Malayalam Text to Speech System

CS492 Main-Project

CSU161 34 Kurian Benoy
B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

May 2020

Model Engineering College Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***Malayalam Text to Speech System*** is a bonafide record of the work done by

CSU161 34 Kurian Benoy

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS492 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **APJ Abdul Kalam University**.

Guide

Jiby J Puthiyidam
Assistant Professor
Computer Engineering

Coordinator

Manilal D L
Head of the Department
Computer Engineering

Head of the Department

Manilal D L
Associate Professor
Computer Engineering

June 14, 2020

Acknowledgements

First of all, I would like to thank God the Almighty for giving me the strength, the ability, and the opportunity to carry out this project work.

I would like to express my deep gratitude and heartfelt thanks to my Project guide, Jiby J Puthiyidam, Assistant Professor for his guidance, invaluable dedication, patience and encouragement all throughout the project.

I am highly indebted to our Project Coordinator and Head of the Department, Manilal D L, Associate Professor for his guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

I am thankful to various members in Swathanthra Malayalam Computing for their gracious support and help in clearing doubts related to Malayalam computing and providing direction to the project, prominently Mr Santhosh Thottingal and Mrs Kavya Manohar.

I would like to thank our esteemed Principal, Prof. (Dr.) Vinu Thomas, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

This acknowledgment would be incomplete without thanking the biggest source of my energy that is my family. I am extremely grateful to my parents for giving me love, prayers and confidence throughout this endeavour.

Kurian Benoy

Abstract

Keywords: Text to Speech, Malayalam, speech synthesis, Neural networks, Statistical methods

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. As for our native language like Malayalam there doesn't exist entities like a big entity like Google, Microsoft etc to make software. So there are local communities like SMC, and few people like you, me to solve this problem.

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.

Contents

List of Figures

Chapter 1

Introduction

1.1 What is Text-to-Speech system?

A text-to-speech (TTS) system [1] converts normal language text into speech. The ultimate goal is to create a natural sound from normal text. The current trend in TTS research calls for systems that enable production of speech in different styles with different speaker characteristics and even emotions. Speech synthesis generally refers to the artificial generation of human voice - either in the form of speech or in other forms like songs etc. The computer system used for speech synthesis is known as speech synthesizer.

A text-to-speech-system is composed of two parts:

- The front-end consists of converting a text into graphemes after text normalization, pre-processing, or tokenisation
- The back-end, referred to as the synthesizer, which converts the symbolic linguistic representation into sound. We can add pitch, prosody etc. to target speech as required.

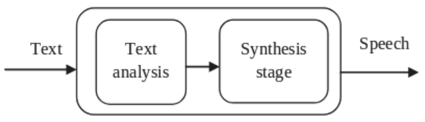


Figure 1.1: Phases of Text to speech system

First, we mention some of the historical developments of Speech synthesis which paved the way for development of contemporary text- to-speech systems. The following contents are extracted from [1] which did a comprehensive historical study of how present system's work. 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 for 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 of producing 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 affected 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.

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

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 HÃ¶gskolan (KTH), Swede. (as mentioned in [1])

For a good TTS it's necessary to have text normalization. Text normalization is the process of converting non-standard words (NSWs) such as numbers, and abbreviations into standard words so that their pronunciations can be derived by a typical means (usually lexicon lookups). Text normalization is, thus, an important component of any text-to-speech (TTS) system. Without text normalization, the resulting voice may sound unintelligent.

1.1.1 Problem Statement

The objective of this project is to attempt to build a Text to speech system in Malayalam using the latest techniques available using ML, and to provide a comprehensive survey on text-to-speech systems for non-English languages which are usually having less resources. As data is a important component of this project, I have helped in contributing to Malayalam Speech Corpus(MSC) and have analysed this dataset.

1.1.2 Proposed Solution

Our solution is to build a text to speech system in Malayalam using the latest techniques of methods and to provide a comprehensive survey on text-to-speech systems for non-English languages which are usually having less resources.

Chapter 2

System Study Report

2.1 Methods of Text to Speech

The various methods for doing Text to speech is mentioned here:

2.1.1 Concatenative Synthesis

Synthesis of unit selection allows use of broad recorded speech repositories. Each documented utterance is segmented into some or all of the following during the compilation of a database: individual phones, diphones, half-phones, syllables, morphemes, verbs, phrases, and sentences. Typically, division into segments is done using a specially modified speech detector set to a "forced alignment" mode with some manual configuration. So Concatenative speech synthesis method involves the production of artificial human like speech from pre-recorded units of speech by phenomes, diphones, syllabus, words or sentences.[6]

2.1.2 Diphone synthesis

Diphone synthesis uses a minimal speech database containing all the diphones (sound-to-sound transitions) that occur in a language. The number of diphones depends on the phonotactics of the language: for example, there are about 800 diphones in Spanish and about 2500 in German. As only single instances of speech are available in speech database, inorder to obtain good quality of synthesised speech with prosody and naturalness, speech processing techniques are applied. PSOLA, TD-PSOLA, LP-PSOLA, ESNOLA, FD-SOLA are some of the common techniques used for obtaining good qualiy synthesised speech. [6]

2.1.3 Formant synthesis

Formant Synthesis does not use human speech samples during runtime. Instead, a synthesized speech output is generated using an additive synthesis and an acoustic model (physical modeling synthesis). Parameters such as fundamental frequency, voice and noise levels vary over time to create a waveform of artificial speech. This method is sometimes referred to as a rule-based synthesis; however, many concatenative systems also have a rule-based component. Many systems based on formative synthesis technology generate artificial, robotic-sounding speech that would never be mistaken for human speech. Formative synthesis systems have advantages over concatenative systems. Formant-synthesized speech can be accurately intelligible, even at very high speeds, eliminating acoustic errors that often affect concatenative systems. High-speed, visually impaired speech is used to easily access devices using a screen reader. Formant synthesizers are usually smaller than Concatenative speech synthesis. Formative systems have benefits over concatenative systems. Formant-synthesized speech can be precisely intelligible, even at very high speeds, removing acoustic errors that often impact concatenation systems. High-speed, visually impaired speech is used for easy access to computers using a screen reader.[7]

2.1.4 HMM based synthesis

HMM-based synthesis is a synthesis process based on secret Markov models, also known as the Statistical Parametric Synthesis. In this method, the frequency spectrum (vocal tract), the fundamental frequency (voice source) and the length (prosody) of speech are simultaneously modeled by HMMs. Speech waveforms are generated from HMMs themselves on the basis of the maximum probability criterion.[12]

2.1.5 Articulatory synthesis

Articulatory synthesis refers to computational techniques for speech synthesis based on models of the human vocal tract and the articulation processes that occur there. The first articulatory synthesizer to be used routinely for laboratory experiments was created at Haskins Laboratories in the mid-1970s by Philip Rubin, Tom Baer and Paul Mermelstein.This synthesizer, known as ASY, was based on the vocal tract models created by Paul Mermelstein, Cecil Coker, and colleagues at Bell Laboratories in the 1960s and 1970s. First articulatory synthesizer was introduced in 1958 by George Rosen at the Massachusetts Institute of Technology, M.I.T. (Klatt 1987).Articulatory speech synthesis refers to speech based on models of human vocal tract and articulation process.[8]

Articulation synthesis comprises of:

1. Generation of vocal tract movements
2. convert movement information into continuos succesion of vocal tract geometries
3. Generate acoustic signals from arcticulatory information

2.1.6 Hybrid Text to Speech synthesis

The Hybrid TTS approach is a combination of the two main approaches of synthesis: Concatenative synthesis and Statistical Synthesis. The hybrid TTS combines the characteristics of smooth transitions between adjacent speech segments of a Statistical TTS with the naturalness of a Concatenative TTS. This is achieved by interweaving natural speech segments and statistically generated speech segments. The statistical segments are positioned so as to smooth discontinuities in the synthesized speech, while enabling as far as possible natural speech sequences as they appear in the training inventory disadvantages of this system are the degradation in speech quality when TTS speech inventory is small and more signal processing requirement [9].

2.1.7 Statistical Parametric Synthesis

Statistical parametric synthesis makes use of averaged acoustic inventories that are extracted from the speech corpus.The extracted parameters of speech are the spectral parameters such are cepstral coefficients or line spectral pairs, and excitation parameters such as fundamental frequency. Statistical Parametric synthesis has the advantages of requiring less memory to store the parameters of the model, rather than the data itself and it allows more Variation in the speech produced for example, an original voice can be converted into another voice. The most commonly used statistical parametric speech synthesis technique is the Hidden Markov Model (HMM) synthesis and Unit Selection synthesis [10].

2.1.8 Effecient speech

A new model for audio synthesis WaveRNN is introduced in [4]. A single layer RNN with a softmax layer WaveRNN matches current SOTA of Wavenet models. Yet it's possible to reduce the N (no of layers) and operations in GPU. A technique of weight pruning is applied to reduce number of weights in WaveRNN. A sparse WaveRNN makes it possible to sample high fidelity audio on a mobile CPU in real time. A new generation scheme based on sub scaling that folds a long sequence into batches of shorter sequences and allows one to generate multiple samples at a time. The subscale WaveRNN produces 10 samples per step without loss of quality and offers an orthogonal method for increasing sampling efficiency. WaveRNN demonstrates that a simple recurrent neural network for sequential modelling of high fidelity audio, and demonstrated a high performance implementation of this model on GPUs. It's shown that large sparse models have much better quality than small dense models with the same number of parameters and we have written high performance block-sparse matrix-vector product operations to demonstrate that sampling time is proportional to parameter count. It showed that high fidelity audio generation is now achievable on widely available low-power mobile CPUs. Finally, this paper introduced the subscale dependency scheme that lets sequential models generate many samples per step while preserving the output quality of the original model. The underlying ideas of the methods we introduce are not specific to audio, and the results of sparse models have implications for inference in all types of neural networks.

2.1.9 Wavenet Based Models

In September 2016, Deep Mind proposed WaveNet, a deep generative model of raw audio waveforms [3]. This shows the community that deep learning-based models have the capability to model raw waveforms and perform well on generating speech from acoustic features like spectrograms or spectrograms in Mel scale, or even from some preprocessed linguistic features. The model is fully probabilistic and self-regressive, with the predictive distribution of each audio sample conditioned on all previous ones. Data with tens of thousands of samples per second of audio can be efficiently trained. Applied to text-to-speech, it produces state-of - the-art performance, with human listeners rating it as significantly more natural than the best sound parametric and concatenative systems can produce for both English and Mandarin. Wavenet model triggered a huge development of Text to speech architectures based on Deep Learning which is mentioned in [11].

2.1.10 Fast Speech

Compared with traditional concatenative and statistical parametric approaches, neural network based end- to-end models suffer from slow inference speed, and the synthesized speech is usually not robust (i.e., some words are skipped or repeated) and lack of controllability (voice speed or prosody control). In this work [5], a novel feed-forward network based on Transformer is used to generate Mel-spectrogram in parallel for TTS. Specifically attention alignments are extracted from an encoder-decoder based teacher model for phoneme duration prediction, which is used by a length regulator to expand the source phoneme sequence to match the length of the targeted Mel-spectrogram sequence for parallel Mel-spectrogram generation. Experiments on the LJ-Speech dataset show that our parallel model matches autoregressive models in terms of speech quality, nearly eliminates the problem of word skipping and repeating in particularly hard cases, and can adjust voice speed smoothly. Most importantly, compared with autoregressive Transformer TTS, our model speeds up Mel- spectrogram generation by 270x and the end-to-end speech synthesis by 38x. Therefore, this method is called Fast speech.

Overview-Methods of Text to Speech
Concatenative Synthesis
Diphone Synthesis
Formant Synthesis
HMM based synthesis
Articulatory synthesis
Efficent Speech
Wavenet based models
Fast Speech

2.2 Literature Survey

2.2.1 Text Normalisation for Bangla, Khmer, Nepali, Javanese, Sinhala and Sudanese TTS [13]

The main contributions of this paper[13] are as follows:

- describe a general method for working with native speakers in identifying patterns and grammars needed to normalize text.
- making available text normalization grammars and their test cases for a wide range of common semiotic classes for 6 low-resourced languages.
- provide a recipe for utilizing these grammars and for integrating them into actual text-to-speech systems.

Text normalization takes plain text as input for any language. It takes input words being separated each other using whitespaces while for languages which do not use whitespaces to separate works - like Khmer, the input is the output after passing text into word segmenter. Text normalization is divided into two phases. First, input text is analyzed and NSWs are classified into semiotic classes. In this phase, some input tokens may be grouped together. For example, input text “15 km” may be grouped together and are classified as a measurement token. Then, verbalizer grammar for each semiotic class will convert the classified NSWs into standard text accordingly After identifying the semiotic classes, they reached out to Native speakers in each language with a questionnaire. The questionnaire contains a set of questions for each semiotic class. The questions were designed to capture the writing and verbalizing system of the language. They took into consideration of various language-specific characteristics that affect text normalization. For each language there are particular characteristics for the same. In case of Bangla: different inflection cases are handled and 4 different time indicators similar to “am/pm”. Besides the above considerations, all languages in this set, other than Javanese and Sudanese, have their own alphabets and digits. Bangla uses the Bengali alphabet. Khmer uses the Khmer script. Nepali uses Devanagari and Sinhala uses the Sinhala alphabet. Khmer writing contains inconsistent usages of the zero-width space character. Bengali, Nepali and Sinhala also utilize zero-width non-

2.2.2 Corpus Driven Malayalam Text-to-Speech Synthesis for Interactive Voice Response System[14]

In paper [14] written by Arun Soman, etc., a corpus-driven Malayalam text-to-speech (TTS) system based on the concatenative synthesis approach is explained. The most important qualities of a synthesized speech are naturalness and intelligibility. In this system, words and syllables are used as the basic units for synthesis. The corpus consists of speech waveforms that are collected for most frequently used words in different domains. The speaker is selected through subjective and objective evaluation of natural and synthesized waveform. The proposed Malayalam text-to-speech system is implemented in Java multimedia framework (JMF) and runs on both in Windows and in Linux platforms. The proposed system provides utility to save the synthesized output. The output generated by the proposed Malayalam text-to-speech synthesis system resembles natural human voice. Text to speech reader software converts a Malayalam text to speech wav file that has high rates of intelligibility and comprehensibility. The corpus was collected by a single best speaker wave form. The best speaker means the speech produced by that speaker have capabilities with respect to energy profile, speaking rate, pronunciation and into nation. Input Malayalam text was first text normalized which made the input text words in non- standard form like Numbers, dates, etc. and punctuations were also removed. Then process of sentence splitting took place in the paragraphs and words were separated out. Romanization is the representation of written word with a roman alphabet. In this system Romanized form of Malayalam words/syllables are generated. For representing the written text the method used for Romanization is transliteration and for spoken word, the method is transcription. The final stage is the concatenation process. All the arranged speech units are concatenated using a concatenation algorithm. The concatenation of speech files is done in java media framework. The main problem in concatenation process is that there will be glitches in the joint. The concatenation process combine all the speech file which is given as a output of the unit selection process and then making in to a single speech file. This can be played and stopped any where needed.

2.2.3 Orthography with Maratha[15]

Speech synthesis models are typically built from a corpus of speech that has accurate transcriptions. Paper [15] consider many of the languages of the world do not have a standardized writing system.

This paper is an initial attempt at building synthetic voices for such languages. It may seem useless to develop a text-to-speech system when there is no text available. A novel method to build synthetic voices from only speech data is shown here. Experimental results and oracle studies shows that it is possible to automatically devise an artificial writing system for these languages, and build synthetic voices that are understandable and usable.

