Malayalam Text To Speech system

Guide: Jiby J P
Project Coordinator: Manilal D.L

Group 16
Kurian Benoy 34
kurianbenoy@mec.ac.in

Abstract

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

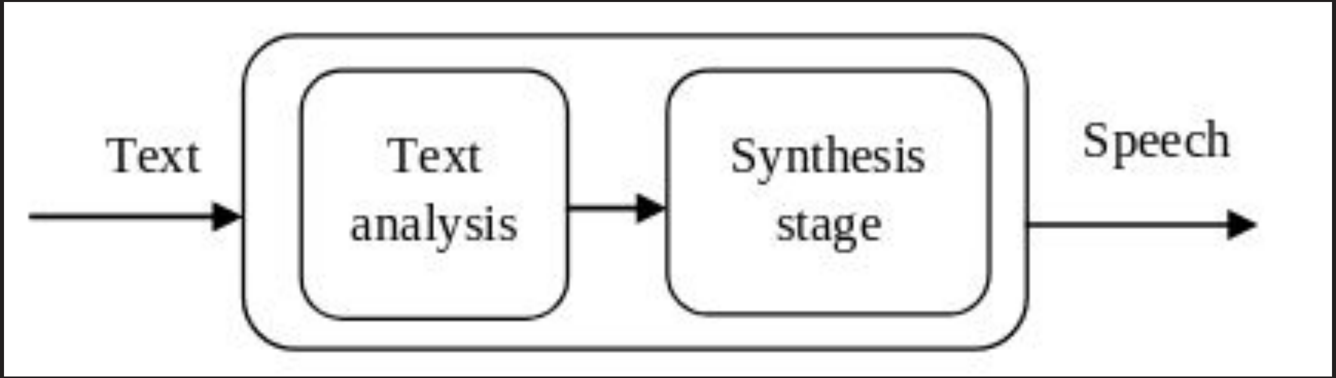
The objective of this project is to attempt to build a Text to speech system in Malayalam using the latest techniques available primarily focusing to reach the state of art performance in speech, and to provide a comprehensive survey on text-to-speech systems for non-English languages which are usually having less resources.

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.


Text
Text analysis
Synthesis stage
Speech

Literature Survey Overview

Research paper	Language	Method	Metric	Disadvantages
Corpus Driven Malayalam Text-to-Speech Synthesis for Interactive Voice Response System	Malayalam	Concatenative synthesis	Mean Opinion Score	MOS score for complex sentences are less
Text Normalization for Bangla, Khmer, Nepali, Javanese, Sinhala and Sudanese TTS	Bangla, Khmer, Nepali, Javanese, Sinhala, Sudanese	NA	No. of sentences correct in tests generated by grammar	Need to add coverage of grammar to have more measurement units in each language and solve test abbreviation issues in each language Not able to evaluate these grammar against some standard unseen text corpora
Orthography with Maratha	Maratha, Telegu	Novel technique specific for this paper only	MCD sore	Detecting noise in ASR transcript and mitigating the effects of that noise in synthesis output. Need to have a larger acoustic model trained on larger phone set
Maithili Text to Speech system	Maithili	Unit Selection Synthesis	Mean Opinion Score (82% accuracy)	Haven't added features like prosody and can be made more robust
Speaking Style Adaption in Text-to-Speech Synthesis using sequence-to-sequence models with attention	Lomard speech(speaking in loud noise)	Wavenet model using seq-to-seq RNN	Mean Opinion Score	Not yet trained Wavenet and Seq2Seq TTS model in a single pipeline.
Indonesian TTS using Diphone based speech TTS	Indonesian	Diphone Concatenative synthesis	Mean Opinion Score	The process of finding an example of the word in phoneme is less precise so result is not perfect In segmentation process diphone still much less precise

Table 2 - Comparative Analysis of Related Works

Hardware Requirements

Server Setup

- Operating system:Debian 10(Buster) / Linux
- RAM:4 GB or more
- Processor:2.3GHz Intel i5 or more
- Memory: 32 GB or greater, HDD
- CUDA supported GPUs(Nvidia 940MX)

Software Requirements

- **Language:** Python
- **Application Framework:** Flask
- **Database:** Sqlite
- **Software packages:** Flite, ESpeak-Ng

Libraries and Packages

Pytorch

PyTorch is an open source machine learning library based on the Torch library,used for applications such as computer vision and natural language processing,[4] primarily developed by Facebook’s AI Research lab (FAIR).
It’s a Python-based scientific computing package targeted at two sets of audiences:
1. A replacement for NumPy to use the power of GPUs
2. a deep learning research platform that provides maximum flexibility and speed

Flite

Flite is an open source small fast run-time text to speech engine. It is the latest addition to the suite of free software synthesis tools including University of Edinburgh’s Festival Speech Synthesis System and Carnegie Mellon University’s FestVox project, tools, scripts and documentation for building synthetic voices. However, flite itself does not require either of these systems to compile and run.

Festival

Festival offers a general framework for building speech synthesis systems as well as including examples of various modules. As a whole it offers full text to speech through a number APIs: from shell level, though a Scheme command interpreter, as a C++ library, and an Emacs interface.Festival is multi-lingual (currently English (US and UK) and Spanish are distributed but a host of other voices have been developed by others) though English is the most advanced.

ESpeak-Ng

The eSpeak NG is a compact open source software text-to-speech synthesizer for Linux, Windows, Android and other operating systems. It supports more than 100 languages and accents. It is based on the eSpeak engine created by Jonathan Duddington. eSpeak NG uses a ”formant synthesis” method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings. It also supports Klatt formant synthesis, and the ability to use MBROLA as backend speech synthesizer.

Flask

Flask (source code) is a Python web framework built with a small core and easy to extend philosophy. Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

Other Python Packages

- Numpy
- Pandas
- Librosa
- Matplotlib
- Scipy

Module Description

1. Exploratory Data Analysis & Dataset collection

Dataset Collected

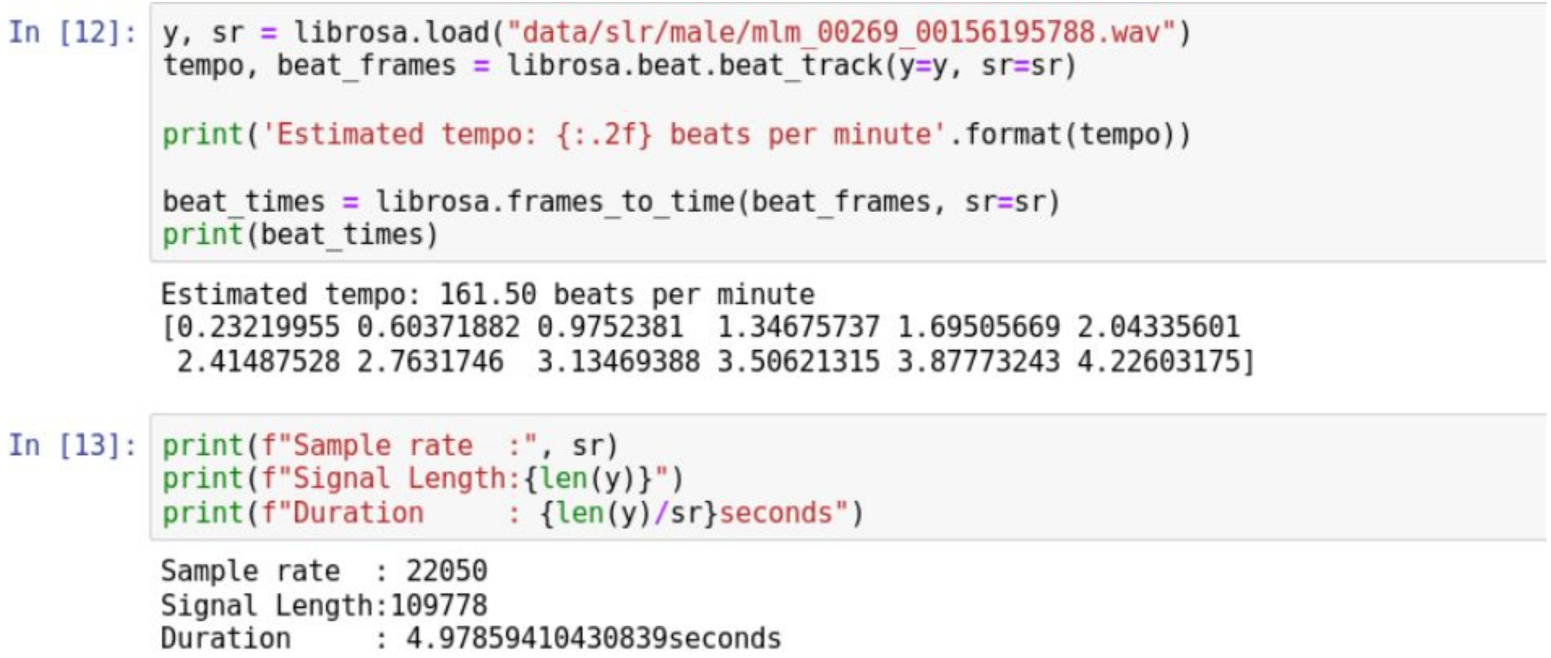
1. Malayalam Speech Corpora, which was initiated to create high quality dataset under SMC.The recording platform can be found online.

