23-2020_03+474M-8dfed6/best_model.pth.tar
> Number of outputs per iteration: 5

> Epoch 1/2
|> Step:20/178 GlobalStep:200 PostNetLoss:3.10644 DecoderLoss:1.54221 StopLoss:4.08622 AlignScore:0.0088 GradNorm:8.46261 GradNormST:59.62710 AvgTextLen:51.9 AvgSpecLen:287.1 StepTime:2.63 LoaderTime:5.12
|> Step:45/178 GlobalStep:225 PostNetLoss:1.46624 DecoderLoss:1.45260 StopLoss:2.51499 AlignScore:0.0101 GradNorm:3.84645 GradNormST:17.94497 AvgTextLen:71.5 AvgSpecLen:400.3 StepTime:2.96 LoaderTime:0.07
|> Step:70/178 GlobalStep:250 PostNetLoss:1.20559 DecoderLoss:1.38109 StopLoss:2.23664 AlignScore:0.0097 GradNorm:1.28384 GradNormST:11.80802 AvgTextLen:80.4 AvgSpecLen:489.0 StepTime:3.10 LoaderTime:0.08
|> Step:95/178 GlobalStep:275 PostNetLoss:1.21434 DecoderLoss:1.30000 StopLoss:1.84824 AlignScore:0.0087 GradNorm:1.95358 GradNormST:8.07510 AvgTextLen:97.5 AvgSpecLen:549.9 StepTime:3.22 LoaderTime:0.12
|> Step:120/178 GlobalStep:300 PostNetLoss:1.17680 DecoderLoss:1.24754 StopLoss:1.44174 AlignScore:0.0090 GradNorm:0.78482 GradNormST:9.71685 AvgTextLen:108.7 AvgSpecLen:632.5 StepTime:3.70 LoaderTime:0.10
|> Step:145/178 GlobalStep:325 PostNetLoss:1.16431 DecoderLoss:1.16431 StopLoss:1.14721 AlignScore:0.0090 GradNorm:1.40870 GradNormST:5.33443 AvgTextLen:121.0 AvgSpecLen:682.4 StepTime:2.40 LoaderTime:0.14
```

# Text to Speech system in English

## Inference Notebook <https://colab.research.google.com/drive/1pyS5yQAe3UlpCV7q1boTqgyBgHCHID>

```
[ ] SENTENCE = 'Bill got in the habit of asking himself "Is that thought true?" And if he wasn't absolutely certain it was, he just let it go.'
```

### Synthesize

```
[ ] align, spec, stop_tokens, wav = tts(model, SENTENCE, CONFIG, use_cuda, ap, speaker_id=0, use_gl=False, figures=False)
```

```
[ ] 180000/181500 -- batch_size: 15 -- gen_rate: 12.2 kHz -- x_realtime: 0.6 > Run-time: 23.488433837890625
```

▶ ● 0:00:00 / 27:03:12 🔊

```
[ ] subin = '3, 2,1 and go.The biggest news in 2019 has to be Janayugoms move to Scribus this is a landmark event since this is the first time an entirely free software based stack is used to produce a newspaper'
```

```
[ ] %time  
align, spec, stop_tokens, wav = tts(model, subin, CONFIG, use_cuda, ap, speaker_id=1, use_gl=False, figures=False)
```

```
[ ] | > Decoder stopped with 'max_decoder_steps  
576000/580800 -- batch_size: 48 -- gen_rate: 40.0 kHz -- x_realtime: 1.8 > Run-time: 37.85575842857361
```

▶ ● 0:00:04 / 27:03:12 🔊

CPU times: user 37.2 s, sys: 683 ms, total: 37.9 s  
Wall time: 38 s

# Exploratory Data Analysis

SampleRate, Signal length, duration

```
In [12]: y, sr = librosa.load("data/slr/male/mlm_00269_00156195788.wav")
tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr)

print('Estimated tempo: {:.2f} beats per minute'.format(tempo))

beat_times = librosa.frames_to_time(beat_frames, sr=sr)
print(beat_times)
```

Estimated tempo: 161.50 beats per minute  
[0.23219955 0.60371882 0.9752381 1.34675737 1.69505669 2.04335601  
2.41487528 2.7631746 3.13469388 3.50621315 3.87773243 4.22603175]

```
In [13]: print(f"Sample rate :", sr)
print(f"Signal Length:{len(y)}")
print(f"Duration : {len(y)/sr}seconds")
```

Sample rate : 22050  
Signal Length:109778  
Duration : 4.97859410430839seconds

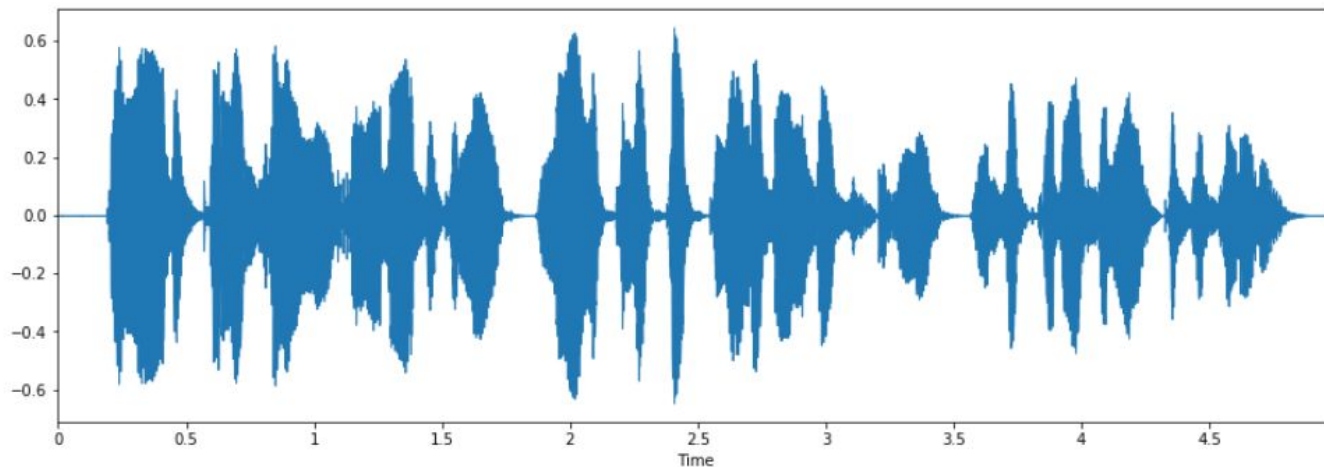
---

# Exploratory Data Analysis

## Waveform display