The speech data in a target language with no well-defined orthography for transcriptions is given. A simple method to deal with this situation is to run an automatic speech recognizer over available speech data and use its output as transcriptions. The caveat with using a speech recognizer is that because our target language does not have a text form, a speech recognizer will not exist in that language. We hence have to use a speech recognizer in another language: a language that has an orthography, and large corpora to train speech recognizers. This presents another caveat: that is recognizing in a different language than the models are trained for. Using the default language model is thus not ideal, and we need to adapt it so that it is suitable for our target language. It also uses phonetic decoding instead of word level decoding.

The paper [15] solution proposes the following:

- Choose an appropriate acoustic model for speech recognition,
- Choose a language that has an orthography and is phonetically close to our target language, and then build a phonetic language model on text in this language.
- Run phonetic decoder on our target speech data with these two models and obtain transcripts.
- Buildvoice using the speech data and the phonetic transcripts just obtained.

The paper has addressed a novel problem of building speech synthesizers for languages without an orthography. In the solution proposed automatically developing a writing system for the language, using a speech recognition system. The iterative method to build targeted acoustic models yield very good improvements in synthesis quality. The objective and subjective results, as well as oracle results on Marathi, which show that direction to building synthesis models without written text is promising. We also showed similar results on Hindi and Telugu, thus showing that our methods are language independent.

2.2.4 Maithili Text to Speech system[16]

This paper describes the method of creating a text-to-speech system for Maithili, which is a similar variant of Hindi. As most Indian languages, Maithili is syllabic in nature and concatenative method is used for purpose of speech generation taking speech syllable as the basic unit. The concatenate Unit Selection Synthesis (USS) technique is used. Naturalness of USS for small amount of data is better compared to other methods.

The workflow of this TTS made by Amit Kumar Jha, etc is:

1. Input Maithili text written in Devanigiri script using UTF-16
2. Inputted text is normalized with the help of three
3. Inputted text is segmented into sentence level. Afterwards, it is segmented into word level using white space.

4. A word level search is done in database and if it is found then corresponding speech file is added into playlist. Else, the word is broken into corresponding syllables and corresponding syllables files are searched and added in playlist.
5. Found speech units are concatenated in playlist using digital signal processing.
6. Add prosodic features to the speak file according to the types of sentence.
7. Play the sound of playlist.

The process of text normalisation for Maithili language was done for this paper and this neatly in [2]. To increase the naturalness of TTS system, 1055 most frequently occuring word have been recorded and stored in separate lexicon. The system support UTF-16 for text input and a C#.NET interface is used for developing TTS system in Maithali.

The speech database consists of 930 syllable (C*V) in total. Each position has 300 syllables and 10 independent vowels. Each position has 300 syllables and 10 independent vowels. 930 units of speech data is build all three positions. This is the first TTS system which exists for Maithili language till date.

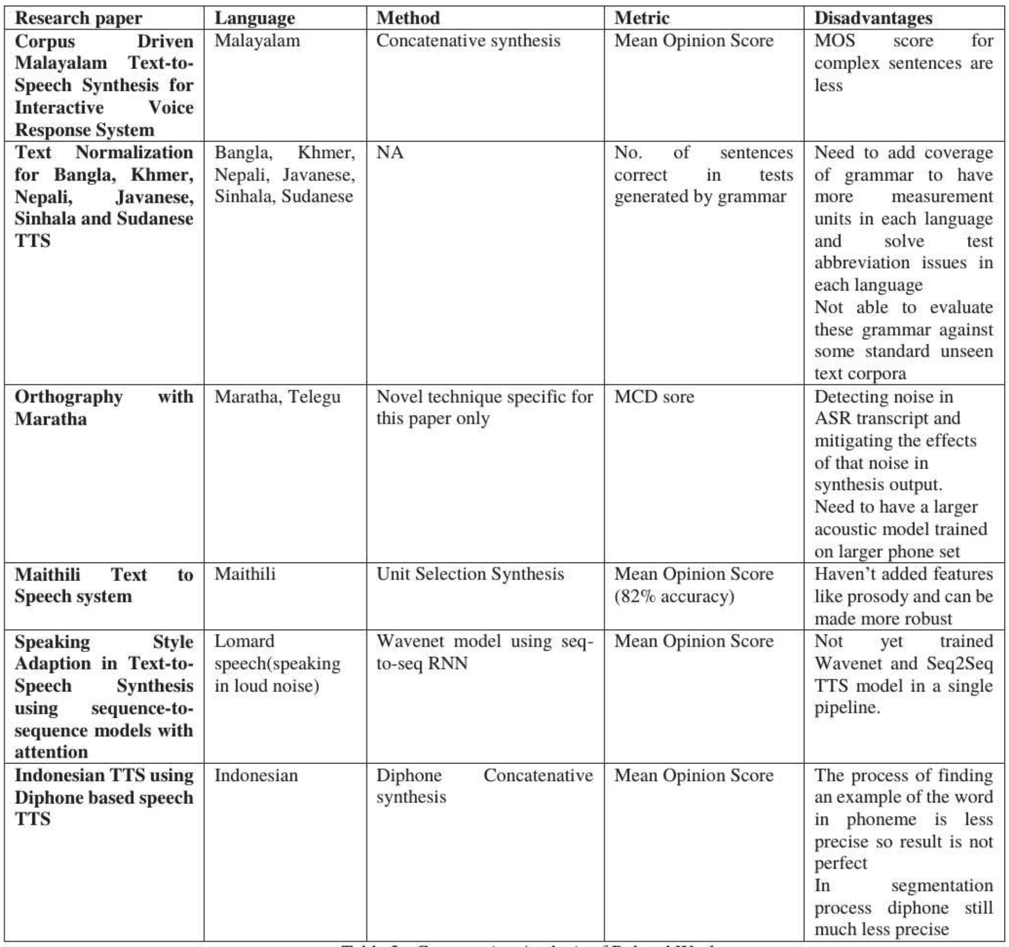
2.2.5 Speaking Style Adaption in Text-to-Speech Synthesis using sequence-to-sequence models with attention [17]

E. Speaking Style Adaption in Text-to-Speech Synthesis using sequence-to-sequence models with attention [17] Neural networks which are data driven is good in Text to Speech synthesis. The paper [17] is aimed for challenging speaking styles like Lombard speech (speaking in loud voice) where it's difficult to generate large corpora. A new transfer learning method which adopt Lombard style from Normal speaking style. They use a learning method to adopt sequence to sequence based TTS system of normal speaking style to Lombard style. Moreover on evaluation results indicated that an adaption system with Wavenet Vocoder clearly outperformed conventional deep neural networks based on TTS.

Here TTS system uses a sequence to sequence model with attention. The model accepts either mono-phonemes or graphemes as inputs and emits acoustic parameters as outputs. It consists of three main components: 1) Encoder 2) attention and 3) decoder. The encoder takes text sequence x of length L as input, which represented either in the character or phoneme domain as one-hot vectors. The encoder learns a continuous sequential representation h using various neural network architectures such as LSTMs or CNN. At each output time step t, both the attention and decoder modules work together. The decoder takes the previous hidden state and current context vector as inputs and generates the current output. The process runs until the end of the utterance is reached. The accuracy of recognizing Lombard speech by our TTS is 95

2.2.6 Indonesian TTS using Diphone based speech TTS[18]

Paper [18] shows a novel approach to create a text to speech system for Indonesian language. This approach first creates a database of Indonesian diphone at first. Indonesian diphone synthesis uses speech segment of recorded voice from text to speech and save it as audio file like WAV, MP3. First a diphone databases is created and then convert text to speech from input of numbers, words. They used diphone concatenative synthesis in which recorded segments were collected. The two main contributions of this paper are: First developed a diphone database including creating a list of words consisting of diphones organized by prioritizing diphones in this system. Second develop system using Microsoft visual Delphi 6.0, includes: the conversion system from the input of numbers, acronyms, words, and sentences into representations diphone. There are two kinds of conversion (process) alleged in analyzing the Indonesian text-to-speech system. One is to convert the text to be sounded to phoneme and the other convert the phoneme to speech. Method used in this research is called Diphone Concatenative synthesis, in which recorded sound segments are collected. Every segment consists of a diphone (2 phonemes). This synthesizer may produce voice with high level of naturalness. The Indonesian Text to Speech system can differentiate special phonemes like in 'Beda' and 'Bedak' but sample of other specific words is necessary to put into the system. This Indonesia TTS system can handle texts with abbreviation, there is the facility to add such words. Indonesian Text-To-Speech System using diphone concatenative synthesis can produce speech or language the natural approach. Speech or resulted sound may not be perfect for several causes such as poor recording and distorted phoneme segmentation process where front, middle and end borders are not synchronized.



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

2.3 Proposed System

Our proposed system will attempt to build a text to speech system/ speech synthesis software using the state of the art methods of speech synthesis. With this system are planning to build a text to speech software in which there are two phases, that is:

- Input text to phenome conversion in form of mel-spectograms
- The phenoes are converted to speech

This is the idea of the proposed system using Tactron architecture, which is the one of the best real-time text to speech synthesis methods currently in English. We are trying to replicate this result in English language.

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

The aim of this project is to create an accurate and robust Text-to Speech system for Malayalam. Most of the earlier TTS systems in Malayalam are not so good now and isn't able to produce speech with decent quality.This project aims to build a Text to speech system not just specifically only for Malayalam, yet for a generalised system for low resource languages.

3.1.2 Document Conventions

The document was created based on the IEEE template for System Requirement Specification Documents. When writing this SRS, the following conventions are used:

- Bold-faced text has been used to emphasize sections and sub-section headings.
- Italicized text is used to label and recognize diagrams.
- Commonly used acronyms are mentioned here:

TTS - Text to Speech system

3.1.3 Intended Audience and Reading Suggestions

Primary readers of this document are the developers and the mentors overseeing the project. The project is not build focusing on the end to end products which can leverage TTS as applications, instead the focus is on building a infrastructure. In the next section system features with their functional requirements are presented to highlight the major services provided by the intended product. Then the external interface requirements highlighting the logical characteristics of each interface between the software product and the users are discussed. Finally, this specification is concluded with the reference documents on which this document is based on.

3.1.4 Project Scope

The scope of creating a text to speech system for Malayalam enables for purposes like improving accessibility applications for blind people.

3.1.5 Overview of Developer's Responsibilities

The developer will be responsible for implementing the following features:

- Build a text to speech system for Malayalam
- Create Python CLI for the core TTS engine
- Build a web api for core TTS engine
- Write clean code which should be open-sourced
- Document the software for public use

3.2 Overall Description

3.2.1 Product Perspective

The product has been build with the perspective to provide a reliable infrastructure for a Text to speech system in Malayalam. Previous TTS system developed for Malayalam was Dhavani, in 2008. For the past ten years, there has been no progress in creating an effective TTS which produce clear and robust sound voice for Malayalam incorporating the latest advances in technology.

3.2.2 Product Functions

Malayalam TTS software will provide a reliable interface, which can generate human-like voice from Malayalam words, which are quite complex in terms of words. The proposed software should have the following functions:

- For a given text in Malayalam generate speech in malayalam
- Generate the adequate text for given voice system

3.2.3 User Classes and Characteristics

The users of this software will only include the end users:

- Users:Any users that needs to use TTS for their purpose
- Application Developers: The developers can build products on top of Core TTS system for Malayalam

3.2.4 Operating Environment

Server side:
The server-side components of the software system must operate within a Linux operating system environment.
Client side:
We are planning to develop a python based CLI tool which has the core TTS engine which can be used for Application developer. A web based client-side components of the software system using TTS must operate within common web browser environments.The following browsers must be supported.

- Google Chrome 45+
- Microsoft Internet Explorer 11+
- Microsoft Edge 12+
- Apple Safari 10+
- Mozilla Firefox 40+

3.2.5 Design and Implementation Constraints

It should use common libraries and tools that can work with all the common internet browser applications with no problem.

3.2.6 User Documentation

User manual will be made available for troubleshooting and help.It would follow common user documentation styles capturing purpose and scope of the product along with key system features and operations step by step instructions for using the system including conventions,messaging structures,quick references,tips for error and malfunctions;pointers to reference documents and glossary of terms.

3.2.7 General Constraints

The components of the system must communicate well with each others as output from one module is given as input to another module. Constant update of database should be done The server must run without a time constraint A stable internet connection is required

3.2.8 Assumptions and Dependencies

- The applications requires a stable network connection.
- Good quality data should be procured for the application

3.3 External Interface Requirements

3.3.1 User Interfaces

The user interface for this software involves following components:

- A Web API can be designed for the product, which responds back with the speech of a given person
- A python based CLI tool which has the core TTS engine which can be used for Application developer

3.3.2 Hardware Interfaces

No specific hardware interface is required except for the device being used to open the application.

3.3.3 Software Interfaces

The core TTS engine developed will be integrated with python based CLI tool and web API.

3.3.4 Communication Interfaces

The communication between the different parts of the system is important,since they depend on each other.However in what way the communication is achieved is not important for the system and is therefore handled by underlying operating system for the application.

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirements

Hardware required for developing TTS:

- Operating system:Debian 10(Buster
- RAM:4 GB or more
- Processor:2.3GHz Intel i5 or more
- 1 TB Hard disk
- CUDA supported GPUs(Nvidia 940MX)

3.4.2 Software Requirements

- Python - Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows to write reliable systems.Python is a widely used high-level programming language for general-purpose programming. Apart from being open source programming language, python is a great object-oriented, interpreted, and interactive programming language. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation). Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces

- 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. It is primarily developed by Facebook's artificial intelligence research group.It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ frontend.Furthermore, Uber's Pyro probabilistic programming language software uses PyTorch as a backend. Most of the latest TTS open source implementation are made over this framework.

- 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.

- Datasets - Open SLR dataset, MSC etc

TBD based on development on tool

3.5 Functional Requirements

3.5.1 Crowd-sourced Dataset

In a low resource language limited amount of data can only be created. We need to consider language's prounciation lexicon for a given text and a set of phonemes should be generated for the language. This initial step is very important.The dataset should be relative recent so as to express the new terms and slangs of the language.

3.6 TTS Engine

For creating the core TTS for malayalam, the given Malayalam text input should be passed through following stages:

3.6.1 Text Normalisation

3.6.2 Text to spectrogram representation

3.6.3 Generate audio speech from spectorgam

3.7 Non-functional Requirements

3.7.1 Performance Requirements

Accuracy is of prime importance.The system should classify the documents with as much accuracy as possible.The extraction and classification time and response time should be as little as possible.

3.7.2 Safety Requirements

The collected information should be securely transmitted to the next level without any changes in the information.

3.7.3 Security Requirements

The software system defined in the SRS must follow industry recommended practices for secure software development.No tampering of images and data should be made.The data held within the system should be confidential.The user should not be able to manipulate the working of the system i.e.,the system should be closed from outside.

3.7.4 Software Quality Attributes

- Reliability
  The solution should provide reliability to the the user that the product will run with all the features mentioned in this document are available and executing perfectly.It should be tested and debugged completely.All exceptions should be well handled.The system should never crash or hang,other than as the result of an operating system error.

- Accuracy
  The system should be able to reach the desired level of accuracy in classification of documents.

- Availability
  The system should be available to user irrespective of date and time .It should be able to deliver the requested services without delays at any random time.

- Maintainability
  All code shall be fully documented.All program files shall include comments concerning authorship and date of last change.The code should be modular to permit future modifications.Anticipated updates include changes to the sets of objects and their description.

3.7.5 Ease of use

The system should be easy to handle with no delays.

3.7.6 Error Handling

The system should be tolerant i.e no hang ups against errors,delays etc.Page not found and error handling should be shown in case of any content not found or some unexcepted error.