2. Crowdsourced high-quality Malayalam multi-speaker speech data set by openslr.org dataset.

3. Another 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.

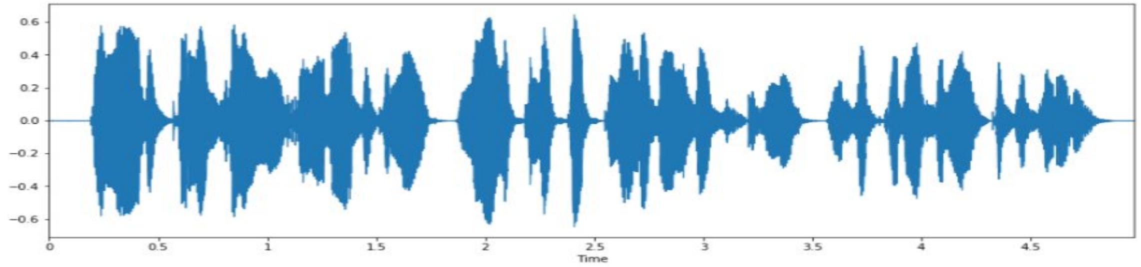
Exploratory Data Analysis

We are exploring the OpenSLR dataset, ie the 2nd dataset collected. OpenSLR dataset has voice recording of about 5 hours in total. We plotted the SampleRate, Signal length, duration of each collected dataset, displayed the waveform, mel-spectogram, and its relation with amplitude plot. MSC dataset was analysed and a PR was being sent to the project site.

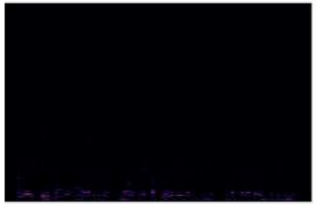


```
In [12]: y, sr = librosa.load("data/slr/male/mlm_00269_00156195788.wav")
         tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr)

         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
```

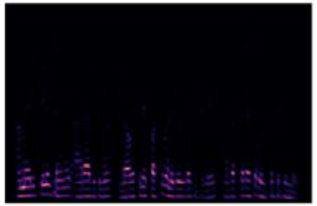
```
In [15]: import librosa.display
         plt.figure(figsize=(15, 5))
         librosa.display.waveplot(y, sr=sr)

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



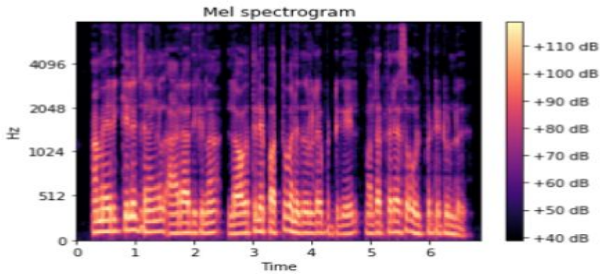
```
In [16]: sg0 = librosa.stft(y)
         sg_mag, sg_phase = librosa.magphase(sg0)
         display(librosa.display.specshow(sg_mag))

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



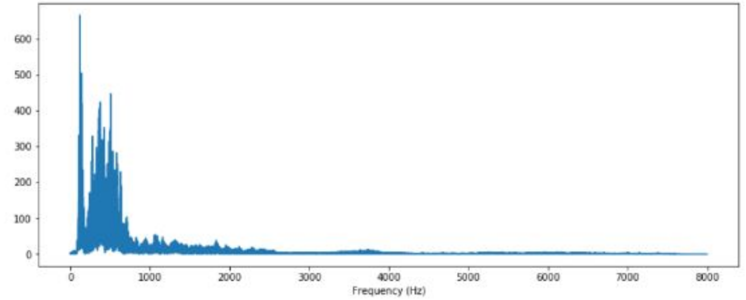
```
In [17]: sg1 = librosa.feature.melspectrogram(S=sg_mag, sr=sr)
         display(librosa.display.specshow(sg1))

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



```
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')
```



```
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)
```



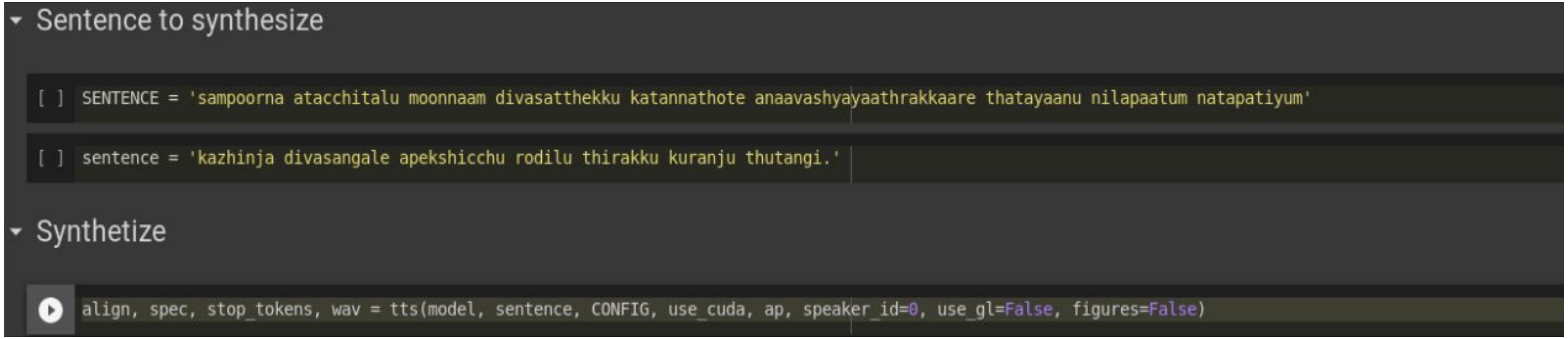
2. Train first Text to speech s/m

Tactron architecture

Tactron model is a sequence to sequence model, which is Currently one of the best Text to speech architectures. An improved architecture called Tactron2 also exists. Seq-2-seq models use RNNs as backbone. They use a sequence-to-sequence model optimized for TTS to map a sequence of letters to a sequence of features that encode the audio. These features, an 80-dimensional audio spectrogram with frames computed every 12.5 milliseconds, capture not only pronunciation of words, but also various subtleties of human speech, including volume, speed and intonation. Finally these features are converted to a 24 kHz waveform using a WaveNet-like architecture.

Transliteration of Malayalam & pass to Tactron-2 based TTS

We transliterated Malayalam text and obtained english like representation and passed it to English TTS and obtained sound samples. The transliteration to Malayalam to English TTS was done using LibIndic library. First voice sample was generated by this method.



Sentence to synthesize

```
[ ] SENTENCE = 'sampoorna atacchitalu moonnaam divasatthekku katannathote anaavashyayaathrakkaare thatayaanu nilapaatum natapatiyum'
```