```
In [15]: import librosa.display  
plt.figure(figsize=(15, 5))  
librosa.display.waveplot(y, sr=sr)
```

```
Out[15]: <matplotlib.collections.PolyCollection at 0x7fcc5d60dbe0>
```

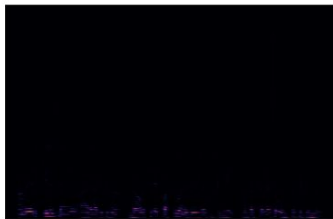


# Exploratory Data Analysis

## Mel Spectrogram

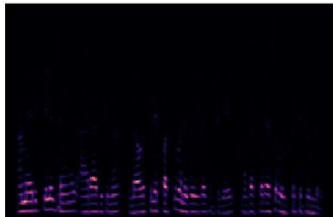
```
In [16]: sg0 = librosa.stft(y)
sg_mag, sg_phase = librosa.magphase(sg0)
display(librosa.display.specshow(sg_mag))

<matplotlib.axes._subplots.AxesSubplot at 0x7fcc5c561860>
```



```
In [17]: sg1 = librosa.feature.melspectrogram(S=sg_mag, sr=sr)
display(librosa.display.specshow(sg1))

<matplotlib.axes._subplots.AxesSubplot at 0x7fcc5c0f1ac8>
```

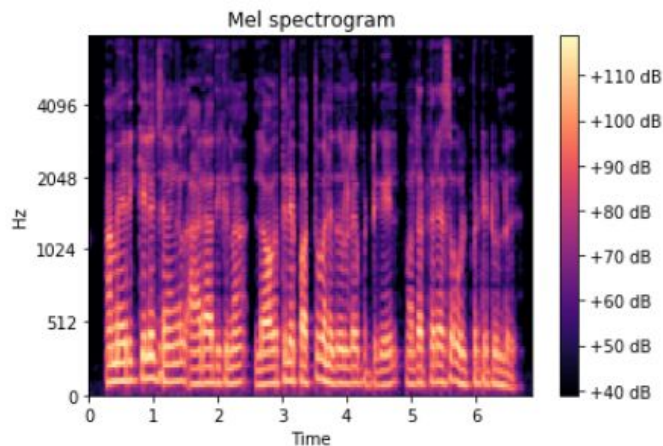


# Exploratory Data Analysis

## Mel Spectrogram vs Amplitude Plot

```
In [20]: # code adapted from the librosa.feature.melspectrogram documentation
librosa.display.specshow(sg2, sr=16000, y_axis='mel', fmax=8000, x_axis='time')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel spectrogram')
```

```
Out[20]: Text(0.5, 1.0, 'Mel spectrogram')
```

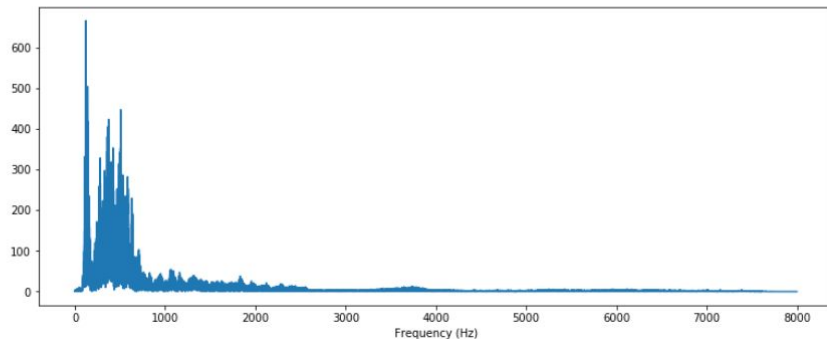


# Exploratory Data Analysis

## Fourier series transform

```
In [28]: # Code adapted from https://musicinformationretrieval.com/fourier_transform.html and the original
# implementation of fastai audio by John Hartquist at https://github.com/sevenfx/fastai_audio/
def fft_and_display(signal, sr):
    ft = scipy.fftpack.fft(signal, n=len(signal))
    ft = ft[:len(signal)//2+1]
    ft_mag = np.absolute(ft)
    f = np.linspace(0, sr/2, len(ft_mag)) # frequency variable
    plt.figure(figsize=(13, 5))
    plt.plot(f, ft_mag) # magnitude spectrum
    plt.xlabel('Frequency (Hz)')
```

```
In [29]: y, sr = librosa.load("data/slr/male/mlm_00269_00156195788.wav", sr=16000)
fft_and_display(y, sr)
```



# Research paper

A study on Text to speech systems for Non-English languages

---



**Thank you!**