Chapter 4

System Design

4.1 System Architecture

In this section, we will walk through the architecture of the proposed system. At first, the sequence flow of generalised neural text to speech architecture is shown below

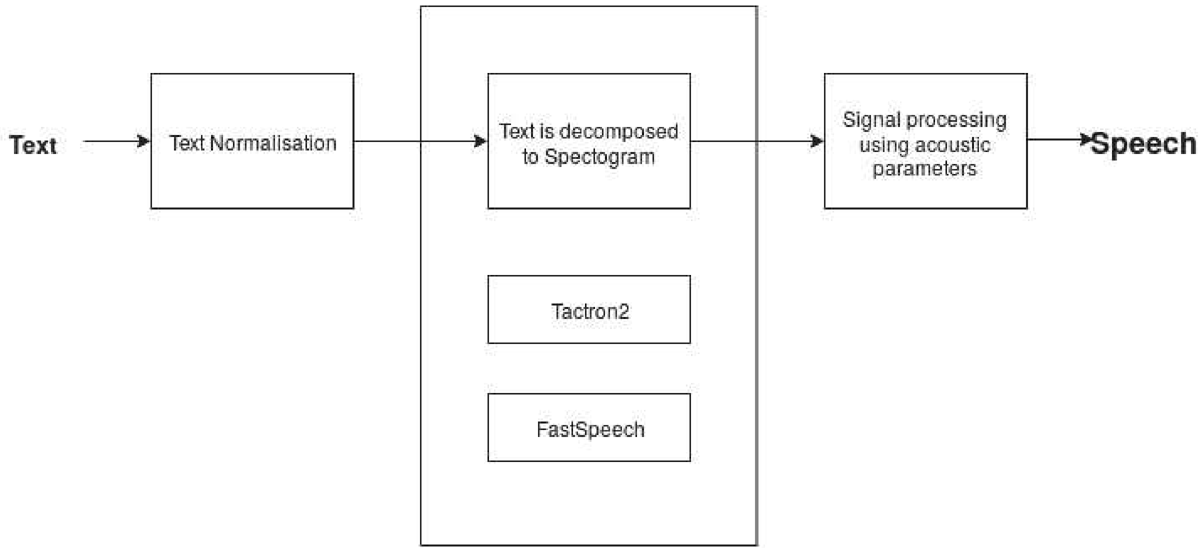

Text
Text Normalisation
Text is decomposed to Spectogram
Tactron2
FastSpeech
Signal processing using acoustic parameters
Speech

Figure 4.1: Generalised architecture of neuralised text to speech system

4.2 Text Normalisation

This is the module where the process of transforming text into a single canonical form that it might not have had before. Normalizing text before storing or processing it allows for separation of concerns, since input is guaranteed to be consistent before operations are performed on it.

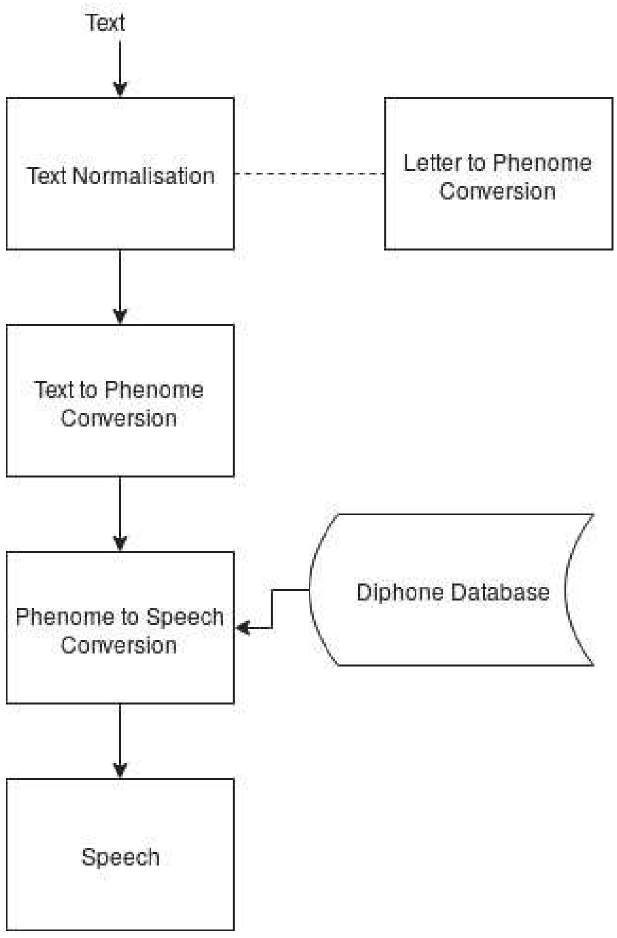
4.3 Normalised text decomposed to melspectogram

This module takes input text and converts it into Mel-spectogram. There are mainly auto-regressive and non-regrissive methods for the same. The architectures like Tactron2, Tactron from Google is already proven architecutures for efficent TTS. Yet newer non-regressive methods like FastSpeech holds out promise of better results.

4.4 Signal processing using acoustic character

Processing of signals from the melspectograms, and producing speech also happens in this phase.

In case this doesn't work out. There we can use Diphone speech synthesis method which is as shown in the figure:



Text
Text Normalisation
Letter to Phenome Conversion
Text to Phenome Conversion
Phenome to Speech Conversion
Diphone Database
Speech

Figure 4.2: Diphone Speech synthesis

In this architecture, a diphone database is to be created, which holds the individual speech tones of a given language for Phenome to Speech conversion of the speech. This architecture had shown good results for TTS in low resource rich languages like Indonesian.

4.5 Input Design

The input section consists of a webapp which takes the the text as the input through a form in our webapp. On pressing the submit button, the text is being stored into a database.

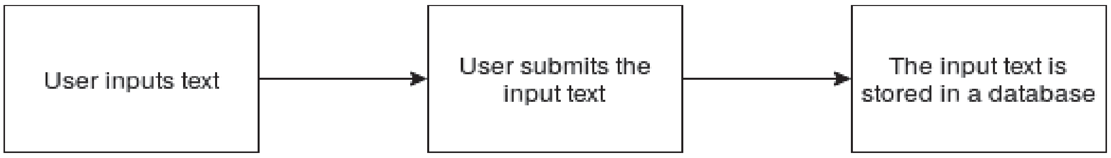

User inputs text
User submits the input text
The input text is stored in a database

Figure 4.3: Input Design

4.6 Database Design

The DB design of the core TTS engine is as shown in the figure. It consists of following tables:

- Users
- Text
- Speech

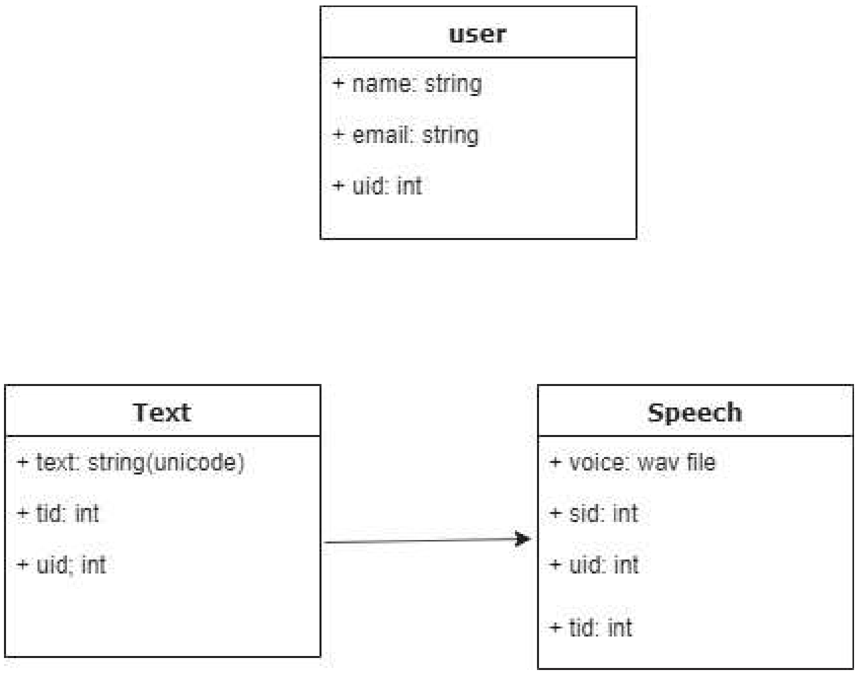


Figure 4.4: Database Design

4.7 Libraries and Packages Used

4.7.1 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).

[22]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

PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU and Tensor processing units(TPUs). PyTorch supports various sub-types of Tensors- adding, initialising randomly tensors and supports numpy indexing as well.

Pytorch has various modules like:

Autograd module

PyTorch uses a technique called automatic differentiation. That is, we have a recorder that records what operations we have performed, and then it replays it backward to compute our gradients. This technique is especially powerful when building neural networks, as we save time on one epoch by calculating differentiation of the parameters at the forward pass itself.

```
from torch.autograd import Variable

x = Variable(train_x)
y = Variable(train_y, requires_grad=False)
```

Optim module

torch.optim is a module that implements various optimization algorithms used for building neural networks. Most of the commonly used methods are already supported, so that we don't have to build them from scratch (unless you want to!).

Below is the code for using an Adam optimizer:

```
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

nn module

PyTorch autograd makes it easy to define computational graphs and take gradients, but raw autograd can be a bit too low-level for defining complex neural networks. This is where the nn module can help.

The nn package defines a set of modules, which we can think of as a neural network layer that produces output from input and may have some trainable weights.

You can consider a nn module as the keras of PyTorch!

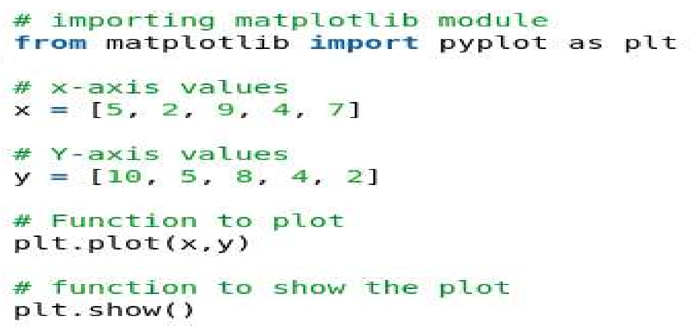
```
import torch

# define model
model = torch.nn.Sequential(
 torch.nn.Linear(input_num_units, hidden_num_units),
 torch.nn.ReLU(),
 torch.nn.Linear(hidden_num_units, output_num_units),
)
loss_fn = torch.nn.CrossEntropyLoss()
```

4.7.2 Numpy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. Different operations can be performed efficiently with numpy,Using numpy data can be more efficiently represented,More operations can be performed on numpy arrays.

4.7.3 Matplotlib

Matplotlib is used for plotting images defined using numpy arrays. matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.



```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot
plt.plot(x,y)

# function to show the plot
plt.show()
```

ıt :

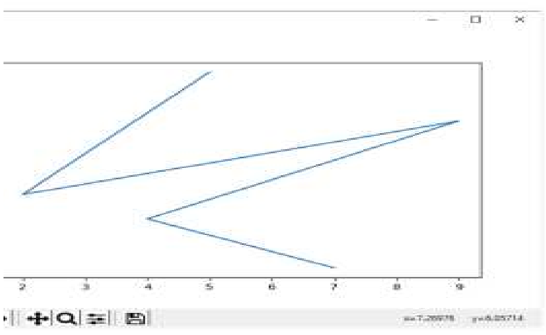


Figure 4.5: Example of plotting images using matplotlib

4.7.4 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.

The core Flite library was developed by Alan W Black awb@cs.cmu.edu (mostly in his so-called spare time) while employed in the Language Technologies Institute at Carnegie Mellon University. The name ”flite”, originally chosen to mean ”festival-lite” is perhaps doubly appropriate as a substantial part of design and coding was done over 30,000ft while awb was travelling, and (usually) isn't in meetings.

The voices, lexicon and language components of flite, both their compression techniques and their actual contents were developed by Kevin A. Lenzo lenzo@cs.cmu.edu and Alan W Black awb@cs.cmu.edu.

Some of the major features of flite are:

o Flite is designed for very small devices, such as PDAs, and also for large server machines which need to serve lots of ports.

o Flite is not a replacement for Festival but an alternative run time engine for voices developed in the FestVox framework where size and speed is crucial.

o Flite is all in ANSI C, it contains no C++ or Scheme, thus requires more care in programming, and is harder to customize at run time.

o It is thread safe

o Voices, lexicons and language descriptions can be compiled (mostly automatically for voices and lexicons) into C representations from their FestVox formats

o All voices, lexicons and language model data are const and in the text segment (i.e. they may be put in ROM). As they are linked in at compile time, there is virtually no startup delay.

o Although the synthesized output is not exactly the same as the same voice in Festival they are effectively equivalent. That is, flite doesn't sound better or worse than the equivalent voice in festival, just faster, smaller and scalable.

o For standard diphone voices, maximum run time memory requirements are approximately less than twice the memory requirement for the waveform generated. For 32bit archtectures this effectively means under 1M.

o The flite program supports, synthesis of individual strings or files (utterance by utterance) to direct audio devices or to waveform files.

o The flite library offers simple functions suitable for use in specific applications.

4.7.5 Festival

Festival Speech Synthesis System, developed at CSTR, University of Edinburgh. The project was originally started by Alan W Black and Paul Taylor but many others have been involved (see ACKNOWLEDGEMENTS file for full list).

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.

The system is written in C++ and uses the Edinburgh Speech Tools for low level architecture and has a Scheme (SIOD) based command interpreter for control. Documentation is given in the FSF texinfo format which can generate, a printed manual, info files and HTML.

4.7.6 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.

Some of the main features of espeak-ng are:

- Includes different Voices, whose characteristics can be altered.
- Can produce speech output as a WAV file.
- SSML (Speech Synthesis Markup Language) is supported (not complete), and also HTML.
- Compact size. The program and its data, including many languages, totals about few Mbytes.
- Can translate text into phoneme codes, so it could be adapted as a front end for another speech synthesis engine.
- Can be used as a front-end to MBROLA diphone voices. eSpeak NG converts text to phonemes with pitch and length information.

4.7.7 DVC

Data Version Control which one of best open source tools available in the market for Machine Learning Models and Dataset Versioning and other amazing features.It's is built to make ML models and datasets shareable and reproducible. It is designed to handle large files, data sets, machine learning models, and metrics as well as code.

It version controls machine learning models, data sets and intermediate files. DVC connects them with code, and uses Amazon S3, Microsoft Azure Blob Storage, Google Drive, Google Cloud Storage, Aliyun OSS, SSH/SFTP, HDFS, HTTP, network-attached storage, or disc to store file contents. Full code and data provenance help track the complete evolution of every ML model. This guarantees reproducibility and makes it easy to switch back and forth between experiments.

It harness the full power of Git branches to try different ideas instead of sloppy file suffixes and comments in code. Use automatic metric-tracking to navigate instead of paper and pencil.

DVC was designed to keep branching as simple and fast as in Git — no matter the data file size. Along with first-class citizen metrics and ML pipelines, it means that a project has cleaner structure. It's easy to compare ideas and pick the best. Iterations become faster with intermediate artifact caching.

For deployment and collabration, instead of using ad-hoc scripts, use push/pull commands to move consistent bundles of ML models, data, and code into production, remote machines, or a colleague's computer.

DVC introduces lightweight pipelines as a first-class citizen mechanism in Git. They are language-agnostic and connect multiple steps into a DAG. These pipelines are used to remove friction from getting code into production.

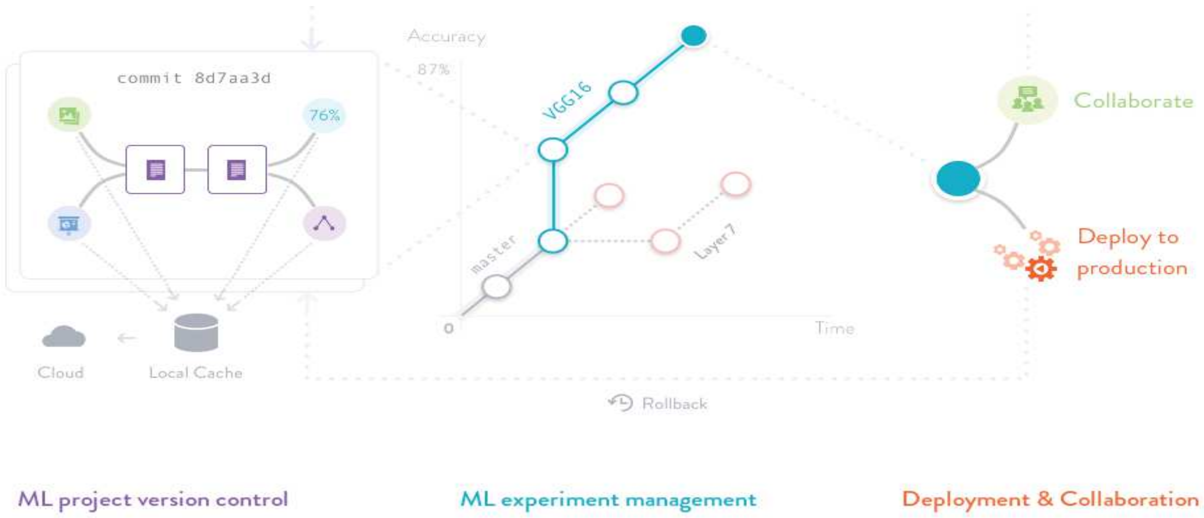


Figure 4.6: Features of DVC

4.7.8 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.

4.8 Module Description

4.8.1 EDA and Dataset Collection, Understand speech synthesis techniques

Dataset Collection

Some of the major datasets in Malayalam for speech synthesis are:

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

3. Another corpus contains 10 words in Malayalam corresponding to 10digits (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

Exploratary 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.

4.8.2 Train first TTS

Sequence to Sequence models

We initially attempted to use latest techniques like sequence to sequence models for the task of text to speech system. Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French). Similarly it converts the text to speech through this models. Some of the major components of Seq-2-seq models are:

- Encoder
- Decoder
- Attention

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.

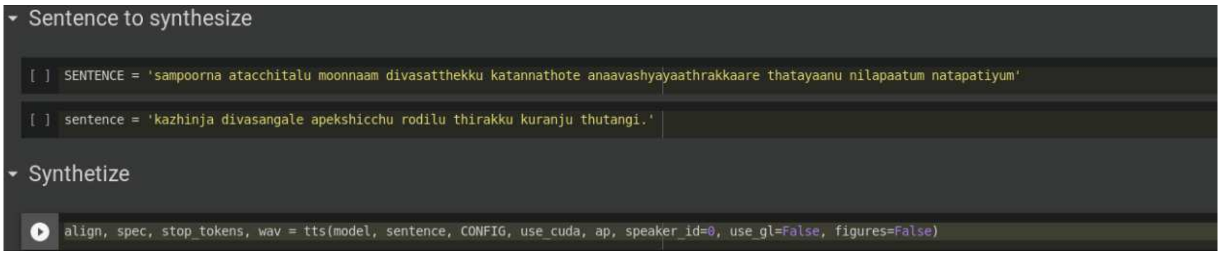
Our major aim is to understand and implement this architecture:

The major components of Tactron architecture are:

1. CBHG module
2. Encoder
3. Decoder

Malayalam TTS by transliterating input text and passing into English TTS

I 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 = 'sampoorna atacchitalu moonnaam divasatthekku katannathote anaavashyayaathrakkaare thatayaanu nilapaatum natapatiyum'
```