```
[ ] sentence = 'kazhinja divasangale apekshicchu rodilu thirakku kuranju thutangi.'
```

Synthetize

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

3. Fine tune TTS system

Tactron2 architecture

Generating very natural sounding speech from text (text-to-speech, TTS) has been a research goal for decades. There has been great progress in TTS research over the last few years and many individual pieces of a complete TTS system have greatly improved. Incorporating ideas from past work such as Tacotron and WaveNet, and these improvements ended up with our new system, Tacotron 2.

In a nutshell, Tactron is used sequence to sequence models optimized for TTS to map a sequence of letters to a sequence of features that encode the audio. These features, are an 80-dimensional audio spectrogram with frames computed every 12.5 milliseconds, capture the complex features. These features are finally forming a waveform using Wavenet like architecture.

ESpeak-Ng architecture for TTS

The eSpeak NG is a compact open source software text-to-speech synthesizer for Linux, Windows, Android and other operating systems. It supports 107 languages and accents. It is based on the eSpeak engine created by Jonathan Duddington. eSpeak NG uses a ”formant synthesis” method.

This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings. It also supports Klatt formant synthesis, and the ability to use MBROLA voices. The technique eSpeak NG uses is quite old, even though the software support text to speech synthesis for Malayalam, the final output sound is understandable for even a simple sentences.

Flite architecture for TTS

Flite is an open source small fast run-time text to speech engine. It is the latest addition to the suite of free software synthesis tools including University of Edinburgh's Festival Speech Synthesis System and Carnegie Mellon University's FestVox project, tools, scripts and documentation for building synthetic voices. However, flite itself does not require either of these systems to compile and run. Flite by default doesn't support languages like Malayalam. Instead we used a Indic voice pre- trained with Tamil, a dravidian language and found that the results were quite good enough and quite audible, compared to existing approaches.

4. User Interface

A web interface allows user to insert the input text using a Malayalam input tools like Swanlekha and enter the text input into the database, and run the TTS engine using the backend.

GUI

1. Webpage containing basic description of project
2. A button for Image Upload

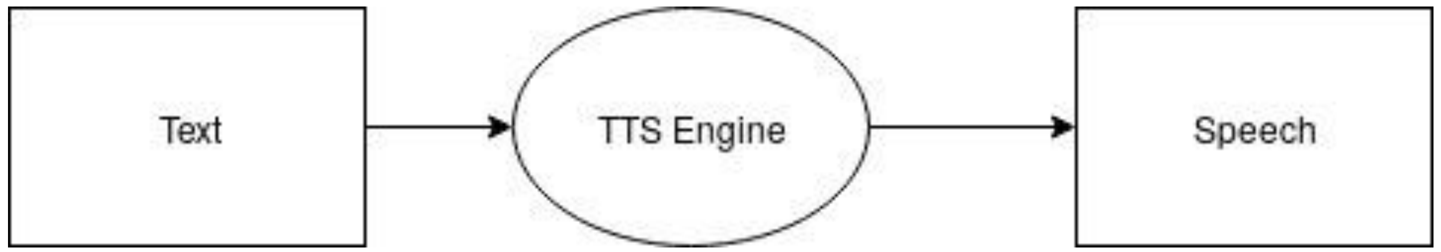
4. User Interface

Backend
Flask is python framework used for building web back-ends.Flask is used in this project to build
the back-end.The application runs the TTS engine inside flask.

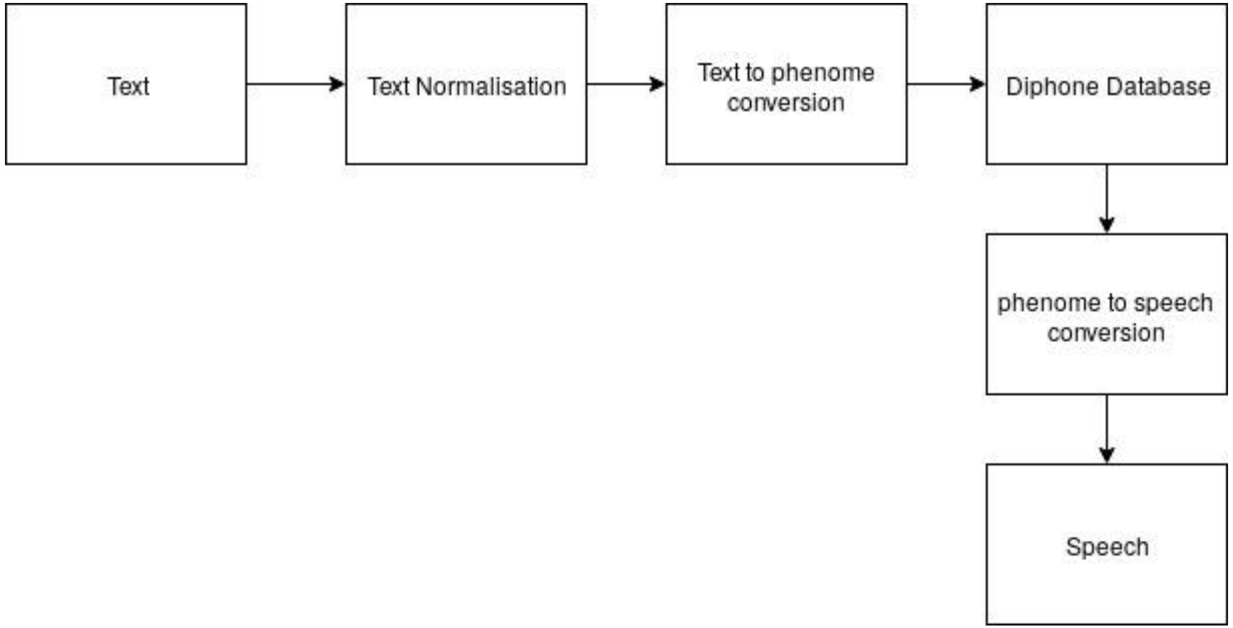
Flite - TTS Engine
The actual text to speech conversion is being done by Flite speeech engine using Diphone speech
synthesis challenge.

Data Flow diagram

Level 0 DFD



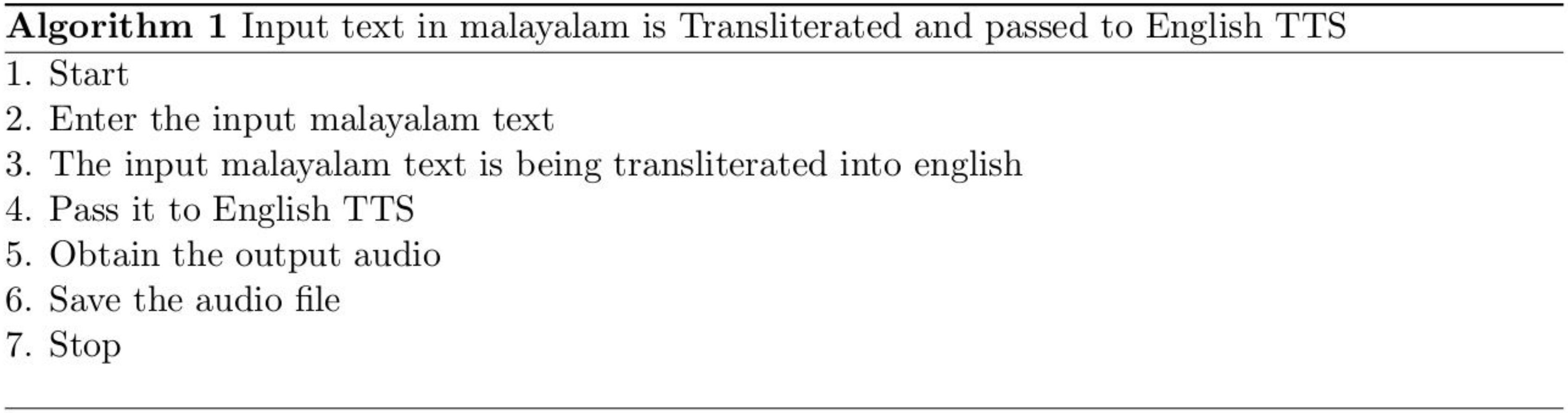
Level 1 DFD


Text
Text Normalisation
Text to phenome conversion
Diphone Database
phenome to speech conversion
Speech

Implementation

Algorithms

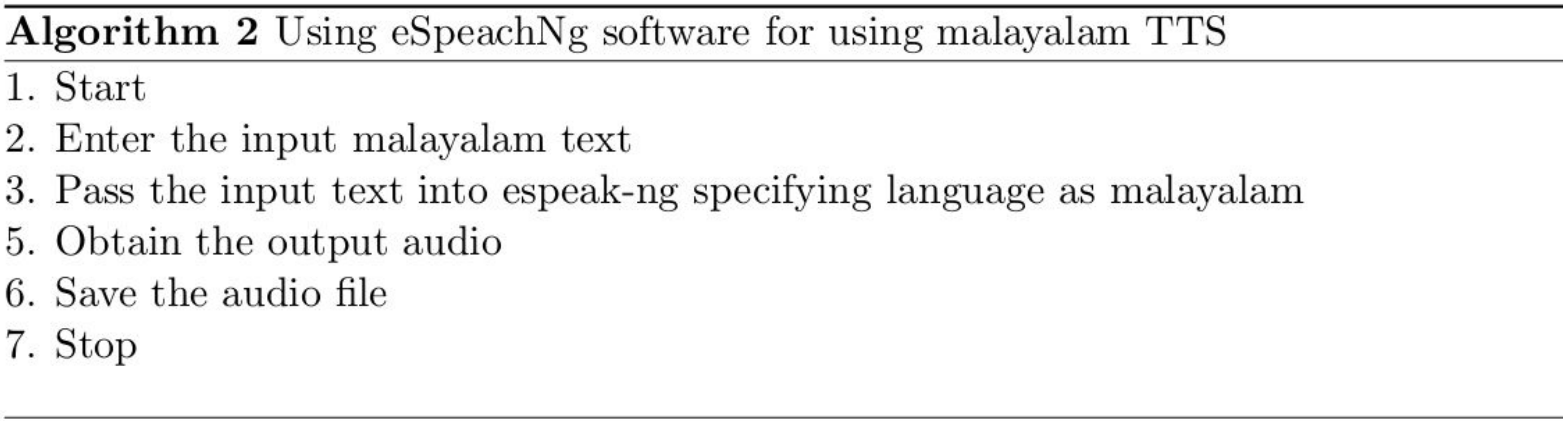
Approach1



Algorithm 1 Input text in malayalam is Transliterated and passed to English TTS

```
1. Start
2. Enter the input malayalam text
3. The input malayalam text is being transliterated into english
4. Pass it to English TTS
5. Obtain the output audio
6. Save the audio file
7. Stop
```

Approach2



Algorithm 2 Using eSpeachNg software for using malayalam TTS

```
1. Start
2. Enter the input malayalam text
3. Pass the input text into espeak-ng specifying language as malayalam
5. Obtain the output audio
6. Save the audio file
7. Stop
```

Approach 3

Algorithm 3 Using Flite software for using malayalam TTS

```
1. Start
2. Enter the input malayalam text
3. Pass the input text into flite specifying language as Kannada
5. Obtain the output audio
6. Save the audio file
7. Stop
```

Text Normalisation

1. Take the input word text
2. Identify tokens in the text like white space as sepearator and punctuation to separate from raw
audio tokens
3. Identify ambiguity in abbreviation, end-of-utterance.
4. Chunk Tokens. Use a Decision tree for this purpose
5. Identify type of tokens
6. Convert tokens to words, by expanding the necessary sequence.

Speech synthesiser in Tactron

Require: To generate a target mel-spectogram sequence in parallel
1. Take the input Phoneme, which is smallest unit of particular sound
2. Pass it on top of a Phoneme embedding
3. Take this phoneme embedding and pass it on top of feed forward transfromer networks
4. Predict time duration with an duration extractor
5. Continue training this untill the MSE Loss is less for the paper
6. Choose the model with best MOS performance

Development Tools

- Git
- Visual Studio Code
- Jupyter Notebooks
- Kaggle Notebooks
- Google Colab
- Postman
- Github

Testing

Each testing methodology has a defined test objective, test strategy and deliverables. We performed the following testing: Unit testing, Integration testing, System testing.

Unit Testing

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. We considered the following units for this process:

Unit Testing

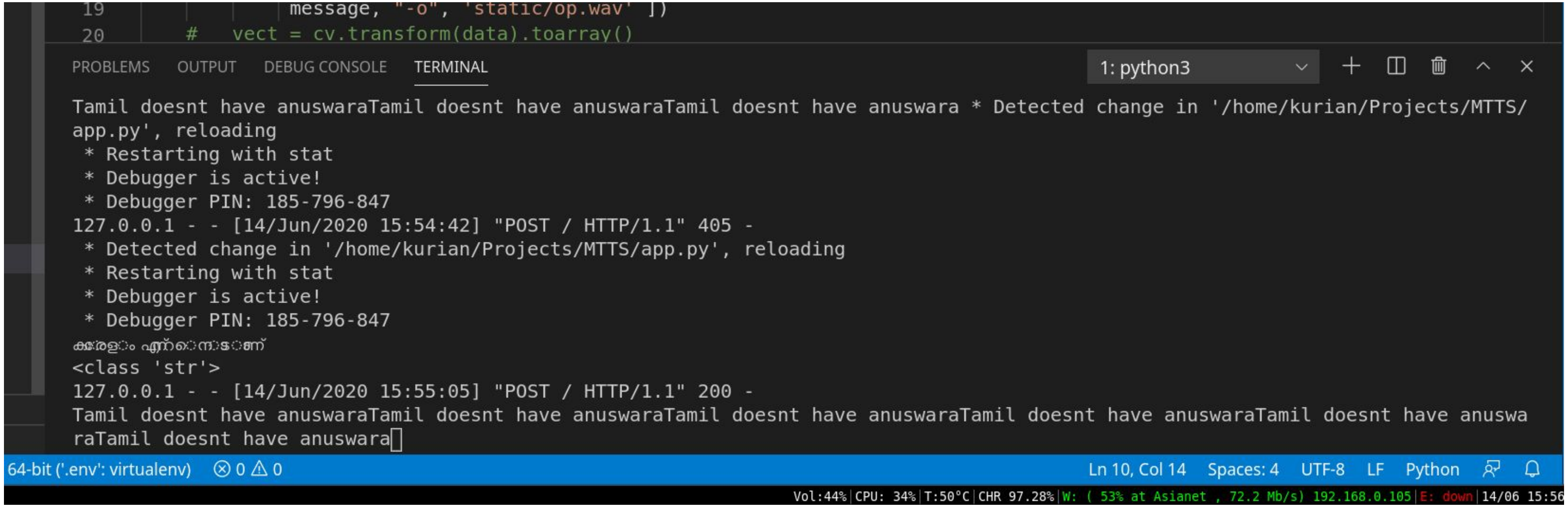
- **Website:** The individual pages of the website was taken and was tested separately to see whether the individual outputs are available, and whether data flow properly takes place.

- **TTS Engine:** Various malayalam words were being passed into TTS engine and checked if the proper output was obtained or not.

Integration Testing

Integration testing refers to the testing of the different modules/components that have been suc-
cessfully unit tested when integrated together to perform specific tasks and activities. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together.

Integration Testing



```
19            message, "-o", 'static/op.wav' ])
20        #   vect = cv.transform(data).toarray()
```

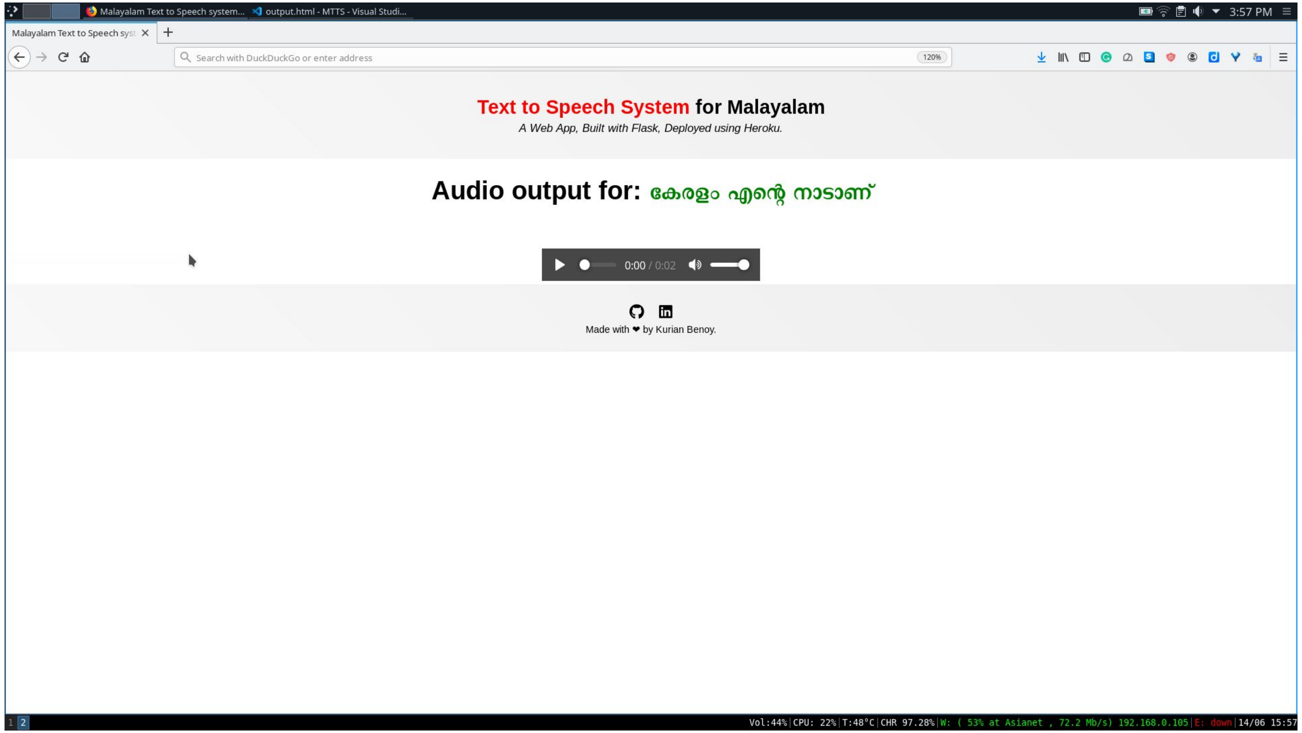
```
Tamil doesnt have anuswaraTamil doesnt have anuswaraTamil doesnt have anuswara * Detected change in '/home/kurian/Projects/MTTS/
app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 185-796-847
127.0.0.1 - - [14/Jun/2020 15:54:42] "POST / HTTP/1.1" 405 -
 * Detected change in '/home/kurian/Projects/MTTS/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 185-796-847
കേരളം എന്റെ നാടാണ്
<class 'str'>
127.0.0.1 - - [14/Jun/2020 15:55:05] "POST / HTTP/1.1" 200 -
Tamil doesnt have anuswaraTamil doesnt have anuswaraTamil doesnt have anuswaraTamil doesnt have anuswaraTamil doesnt have anuswa
raTamil doesnt have anuswara
```