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

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

Figure 4.7: Transilerated malayalam text passed to TTS

4.8.3 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.

I 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 spectogram with frames computed every 12.5 milliseconds, capture the complex features. These features are finally forming a waveform using Wavenet like architecture.

The complete description of architecture can be found here:

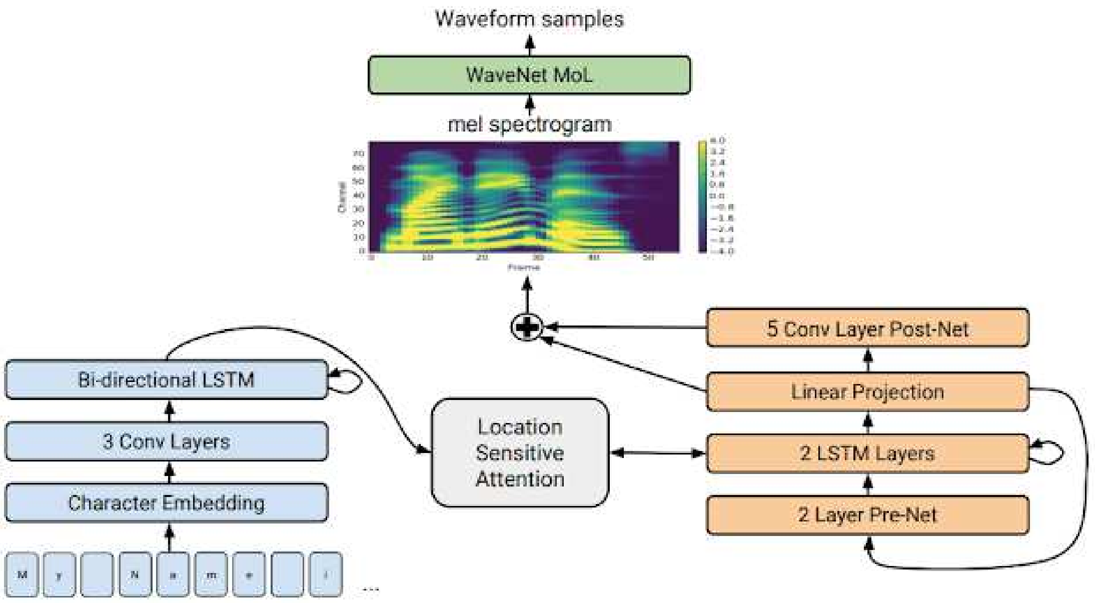


Figure 4.8: Detailed architecture of Tactron 2

A full description of our new system can be found in our paper “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.” In a nutshell it works like this: We 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

Using 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 barely understandable for even a simple human being.

Using 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 Kannada, a dravidian language and found that the results were quite good enough and quite audible, compared to exsisting approaches.

4.8.4 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

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.

Chapter 5

Data Flow Diagram

The various levels of data flows relevant for this project is as shown. The overall data flow of the project is as shown here:

5.1 Level 0 DFD

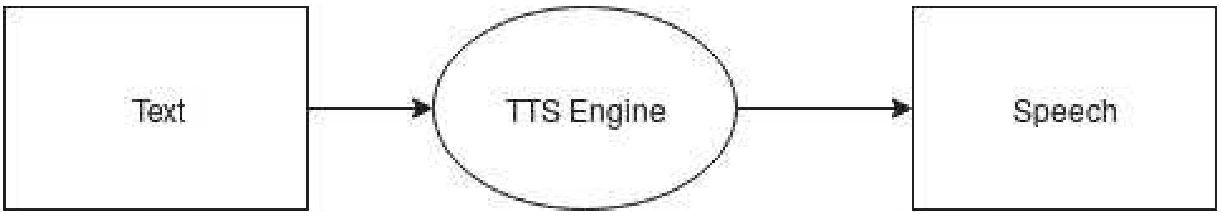


Figure 5.1: DF-0 diagram

The above data flow shows the basic figure of core TTS input and output. A more detailed expanded stage w.r.t on using the technique of Diphone Concatenative SPeech synthesis, is given below. The Data flow between various components is shown the figure.

5.2 Level 1 DFD

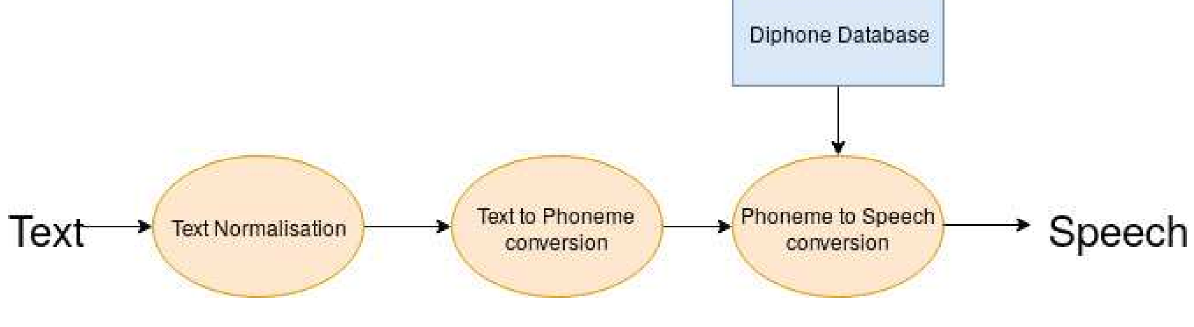


Figure 5.2: DF-1 diagram

Chapter 6

Implementation

6.1 Algorithms

For our implementation we used mainly three approaches:

1. Input text in malayalam is Transliterated and passed to English TTS
2. Implement tactron2 algorithm
3. Using eSpeachNg software for using malayalam TTS
4. using Flite architecture trained in Kannada for Malayalam TTS

6.1.1 Approach1

The input malayalam text is being transliterated ie process of conversion of a native language into English. The transliterated text is being passed into a TTS engine trained in English.

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
```

6.1.2 Approach 2

The input malayalam text uses Espeach-NG architecuture which use concatenative synthesis.

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
```

6.1.3 Approach 3

The input malayalam text uses festival voices trained in Kannada and use flite for speech production

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
```

6.1.4 Algorithm: Text Normalisation

1. Take the input word text
2. Identify tokens in the text like whitespace as seperator and punctuation to seperate from raw audio tokens
3. Identify ambiguity in abbrevation, 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.

6.1.5 Algorithm: Speech synthesiser in Tacron architecture

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

6.2 Development Tools

6.2.1 Microsoft Visual Studio

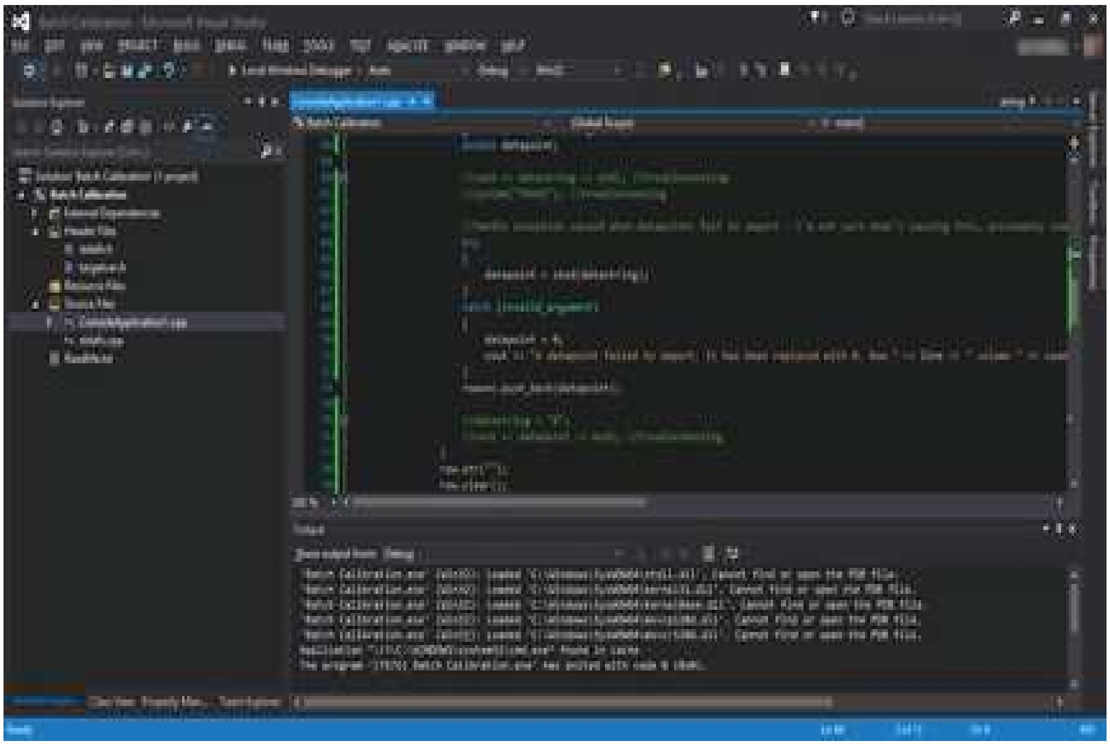


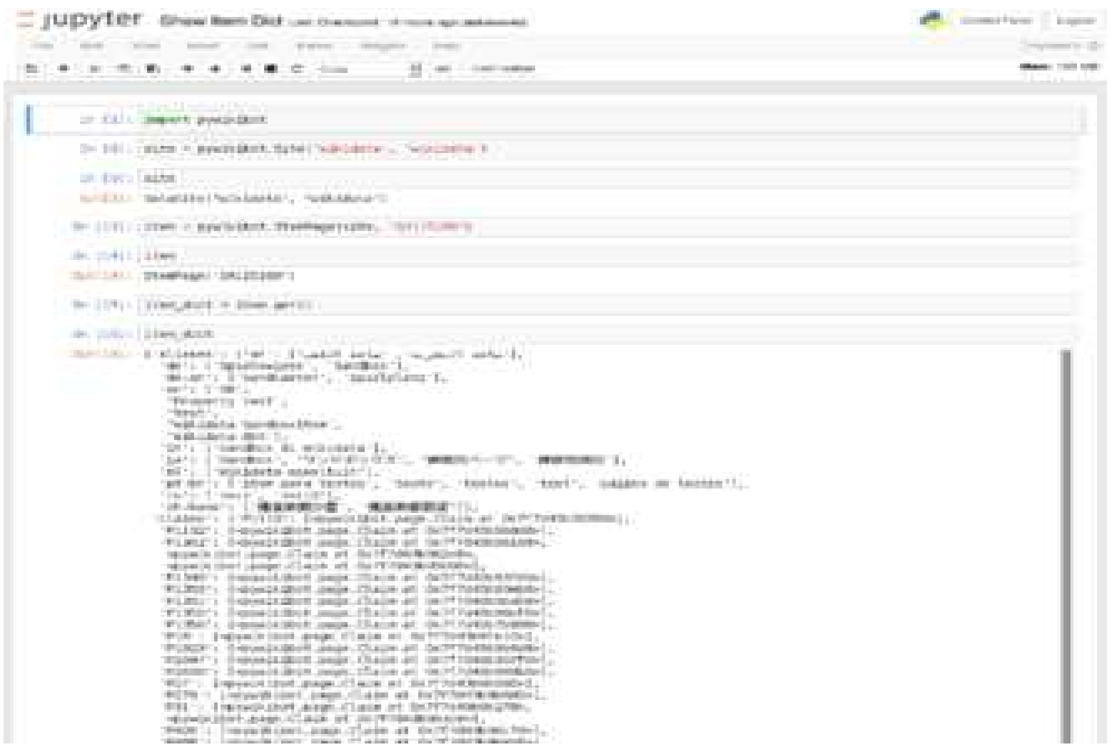
Figure 6.1: Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Azure DevOps client: Team Explorer). Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C,C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python,Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) were supported in the past.

The most basic edition of Visual Studio, the Community edition, is available free of charge. The slogan for Visual Studio Community edition is "Free, fully-featured IDE for students, open-source and individual developers".

6.2.2 Jupyter Notebook

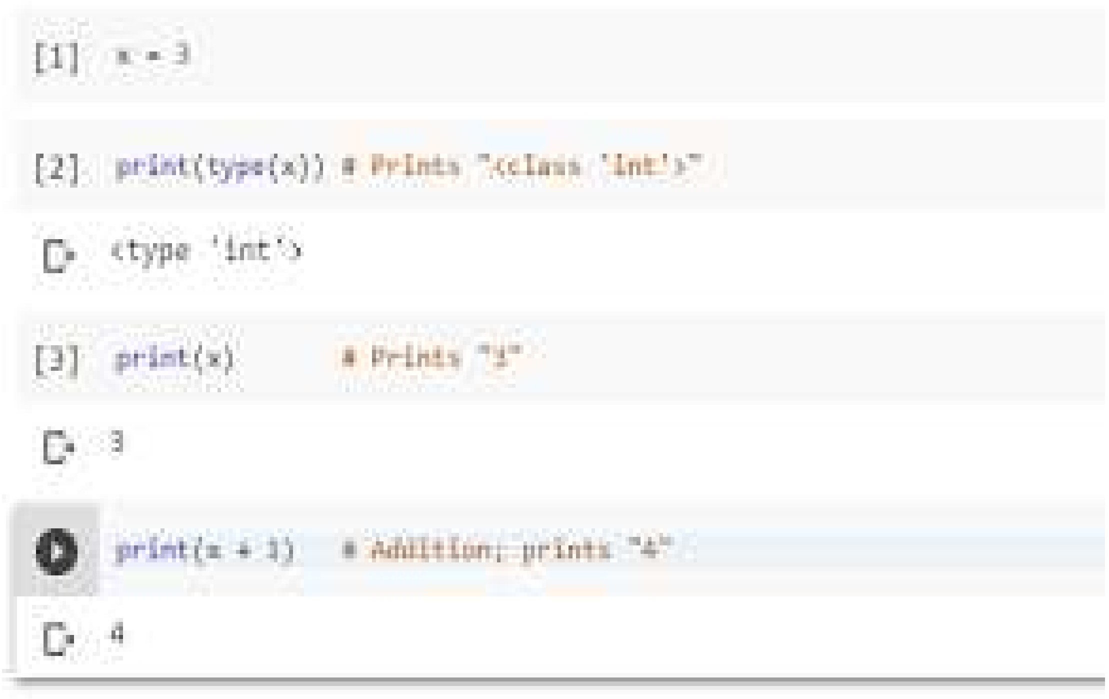