```
64-bit ('.env': virtualenv)   ⊗ 0 ⚠ 0                     Ln 10, Col 14   Spaces: 4   UTF-8   LF   Python
Vol:44%|CPU: 34%|T:50°C|CHR 97.28%|W: ( 53% at Asianet , 72.2 Mb/s) 192.168.0.105|E: down|14/06 15:56
```

System Testing

System testing is testing conducted on a complete integrated system to evaluate the system’s compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together. All modules were integrated at the end of integration testing and the entire system was tested.

System Testing



Text to Speech System for Malayalam

A Web App, Built with Flask, Deployed using Heroku.

Audio output for: കേരളം എന്റെ നാടാണ്

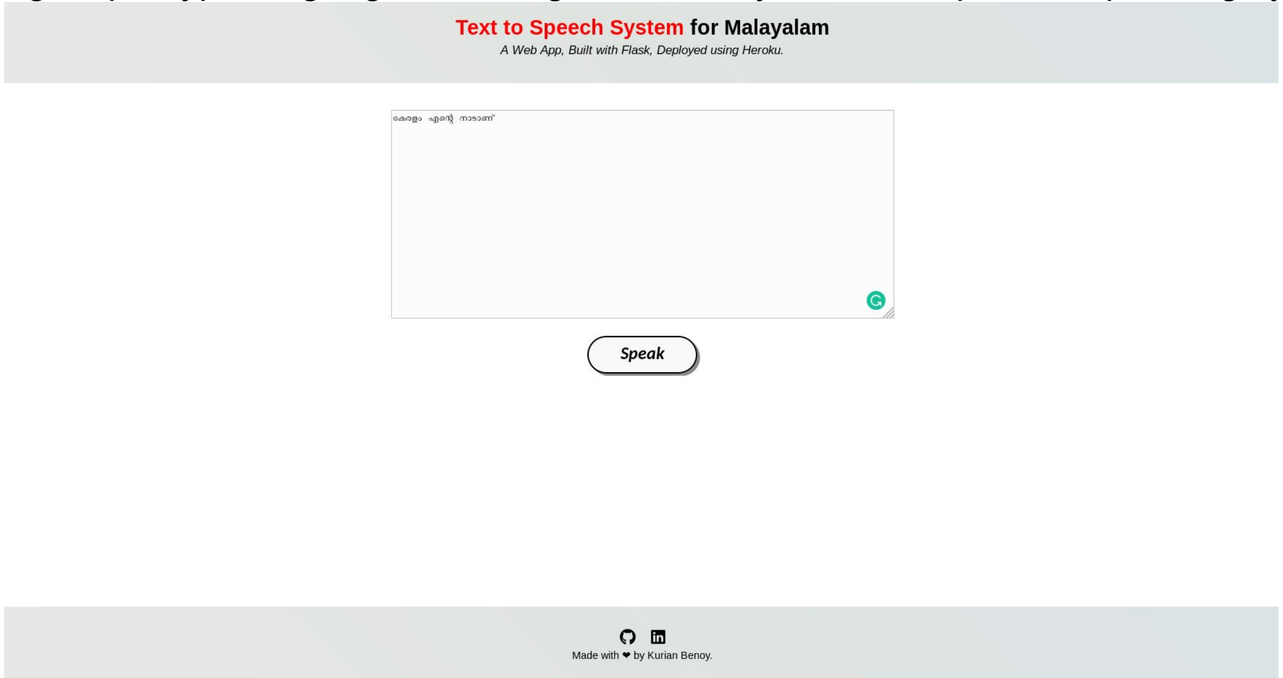
Made with ❤ by Kurian Benoy.

Malayalam Text to Speech system...
output.html - MTTS - Visual Studi...
3:57 PM
Search with DuckDuckGo or enter address
120%
0:00 / 0:02
Vol:44% | CPU: 22% | T:48°C | CHR 97.28% | W: (53% at Asianet , 72.2 Mb/s) 192.168.0.105 | E: down | 14/06 15:57

GUI Components

Home Page

The user can input the malayalam text to be synthesised using Input tool methods like Swanlekha, or through Malayalam alphabet keyboards in various fonts in Malayalam like Gayathri, Rachana, Mancheri, and other fonts. I would highly recommend users to download fonts from smc website, and change input type language from English to Malayalam in respective operating system.


Text to Speech System for Malayalam

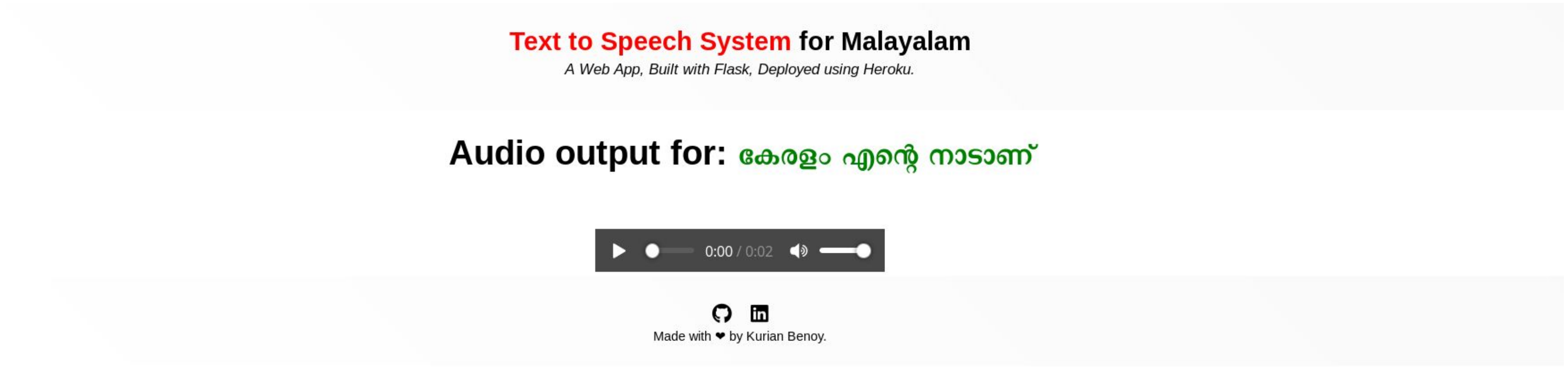
A Web App, Built with Flask, Deployed using Heroku.

കേരളം എന്റെ നാടാണ്

Made with ❤ by Kurian Benoy.

Result page

The output after the TTS input being passed through the front end is shown in a output page as shown in the image.



Text to Speech System for Malayalam

A Web App, Built with Flask, Deployed using Heroku.

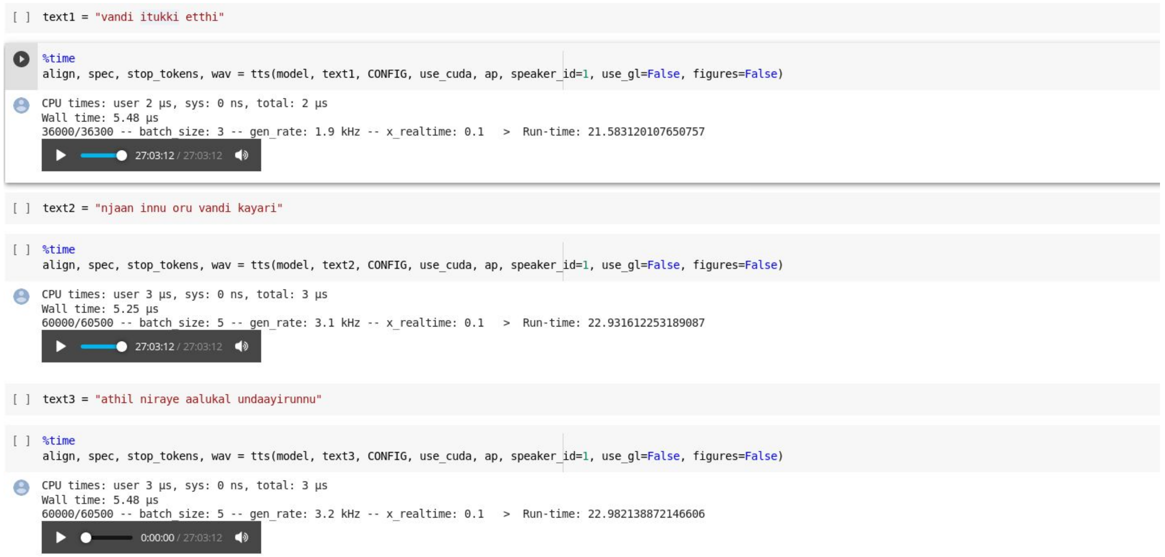
Audio output for: കേരളം എന്റെ നാടാണ്

Made with ❤ by Kurian Benoy.

Results

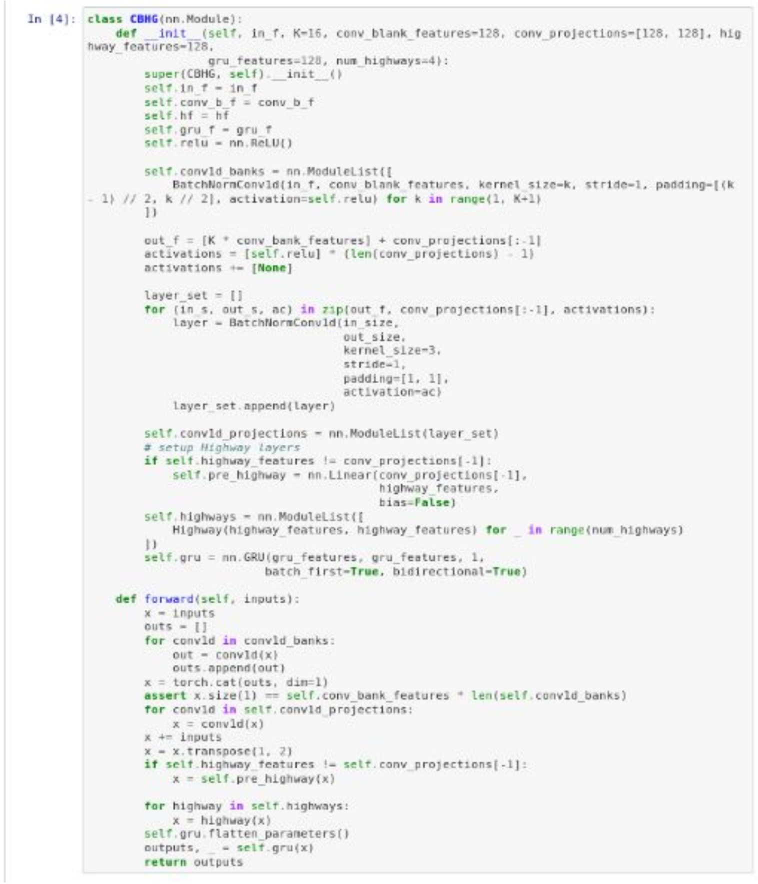
Transliterated malayalam

Here the malayalam input text is being passed into a library to transliterate malayalam into english. We used LibIndic library developed by Indic project for this purpose. The english text is then passed into a text to speech system trained using Tactron-2 architecture and then we generate input audio text.



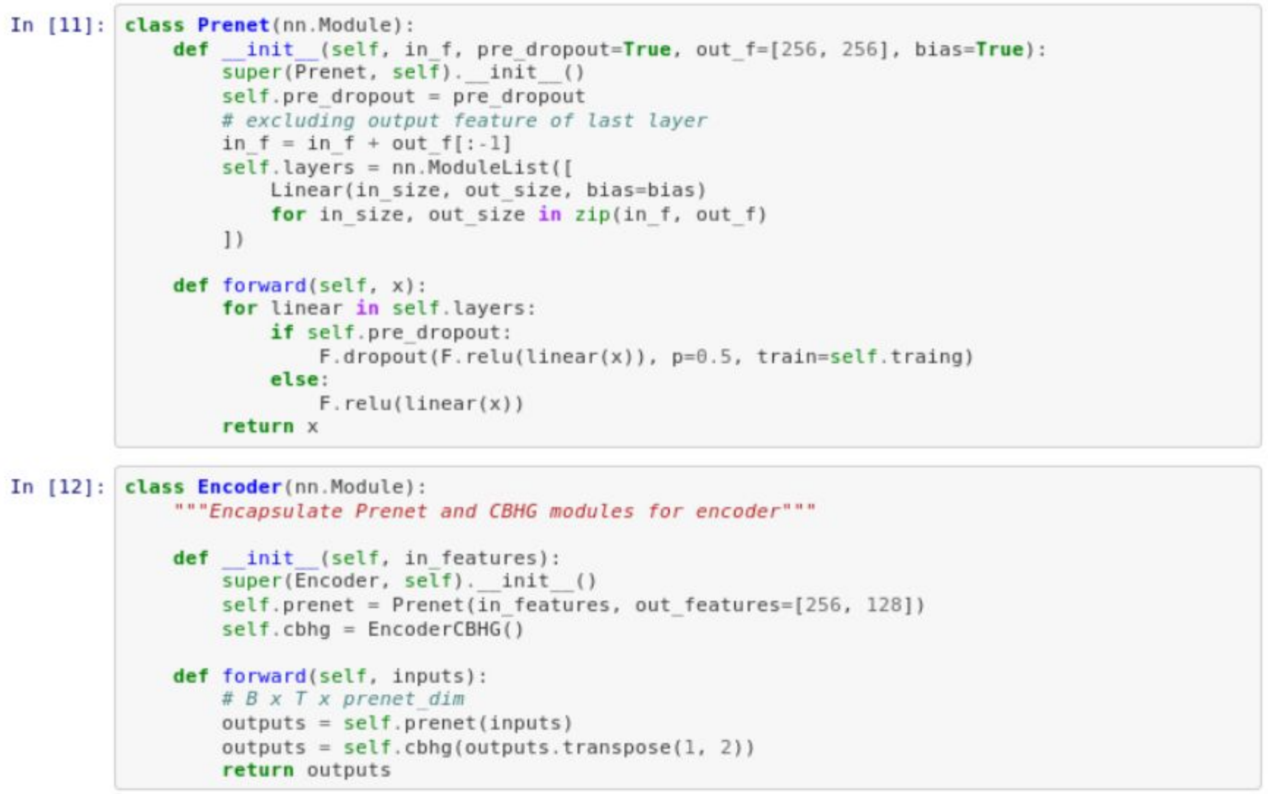
```
[ ] text1 = "vandi itukki etthi"