```
In [1]: import pywikibot
In [10]: site = pywikibot.Site('wikidata', 'wikidata')
In [11]: site
Out[11]: DataSite('wikidata', 'wikidata')
In [13]: item = pywikibot.ItemPage(site, [illegible])
In [14]: item
Out[14]: ItemPage([illegible])
In [15]: item_dict = item.get()
In [16]: item_dict
Out[16]: [illegible]
```

Figure 6.2: Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.



```
[1]  x = 3

[2]  print(type(x)) # Prints "<class 'int'>"

     <type 'int'>

[3]  print(x)       # Prints "3"

     3

     print(x + 1)   # Addition; prints "4"

     4
```

Figure 6.3: Google Collab

6.2.3 Google Colab

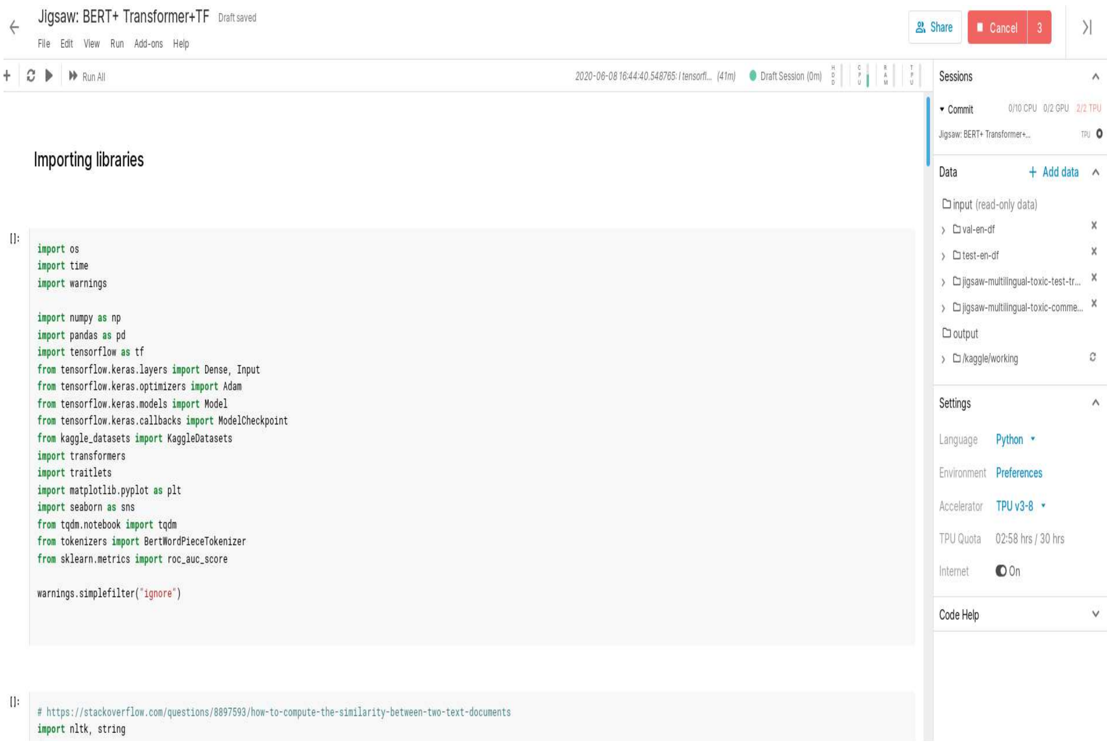
Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Colaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab.

Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

Irrespective of the reasons, the introduction of Colab has eased the learning and development of machine learning applications. Google Colab is a free cloud service and now it supports free GPU! You can; improve your Python programming language coding skills. develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV. The most important feature that distinguishes Colab from other free cloud services is; Colab provides GPU and is totally free.

6.2.4 Kaggle

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers,



Language Python
Environment Preferences
Accelerator TPU v3-8
TPU Quota 02:58 hrs / 30 hrs
Internet On

Importing libraries

```
import os
import time
import warnings

import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from kaggle_datasets import KaggleDatasets
import transformers
import traitlets
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm.notebook import tqdm
from tokenizers import BertWordPieceTokenizer
from sklearn.metrics import roc_auc_score

warnings.simplefilter("ignore")
```

```
# https://stackoverflow.com/questions/8897593/how-to-compute-the-similarity-between-two-text-documents
import nltk, string
```

Figure 6.4: Kaggle Kernels

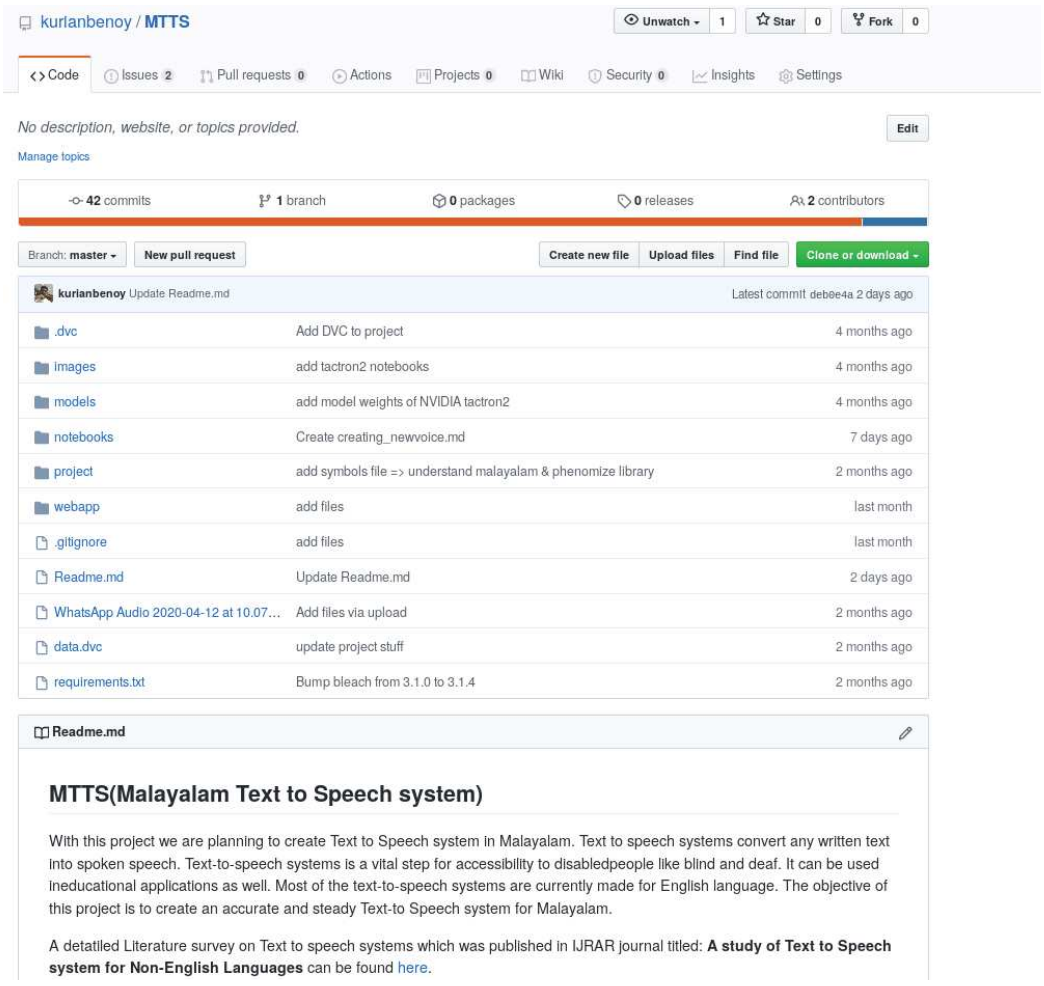
and enter competitions to solve data science challenges.

Some of the major Kaggle services are:

1. Machine learning competitions: this was Kaggle's first product. Companies post problems and machine learners compete to build the best algorithm.
2. Kaggle Kernels: a cloud-based workbench for data science and machine learning. Allows data scientists to share code and analysis in Python and R. Over 150K "kernels" (code snippets) have been shared on Kaggle covering everything from sentiment analysis to object detection.
3. Public datasets platform: community members share datasets with each other. Has datasets on everything from bone x-rays to results from boxing bouts.
4. Kaggle Learn: for short-form AI education.
5. Jobs board: employers post machine learning and AI jobs.

6.2.5 Github

Git is a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version 2. This tutorial explains how to use Git for project version control in a distributed environment while working on web-based and non web-based applications development. Version Control System (VCS)



No description, website, or topics provided.

42 commits · 1 branch · 0 packages · 0 releases · 2 contributors

kurianbenoy Update Readme.md — Latest commit debee4a 2 days ago

.dvc	Add DVC to project	4 months ago
images	add tactron2 notebooks	4 months ago
models	add model weights of NVIDIA tactron2	4 months ago
notebooks	Create creating_newvoice.md	7 days ago
project	add symbols file => understand malayalam & phenomize library	2 months ago
webapp	add files	last month
.gitignore	add files	last month
Readme.md	Update Readme.md	2 days ago
WhatsApp Audio 2020-04-12 at 10.07…	Add files via upload	2 months ago
data.dvc	update project stuff	2 months ago
requirements.txt	Bump bleach from 3.1.0 to 3.1.4	2 months ago

Readme.md

MTTS(Malayalam Text to Speech system)

With this project we are planning to create Text to Speech system in Malayalam. Text to speech systems convert any written text into spoken speech. Text-to-speech systems is a vital step for accessibility to disabledpeople like blind and deaf. It can be used ineducational applications as well. Most of the text-to-speech systems are currently made for English language. The objective of this project is to create an accurate and steady Text-to Speech system for Malayalam.

A detatiled Literature survey on Text to speech systems which was published in IJRAR journal titled: **A study of Text to Speech system for Non-English Languages** can be found here.

Figure 6.5: Github

is a software that helps software developers to work together and maintain a complete history of their work.

6.2.6 Postman

Postman is an API(application programming interface) development tool which helps to build, test and modify APIs. It has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH) and provides a nice GUI for constructing requests and reading responses. It is an extremely useful tool during the development and testing of APIs. Postman was used to test the API endpoints during development.

Chapter 7

Testing

7.1 Testing Methodologies

Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets user expectations. Test Methodologies include functional and non-functional testing to validate the Application Under Test. Each testing methodology has a defined test objective, test strategy and deliverables. We performed the following testing: Unit testing, Integration testing, System testing.

7.2 Unit Testing

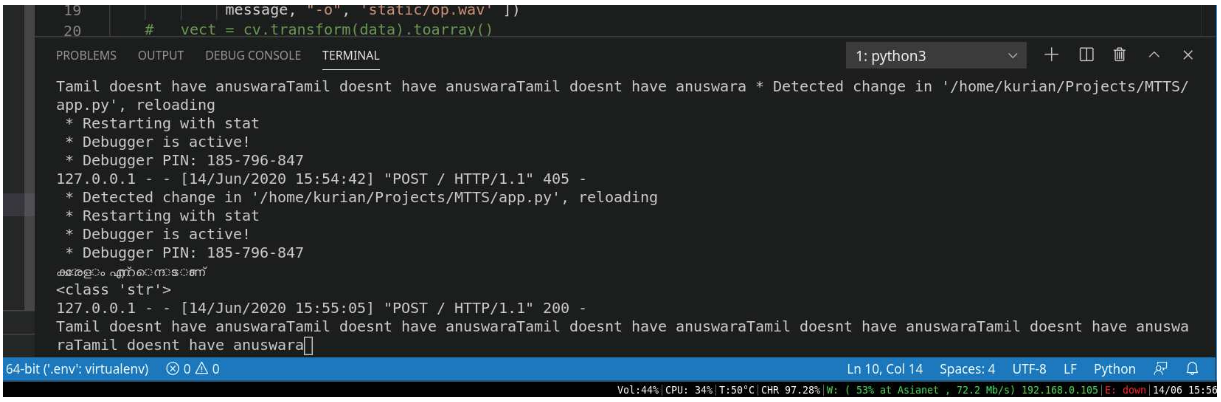
Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. We considered the following units for this process:

1. 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.

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

7.3 Integration Testing

Integration testing refers to the testing of the different modules/components that have been successfully 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.



```
            message, "-o", 'static/op.wav' ])
    #   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
```

Figure 7.1: The webapp with frontend and backend being integrated together, with no error

7.4 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.

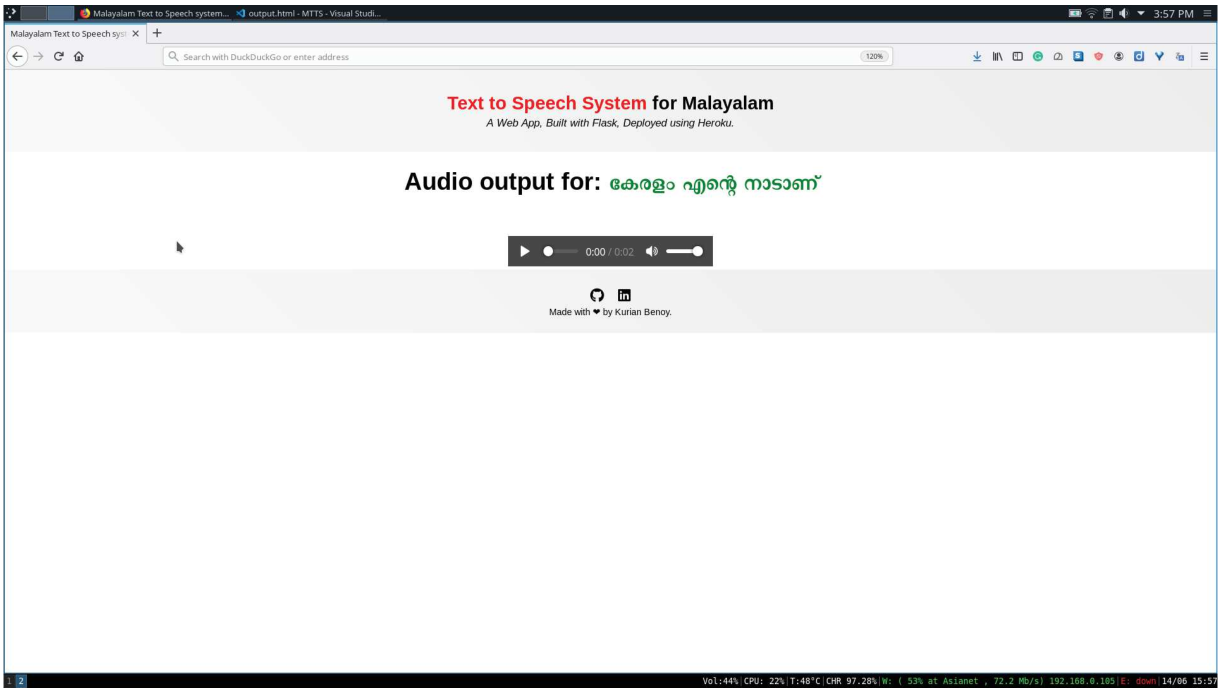

Malayalam Text to Speech system...
output.html - MTTS - Visual Studi...
3:57 PM
Malayalam Text to Speech syst
Search with DuckDuckGo or enter address
120%
Text to Speech System for Malayalam
A Web App, Built with Flask, Deployed using Heroku.
Audio output for: കേരളം എന്റെ നാടാണ്
0:00 / 0:02
Made with ❤ 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

Figure 7.2: UI with all the components being integrated together

Chapter 8

Graphical User Interface

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

GUI objects include icons, cursors, and buttons. These graphical elements are sometimes enhanced with sounds, or visual effects like transparency and drop shadows.

A GUI is considered to be more user-friendly than a text-based command-line interface, such as MS-DOS, or the shell of Unix-like operating systems.

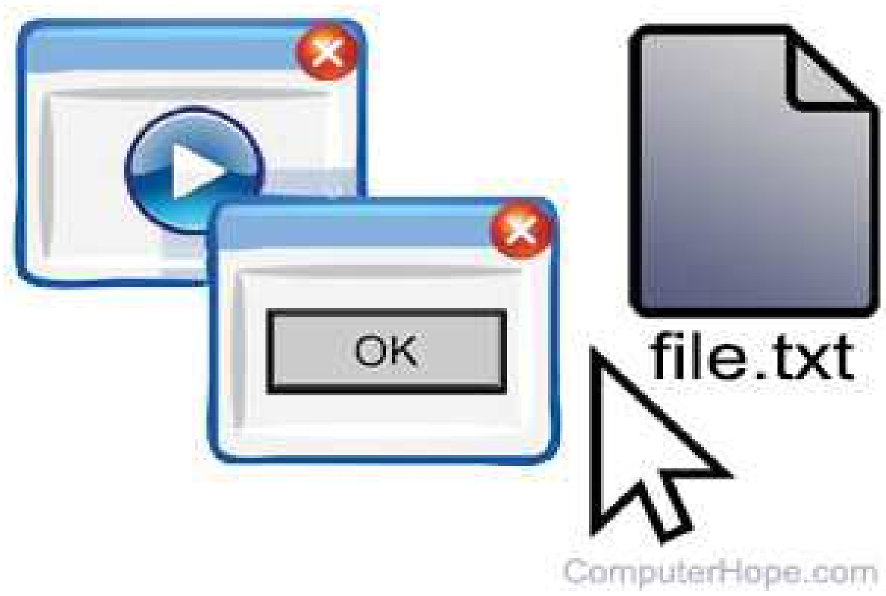


Figure 8.1: example GUI Icons

8.1 GUI Overview

For several decades, GUIs were controlled exclusively by a mouse and a keyboard. While these types of input devices are sufficient for desktop computers, they do not work as well for mobile

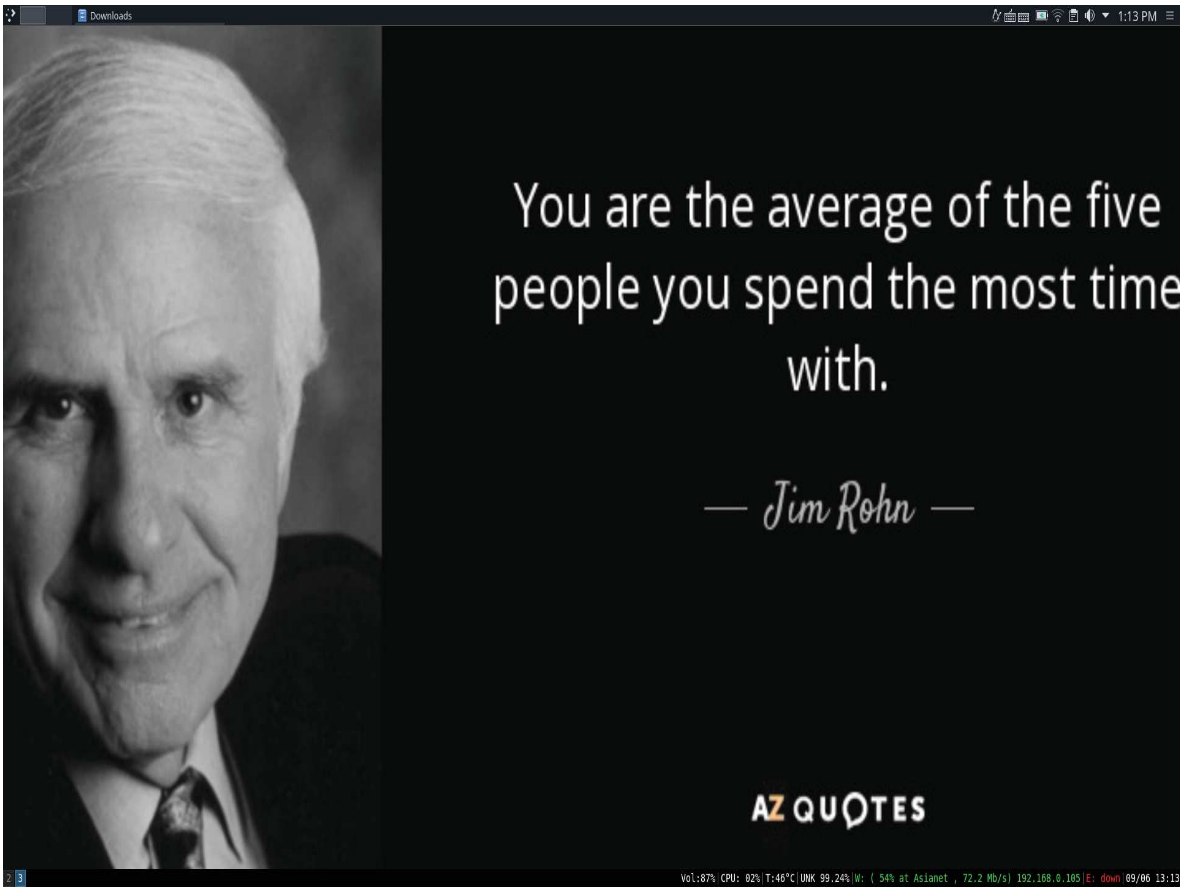

Downloads
1:13 PM
You are the average of the five people you spend the most time with.
— Jim Rohn —
AZ QUOTES

Figure 8.2: example of KDE Linux Desktop

devices, such as smartphones and tablets. Therefore, mobile operating systems are designed to use a touchscreen interface. Many mobile devices can now be controlled by spoken commands as well.

Because there are now many types of digital devices available, GUIs must be designed for the appropriate type of input. For example, a desktop operating system, such as OS X, includes a menu bar and windows with small icons that can be easily navigated using a mouse. A mobile OS, like iOS, includes larger icons and supports touch commands like swiping and pinching to zoom in or zoom out. Automotive interfaces are often designed to be controlled with knobs and buttons, and TV interfaces are built to work with a remote control. Regardless of the type of input, each of these interfaces are considered GUIs since they include graphical elements.

8.1.1 How does a GUI work?

A GUI uses windows, icons, and menus to carry out commands, such as opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a mouse, a keyboard can also be used via keyboard shortcuts or the arrow keys.

As an example, if you wanted to open a program on a GUI system, you would move the mouse pointer to the program's icon and double-click it.

8.1.2 Benefits of GUI

Unlike a command-line operating system or CUI, like Unix or MS-DOS, GUI operating systems are much easier to learn and use because commands do not need to be memorized. Additionally, users do not need to know any programming languages. Because of their ease of use and more modern appearance, GUI operating systems have come to dominate today's market.

8.1.3 User interact with a GUI

A pointing device, such as the mouse, is used to interact with nearly all aspects of the GUI. More modern (and mobile) devices also utilize a touch screen. However, as stated in previous sections, it is also possible to navigate a GUI using a keyboard.

8.1.4 Web Based GUIs

A browser user interface (or BUI) is a method of interacting with an application, typically hosted on a remote device, via controls presented within a web browser. This is an alternative to providing controls via a separate application with a dedicated graphical user interface (GUI) or command-line interface (CLI).

The project GUI consists of :

- Website

8.2 Main GUI components

8.2.1 Website

The website is accessible to normal users of Malayalam TTS as a basic user interface. The website contains just a home page to input the text.

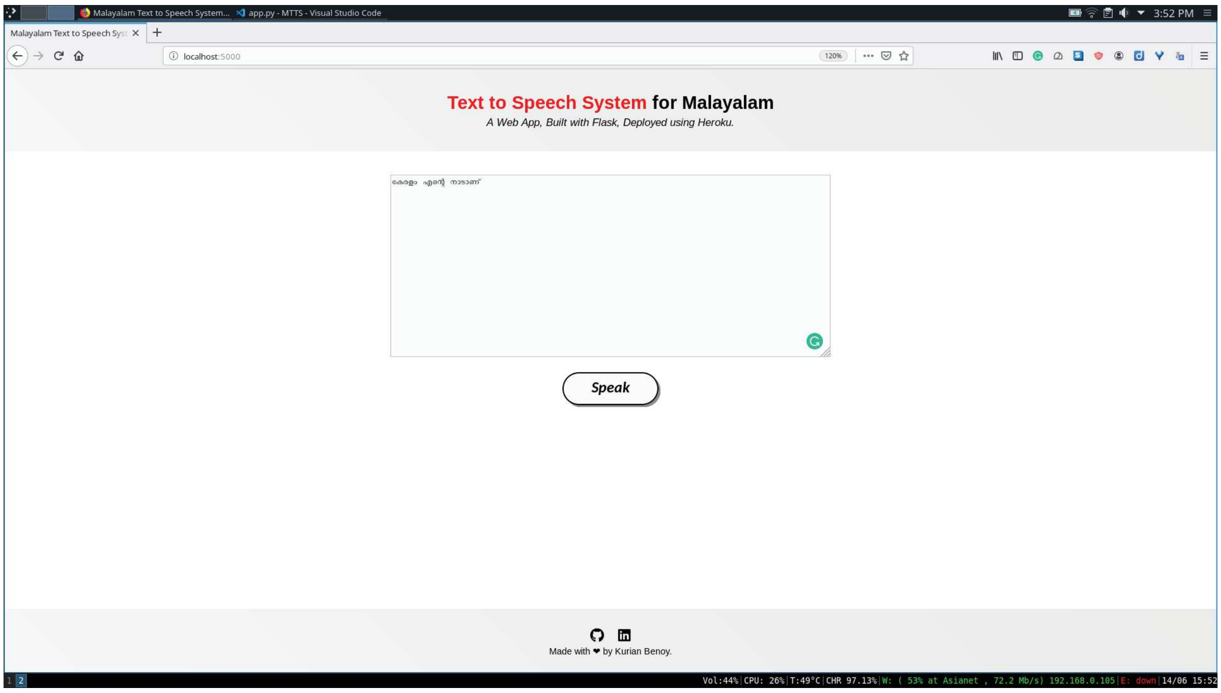
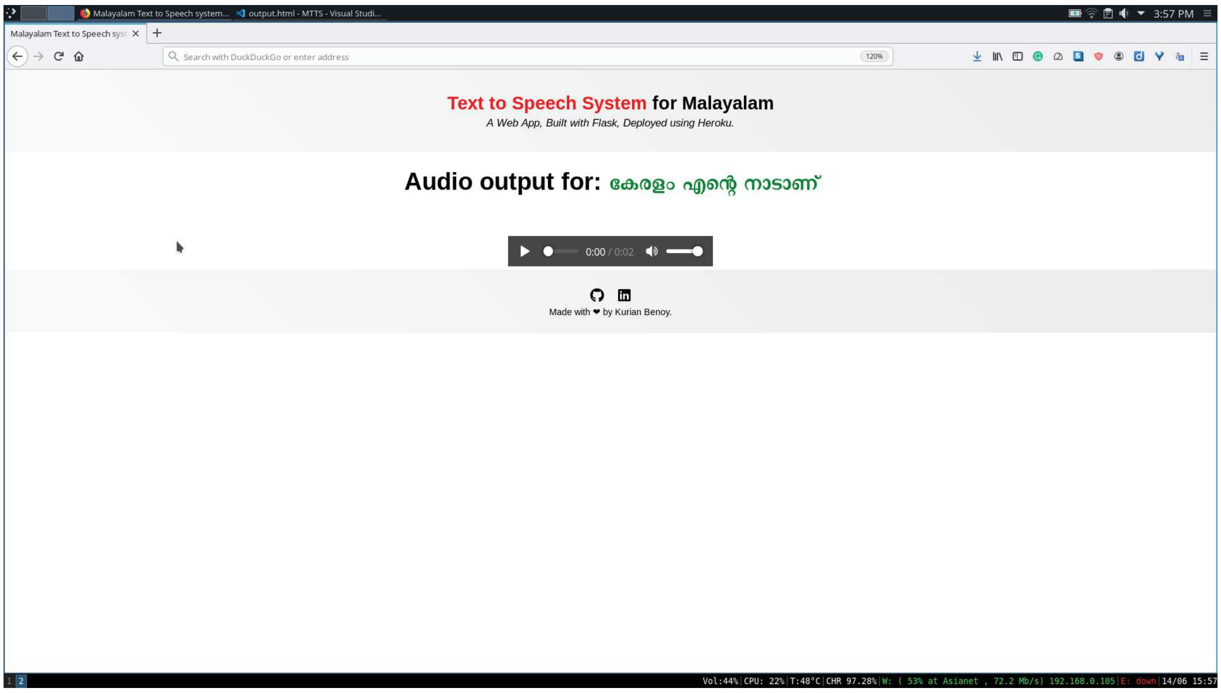

Malayalam Text to Speech System...
app.py - MTTS - Visual Studio Code
3:52 PM
Malayalam Text to Speech Sys
localhost:5000
120%
Text to Speech System for Malayalam
A Web App, Built with Flask, Deployed using Heroku.
കേരളം എന്റെ നാടാണ്
Speak
Made with ❤ by Kurian Benoy.
Vol:44% CPU: 26% T:49°C CHR 97.13% W: (53% at Asianet , 72.2 Mb/s) 192.168.0.105 E: down 14/06 15:52

Figure 8.3: Home page/ User input page

The user can input the malayalam text 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 [25], and change input type language from English to Malayalam in respective operating system.

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



Malayalam Text to Speech system... output.html - MTTS - Visual Studi...

Search with DuckDuckGo or enter address

Text to Speech System for Malayalam

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

Audio output for: കേരളം എന്റെ നാടാണ്

0:00 / 0:02

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

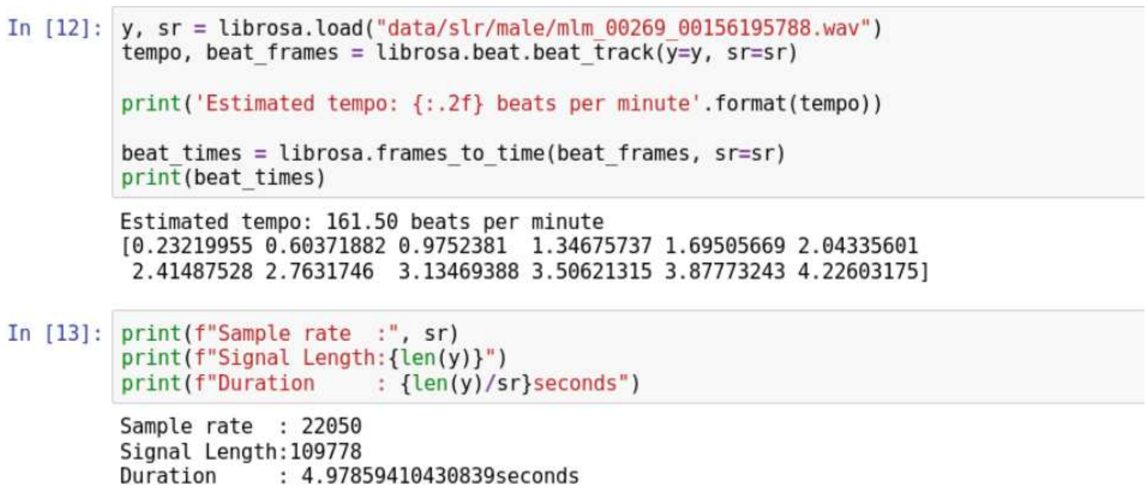
Figure 8.4: Audio output generation page

Chapter 9

Results

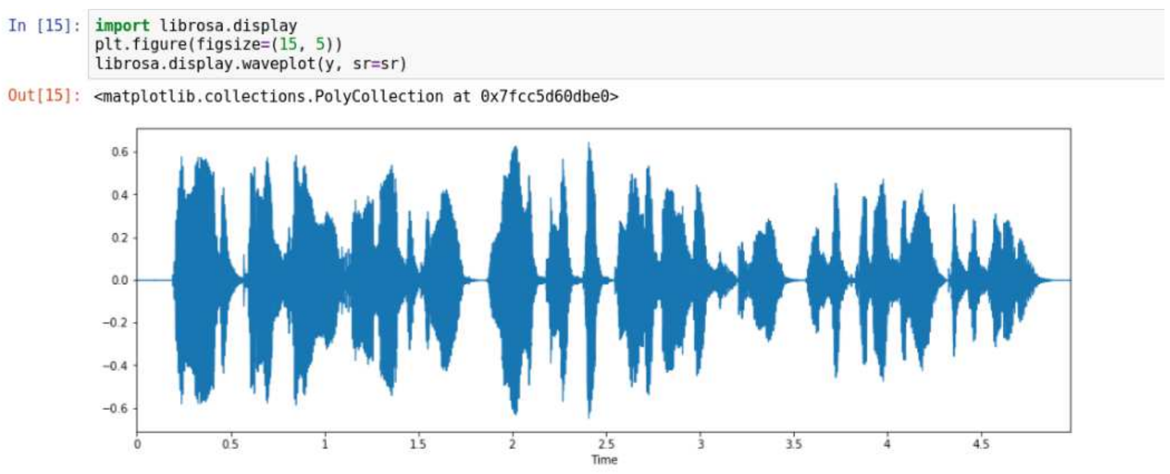
9.1 Exploratory Data Analysis

One of the first tasks for our project was the creation of a exploratory data analysis of the collected dataset. We did a exploratory data analysis on the OpenSLR data[23] of malayalam voices amounting to about 6 hours and Malayalam Speech corpus for about 1 hours[23]. The analysis on Malayalam Speech corpus was contributed to Swathanthra malayalam computings repository as a pull request.



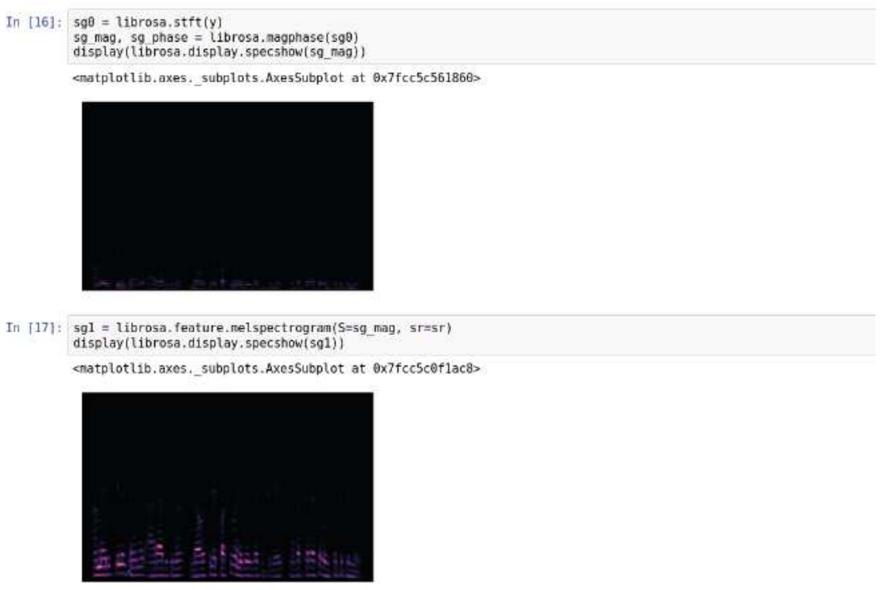
```
In [12]: y, sr = librosa.load("data/slr/male/mlm_00269_00156195788.wav")
         tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr)