%time
align, spec, stop_tokens, wav = tts(model, text1, CONFIG, use_cuda, ap, speaker_id=1, use_gl=False, figures=False)

CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.48 µs
36000/36300 -- batch_size: 3 -- gen_rate: 1.9 kHz -- x_realtime: 0.1   > Run-time: 21.583120107650757
27:03:12 / 27:03:12

[ ] text2 = "njaan innu oru vandi kayari"

[ ] %time
align, spec, stop_tokens, wav = tts(model, text2, CONFIG, use_cuda, ap, speaker_id=1, use_gl=False, figures=False)

CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.25 µs
60000/60500 -- batch_size: 5 -- gen_rate: 3.1 kHz -- x_realtime: 0.1   > Run-time: 22.931612253189087
27:03:12 / 27:03:12

[ ] text3 = "athil niraye aalukal undaayirunnu"

[ ] %time
align, spec, stop_tokens, wav = tts(model, text3, CONFIG, use_cuda, ap, speaker_id=1, use_gl=False, figures=False)

CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.48 µs
60000/60500 -- batch_size: 5 -- gen_rate: 3.2 kHz -- x_realtime: 0.1   > Run-time: 22.982138872146606
0:00:00 / 27:03:12
```

Tactron architecture



```
In [4]: class CBHG(nn.Module):
            def __init__(self, in_f, K=16, conv_blank_features=128, conv_projections=[128, 128], hig
        hway_features=128,
                         gru_features=128, num_highways=4):
                super(CBHG, self).__init__()
                self.in_f = in_f
                self.conv_b_f = conv_b_f
                self.hf = hf
                self.gru_f = gru_f
                self.relu = nn.ReLU()

                self.conv1d_banks = nn.ModuleList([
                    BatchNormConv1d(in_f, conv_blank_features, kernel_size=k, stride=1, padding=[(k
        - 1) // 2, k // 2], activation=self.relu) for k in range(1, K+1)
                ])

                out_f = [K * conv_bank_features] + conv_projections[:-1]
                activations = [self.relu] * (len(conv_projections) - 1)
                activations += [None]

                layer_set = []
                for (in_s, out_s, ac) in zip(out_f, conv_projections[:-1], activations):
                    layer = BatchNormConv1d(in_size,
                                            out_size,
                                            kernel_size=3,
                                            stride=1,
                                            padding=[1, 1],
                                            activation=ac)
                    layer_set.append(layer)

                self.conv1d_projections = nn.ModuleList(layer_set)
                # setup Highway layers
                if self.highway_features != conv_projections[-1]:
                    self.pre_highway = nn.Linear(conv_projections[-1],
                                                 highway_features,
                                                 bias=False)
                self.highways = nn.ModuleList([
                    Highway(highway_features, highway_features) for _ in range(num_highways)
                ])
                self.gru = nn.GRU(gru_features, gru_features, 1,
                                  batch_first=True, bidirectional=True)

            def forward(self, inputs):
                x = inputs
                outs = []
                for conv1d in conv1d_banks:
                    out = conv1d(x)
                    outs.append(out)
                x = torch.cat(outs, dim=1)
                assert x.size(1) == self.conv_bank_features * len(self.conv1d_banks)
                for conv1d in self.conv1d_projections:
                    x = conv1d(x)
                x += inputs
                x = x.transpose(1, 2)
                if self.highway_features != self.conv_projections[-1]:
                    x = self.pre_highway(x)

                for highway in self.highways:
                    x = highway(x)
                self.gru.flatten_parameters()
                outputs, _ = self.gru(x)
                return outputs
```

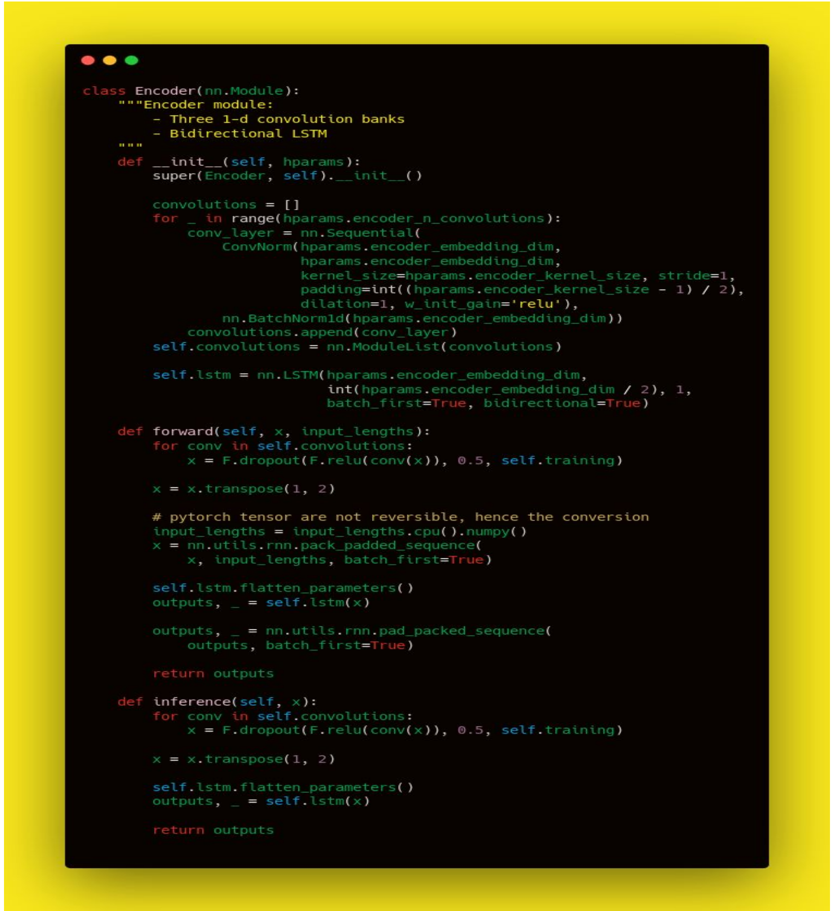
Tactron architecture



```
In [11]: class Prenet(nn.Module):
             def __init__(self, in_f, pre_dropout=True, out_f=[256, 256], bias=True):
                 super(Prenet, self).__init__()
                 self.pre_dropout = pre_dropout
                 # excluding output feature of last layer
                 in_f = in_f + out_f[:-1]
                 self.layers = nn.ModuleList([
                     Linear(in_size, out_size, bias=bias)
                     for in_size, out_size in zip(in_f, out_f)
                 ])

             def forward(self, x):
                 for linear in self.layers:
                     if self.pre_dropout:
                         F.dropout(F.relu(linear(x)), p=0.5, train=self.traing)
                     else:
                         F.relu(linear(x))
                 return x
```

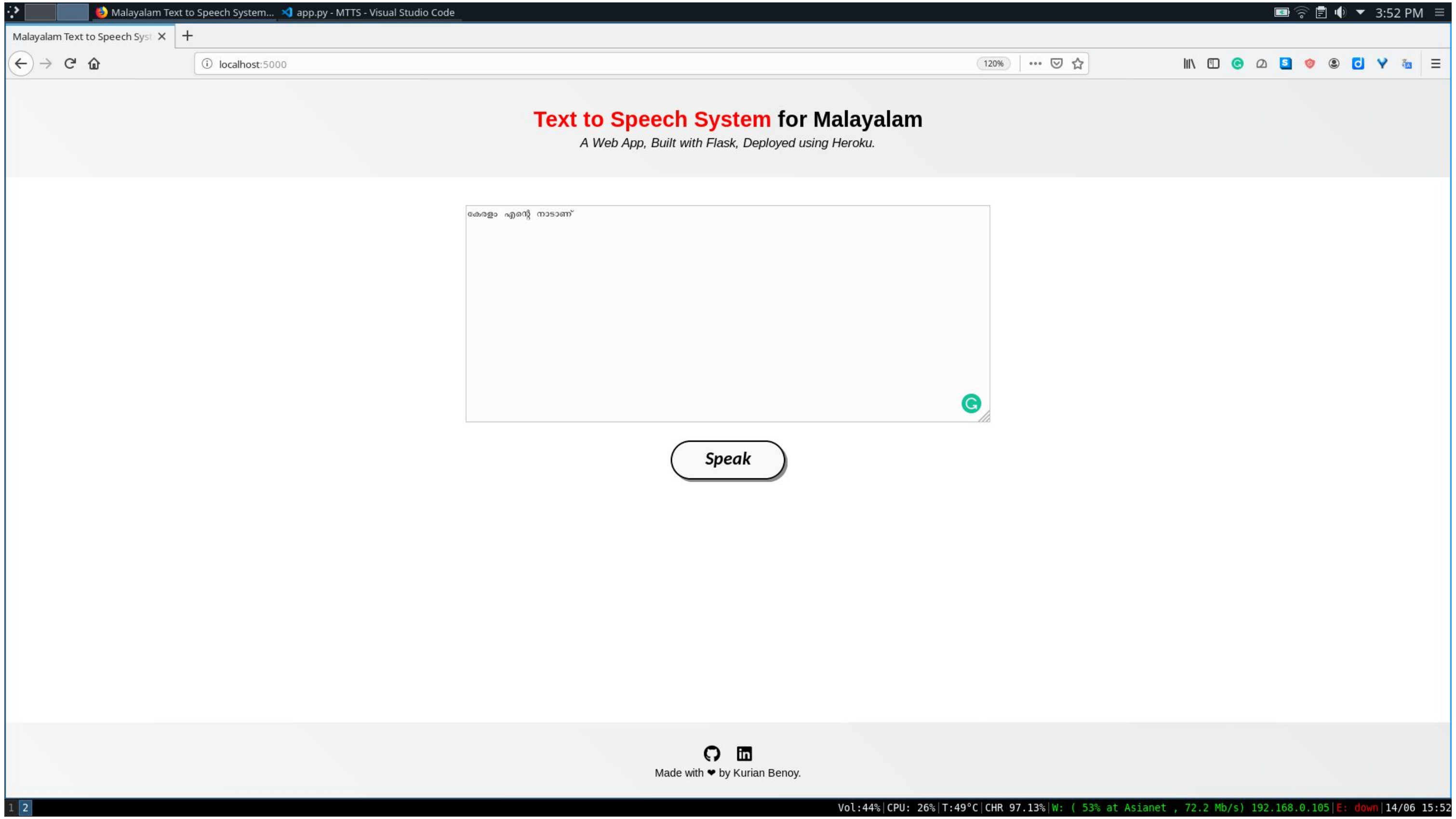
```
In [12]: class Encoder(nn.Module):
             """Encapsulate Prenet and CBHG modules for encoder"""

             def __init__(self, in_features):
                 super(Encoder, self).__init__()
                 self.prenet = Prenet(in_features, out_features=[256, 128])
                 self.cbhg = EncoderCBHG()

             def forward(self, inputs):
                 # B x T x prenet_dim
                 outputs = self.prenet(inputs)
                 outputs = self.cbhg(outputs.transpose(1, 2))
                 return outputs
```

Tactron2 architecture