         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
```

Figure 9.1: SampleRate, Signal length, duration



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

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

Figure 9.2: Displaying Waveform



```
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>
```

Figure 9.3: Mel-spectograms

```
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')
```

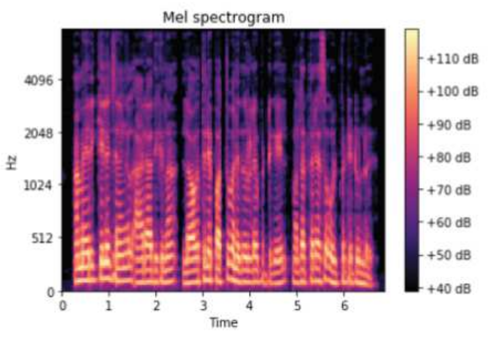
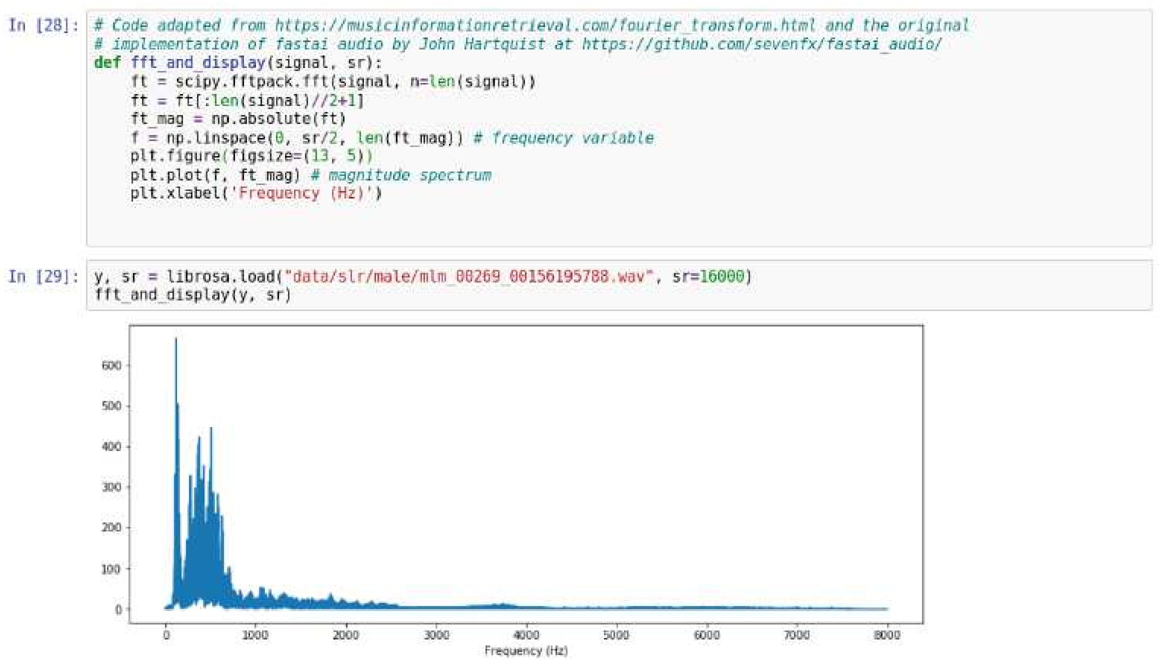
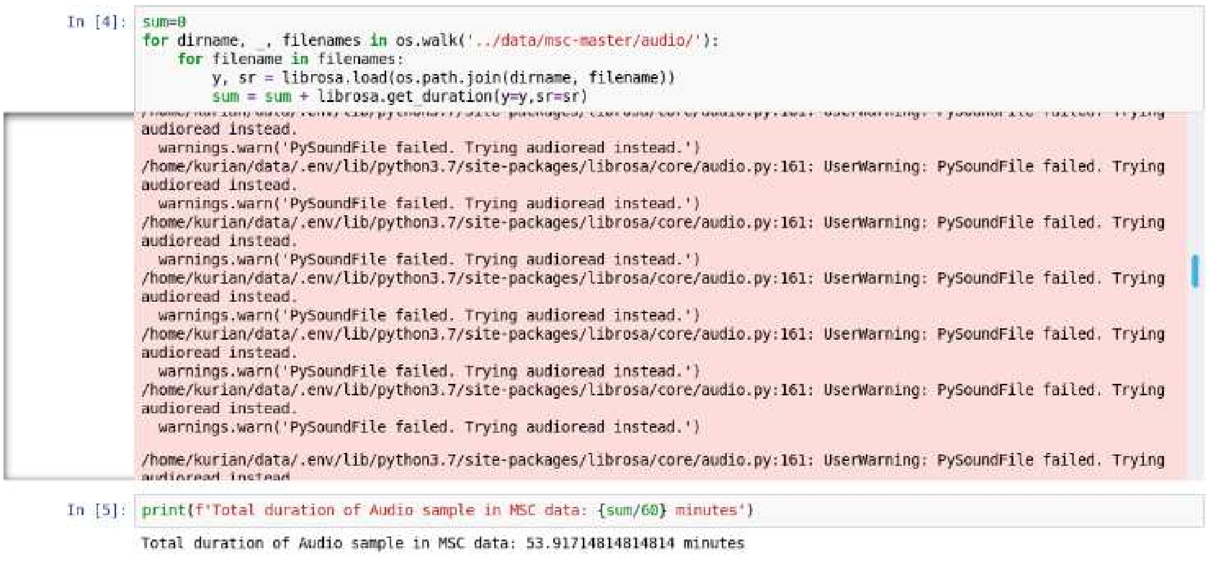


Figure 9.4: Mel-spectogram vs Amplitude Plot



```
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)
```

Figure 9.5: Fourier Transform



```
In [4]: sum=0
        for dirname, _, filenames in os.walk('../data/msc-master/audio/'):
            for filename in filenames:
                y, sr = librosa.load(os.path.join(dirname, filename))
                sum = sum + librosa.get_duration(y=y,sr=sr)

        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')
        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying
        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')
        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying
        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')
        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying
        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')
        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying
        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')
        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying
        audioread instead.
          warnings.warn('PySoundFile failed. Trying audioread instead.')

        /home/kurian/data/.env/lib/python3.7/site-packages/librosa/core/audio.py:161: UserWarning: PySoundFile failed. Trying

In [5]: print(f'Total duration of Audio sample in MSC data: {sum/60} minutes')

        Total duration of Audio sample in MSC data: 53.91714814814814 minutes
```

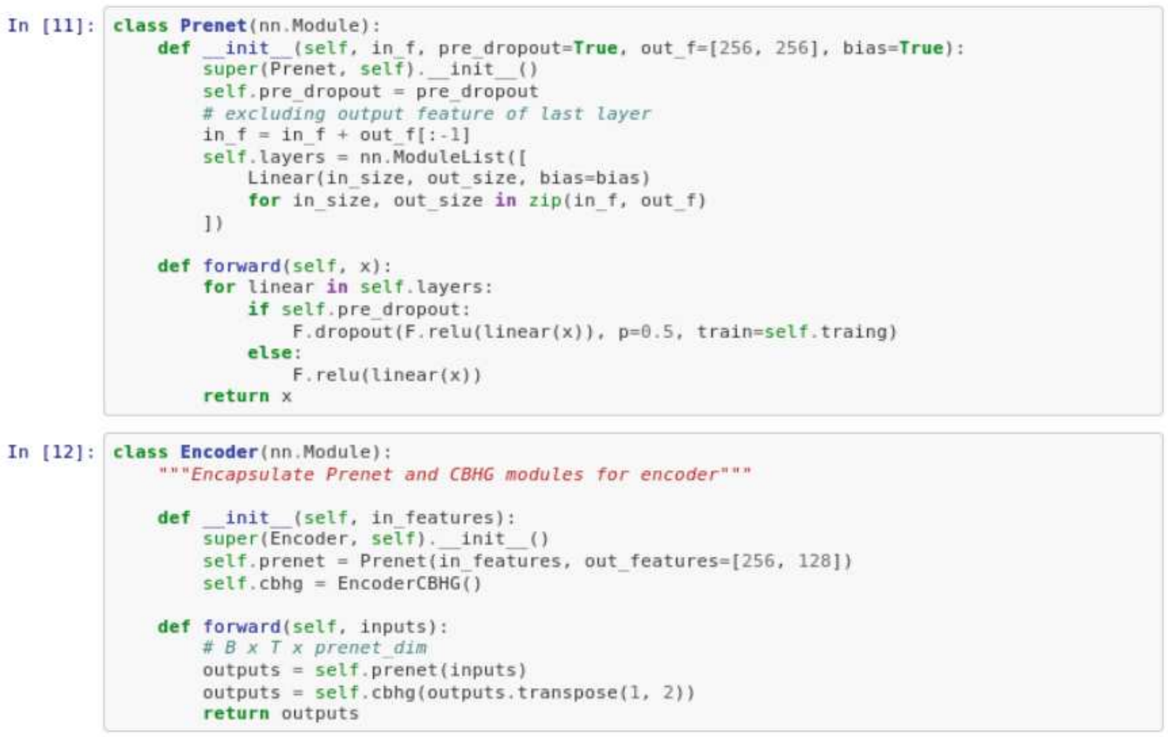
Figure 9.6: Total duration of MSC dataset

9.2 Implementing Tactron architecture

At first we attempted to build a text to speech system by implementing Tactron architecture and Tacron2 architecture. We successfully implemented Tactron architecture based on the paper which can be build from ¡text, audio¿ speech pairs.

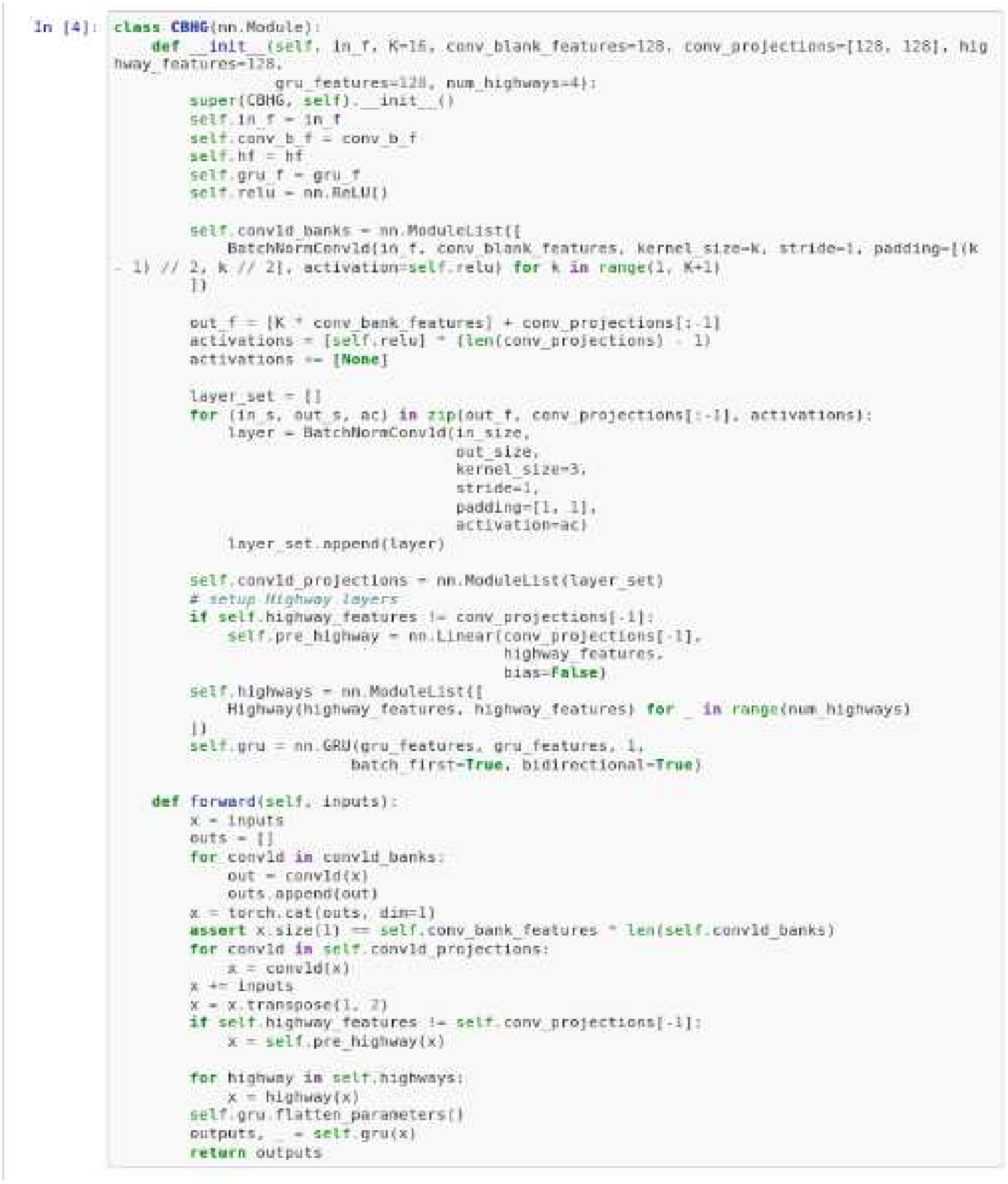
The components of Tactron architecture are:

- CBHG Module
- Encoder
- Decoder