```python
class Encoder(nn.Module):
    """Encoder module:
        - Three 1-d convolution banks
        - Bidirectional LSTM
    """
    def __init__(self, hparams):
        super(Encoder, self).__init__()

        convolutions = []
        for _ in range(hparams.encoder_n_convolutions):
            conv_layer = nn.Sequential(
                ConvNorm(hparams.encoder_embedding_dim,
                         hparams.encoder_embedding_dim,
                         kernel_size=hparams.encoder_kernel_size, stride=1,
                         padding=int((hparams.encoder_kernel_size - 1) / 2),
                         dilation=1, w_init_gain='relu'),
                nn.BatchNorm1d(hparams.encoder_embedding_dim))
            convolutions.append(conv_layer)
        self.convolutions = nn.ModuleList(convolutions)

        self.lstm = nn.LSTM(hparams.encoder_embedding_dim,
                            int(hparams.encoder_embedding_dim / 2), 1,
                            batch_first=True, bidirectional=True)

    def forward(self, x, input_lengths):
        for conv in self.convolutions:
            x = F.dropout(F.relu(conv(x)), 0.5, self.training)

        x = x.transpose(1, 2)

        # pytorch tensor are not reversible, hence the conversion
        input_lengths = input_lengths.cpu().numpy()
        x = nn.utils.rnn.pack_padded_sequence(
            x, input_lengths, batch_first=True)

        self.lstm.flatten_parameters()
        outputs, _ = self.lstm(x)

        outputs, _ = nn.utils.rnn.pad_packed_sequence(
            outputs, batch_first=True)

        return outputs

    def inference(self, x):
        for conv in self.convolutions:
            x = F.dropout(F.relu(conv(x)), 0.5, self.training)

        x = x.transpose(1, 2)

        self.lstm.flatten_parameters()
        outputs, _ = self.lstm(x)

        return outputs
```

User Interface



Malayalam Text to Speech System... app.py - MTTS - Visual Studio Code 3:52 PM

Malayalam Text to Speech Syst

localhost:5000 120%

Text to Speech System for Malayalam

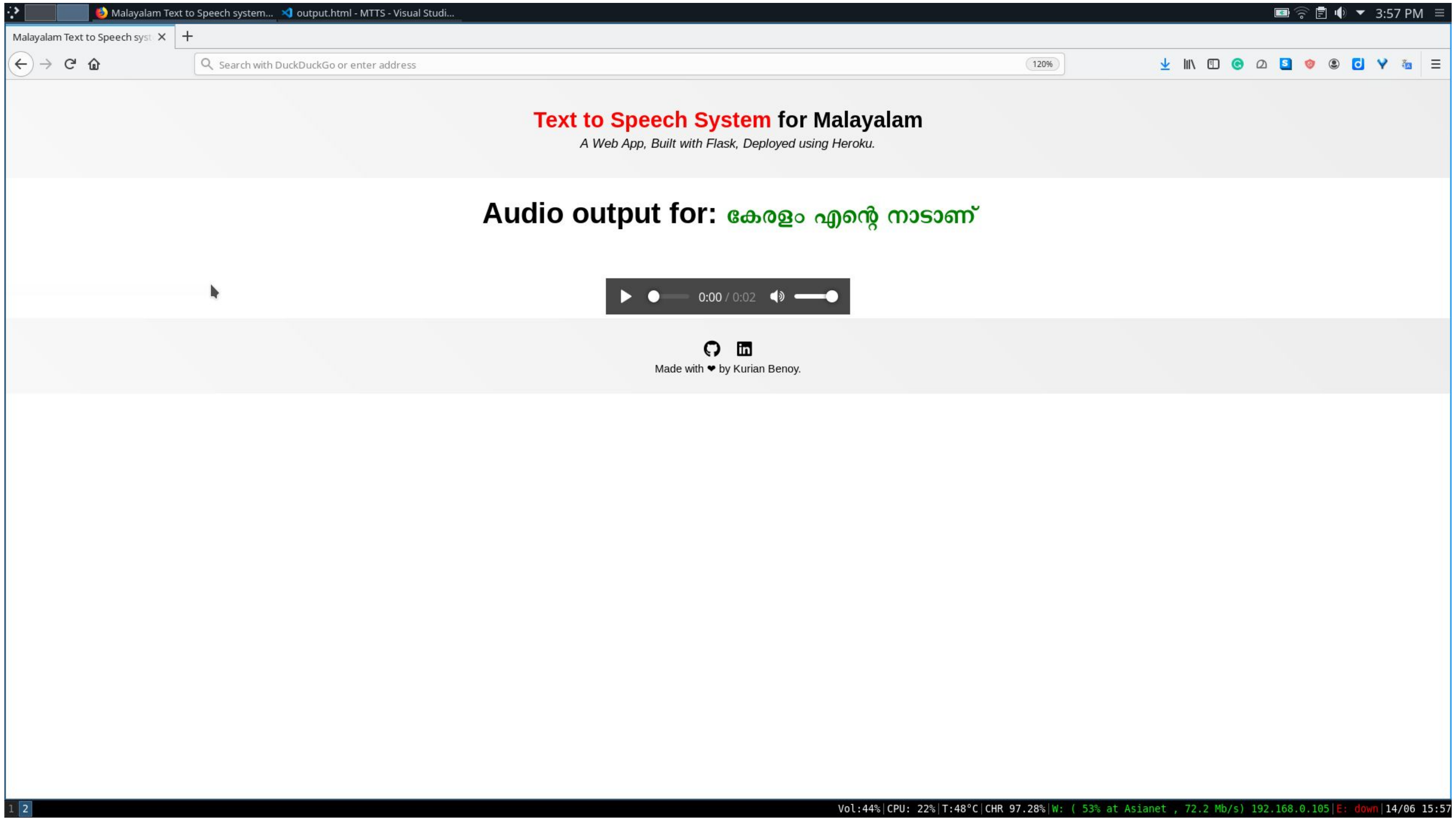
A Web App, Built with Flask, Deployed using Heroku.

കേരളം എന്റെ നാടാണ്

Made with ❤ by Kurian Benoy.

```
Vol:44%|CPU: 26%|T:49°C|CHR 97.13%|W: ( 53% at Asianet , 72.2 Mb/s) 192.168.0.105|E: down|14/06 15:52
```

User Interface


Malayalam Text to Speech system...
output.html - MTTS - Visual Studi...
3:57 PM
Malayalam Text to Speech syst
Search with DuckDuckGo or enter address
120%

Text to Speech System for Malayalam

A Web App, Built with Flask, Deployed using Heroku.

Audio output for: കേരളം എന്റെ നാടാണ്

Made with ❤ by Kurian Benoy.

```
Vol:44%|CPU: 22%|T:48°C|CHR 97.28%|W: ( 53% at Asianet , 72.2 Mb/s) 192.168.0.105|E: down|14/06 15:57
```

Mean Opinion Score

Method	MOS score
Transliterate Malayalam & pass to TTS	1.5
Espeak-ng architecture approach	2.5
Dhavani	3.0
Flite architecture	4.0

Conclusion

The scope of the project merely did not involve merely making a text to speech system in Malayalam. In scope of the project there were things like doing exploratory analysis of malayalam speech corpus, a open source contribution, to learning and doing a survey of various techniques of text to speech synthesis and finally making a text to speech system in Malayalam.

The text to speech produced gives better sound quality than samples of the old and deprecated Dhavani project build using Rule based concatenative synthesis, on a comparative study.

Future Scope

The Flite architecture which gave the best result was trained on Tamil. If results of a voice architecture trained for a dravidian language like Tamil is good, we are expecting even better results for a Flite architecture trained with Malayalam voices. In order to train such a model first we need to generate festvox files in malayalam, which is a bit long process.

To solve for neural network architectures like Tactron2 for Malayalam, we first need to understand the semantic representation of language with a Morphological analyzer, and train a festival library module in Malayalam for architectures like Tactron2 to work. Most of the recent developments in text to speech arena, are build on top of this Tactron architecture as a building block.

Publications

International Journal of Research and Analytical Reviews (IJRAR) is a Leading high quality open access and peer reviewed quarterly published research journal. IJRAR is providing a platform for the researchers, academicians, professional, practitioners and students to impart and share knowledge in the form of high quality empirical and theoretical research papers, case studies, literature reviews and book reviews. The aim of the journal is to provide platform for diversity of intellectual pursuit from all corners of the society for enrichment and enhancement of the group readers.

It’s a **UGC approved journal with impact factor of 5.5.**

Title: A Study of Text to Speech systems for Non-English Languages
Online: Paper is available in this link
Authors: Jiby J Puthiyidam, Kurian Benoy
Pages: 7

Thank you!