```
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
```

Figure 9.7: Encoder module



```
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
```

Figure 9.8: CBHG module

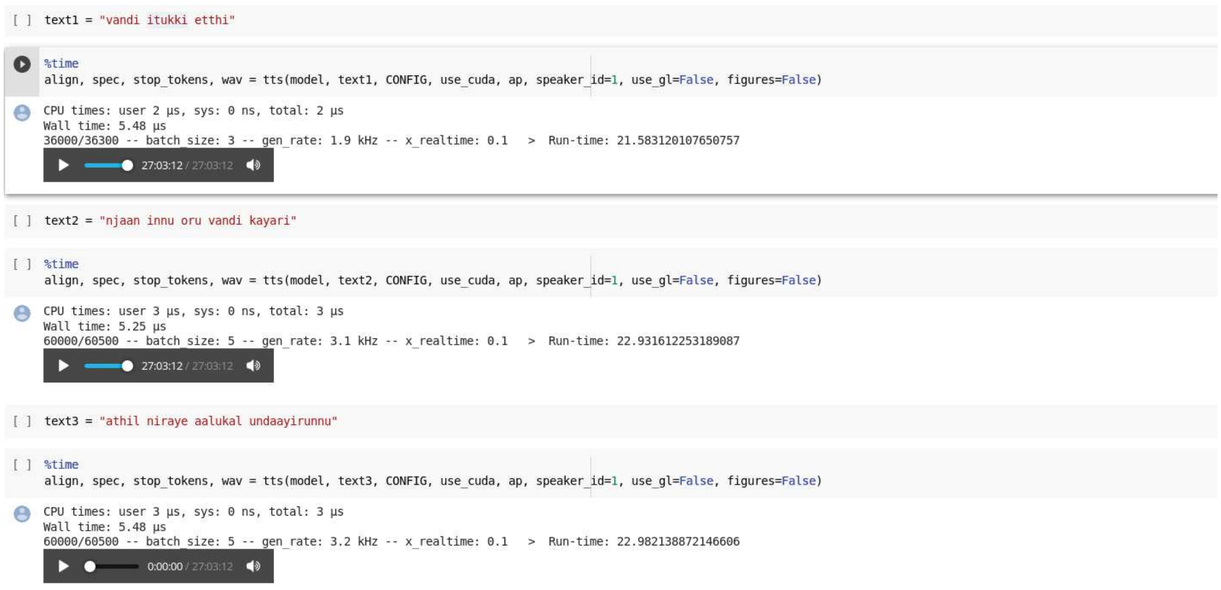
9.3 Implementing Tactron2 architecture

Tactron2, a neural network architecture for speech synthesis directly from text. The system consists of recurrent sequence to sequence feature extraction networks that map character embeddings to mel-scale spectrogram, which is currently one of the best Text to speech architectures, followed by modified Wavenet model acting as a synthesis domain. Model achieves 4.53 MOS score. Seq-2-seq models made of RNNs act as backbone networks for this.

Code can be found in the github repo. Further attempts to take this architecture forward and make a Text to speech synthesis system faced issues because festvox one of the software dependencies needed for the project, was not available for malayalam.

9.4 Transliteration of Text

Here the malayalam input text is being passed into a library to transliterate malayalam into english. We used LibIndic library developed by Indicproject 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
```

Figure 9.9: Output of transliteration of text

9.5 Espeak-ng approach

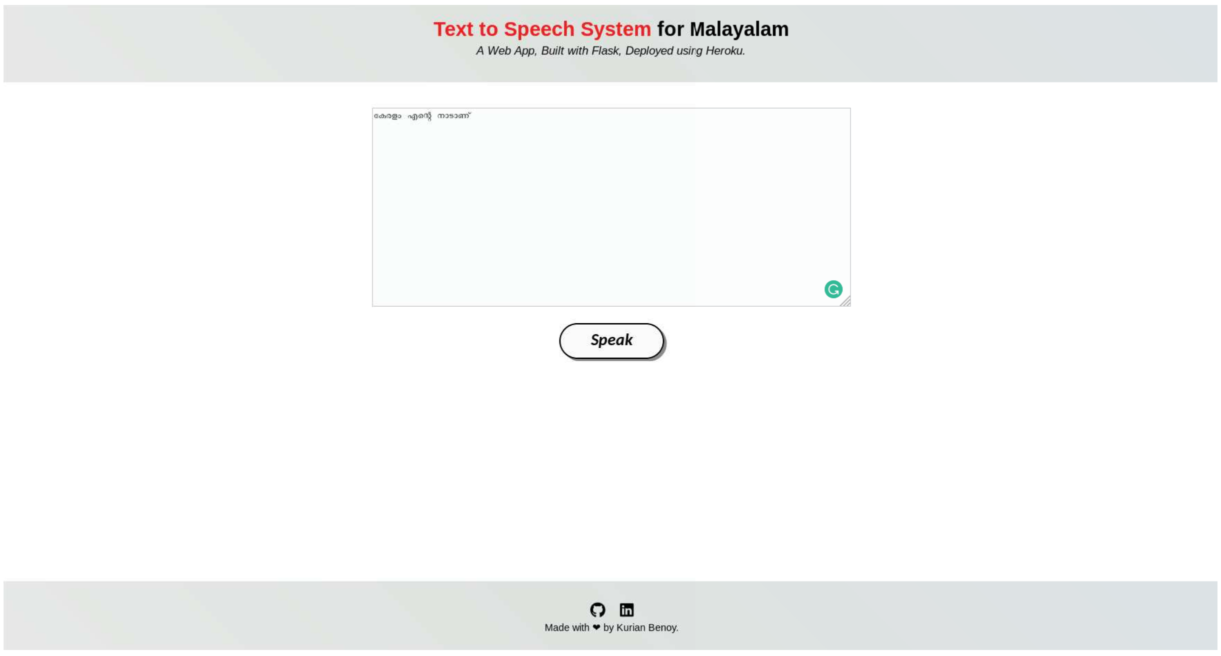
The eSpeak NG is a compact open source software text-to-speech synthesizer for Linux, Windows, Android and other operating systems. It supports 107 languages and accents. It is based on the eSpeak engine created by Jonathan Duddington. eSpeak NG uses a ”formant synthesis” method.

9.6 Flite approach

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. Flite by default doesn’t support languages like Malayalam. Instead we used a Indic voice pre-trained with Kannada, a dravidian language and found that the results were quite good enough and quite audible, compared to exsisting approaches.

The voices samples of each approach can be found in the project repository in github[25].

9.7 User Interface

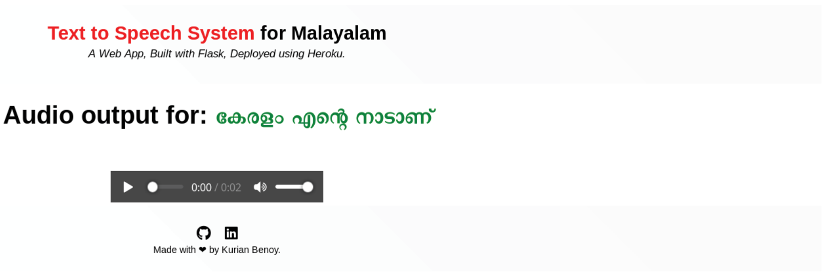

Text to Speech System for Malayalam

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

കേരളം എന്റെ നാടാണ്

Made with ❤ by Kurian Benoy.

Figure 9.10: Home page to input text



Text to Speech System for Malayalam

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

Audio output for: കേരളം എന്റെ നാടാണ്

Figure 9.11: Result page which speaks the sound and plays it

Chapter 10

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.

Chapter 11

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 analayser, 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.

Chapter 12

Publication

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

A STUDY OF TEXT TO SPEECH SYSTEMS FOR NON-ENGLISH LANGUAGES

[1]Kurian Benoy, [2]Jiby J Puthiyidam

[1 2] Govt. Model Engineering College, Ernakulam, Kerala

Abstract: 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 educational applications as well. Most of the text-to-speech systems are currently made for English language. The objective of this paper is to provide an overview of existing Text-To-Speech synthesis techniques and to provide a comprehensive survey on text-to-speech systems for non-English languages having lesser resources.

Index Terms **– Text-to-Speech, Synthesis techniques, Deep Learning**

I. INTRODUCTION

A text-to-speech (TTS) system [1] converts normal language text into speech. The ultimate goal is to create a natural sound from normal text. The current trend in TTS research calls for systems that enable production of speech in different styles with different speaker characteristics and even emotions. Speech synthesis generally refers to the artificial generation of human voice - either in the form of speech or in other forms like songs etc. The computer system used for speech synthesis is known as speech synthesizer.
A text-to-speech-system is composed of two parts:

- The front-end consists of converting a text into graphemes after text normalization, pre-processing, or tokenisation
- The back-end, referred to as the synthesizer, which converts the symbolic linguistic representation into sound.
  We can add pitch, prosody etc. to target speech as required.

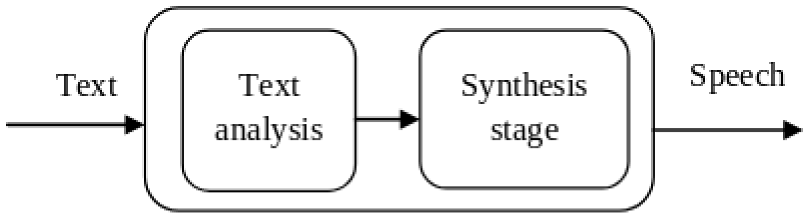


Figure 1 - Phases of Text to Speech System [21]

First, we mention some of the historical developments of Speech synthesis which paved the way for development of contemporary text-to-speech systems. The following contents are extracted from [1] which did a comprehensive historical study of how present system's work.

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 for 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 of producing 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 affected 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.

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

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 HÃ¶gskolan (KTH), Swede. (as mentioned in [1])

For a good TTS it's necessary to have text normalization. Text normalization is the process of converting non-standard words (NSWs) such as numbers, and abbreviations into standard words so that their pronunciations can be derived by a typical means (usually lexicon lookups). Text normalization is, thus, an important component of any text-to-speech (TTS) system. Without text normalization, the resulting voice may sound unintelligent.

This paper is structured as following. First, we talk about the prominent methods of Text to Speech in this era and this section captures most of the recent advances in TTS systems which came due to the advent of Deep Learning. This will help any practitioners to understand the various techniques in this field. As our research interests lies in TTS systems in Non-English languages, we have focused on selecting the most appropriate methods of TTS for such languages. We have mentioned some of the prominent TTS methods implemented in low resource Non-English languages in the related works section. Finally, we talk about the experimental results of selected TTS papers and our future plans.

II. Methods of Text-to-Speech

In this section we discuss some of the prominent methods of Text-to-Speech generation in the contemporary world. We have included some of the most popular and innovative techniques which came in the near future as well here.

A. Concatenative Synthesis

Synthesis of unit selection allows use of broad recorded speech repositories. Each documented utterance is segmented into some or all of the following during the compilation of a database: individual phones, diphones, half-phones, syllables, morphemes, verbs, phrases, and sentences. Typically, division into segments is done using a specially modified speech detector set to a forced alignment mode with some manual configuration. So concatenative speech synthesis method involves the production of artificial human like speech from pre-recorded units of speech by phonemes, diphones, syllabus, words or sentences.[6]

B. Diphone synthesis

Diphone synthesis uses a minimal speech database containing all the diphones (sound-to-sound transitions/combinations) that occur in a language. The number of diphones depends on the phonotactics of the language: for example, there are about 800 diphones in Spanish and about 2500 in German. As only single instances of speech are available in speech database, in order to obtain good quality of synthesised speech with prosody and naturalness, speech processing techniques are applied. PSOLA, TD-PSOLA, LP-PSOLA, ESNOLA, FD-SOLA are some of the common techniques used for obtaining good quality synthesised speech.[6]

C. Formant synthesis

Formant Synthesis does not use human speech samples during run time. Instead, a synthesized speech output is generated using an additive synthesis and an acoustic model (physical modeling synthesis). Parameters such as fundamental frequency, voice and noise levels vary over time are used to create a waveform of artificial speech. This method is sometimes referred to as a rule-based synthesis; however, many concatenative systems also have a rule-based component. Many systems based on formative synthesis technology generate artificial, robotic-sounding speech that would never be mistaken for human speech. Formative synthesis systems have advantages over concatenative systems. Formant synthesized speech can be accurately intelligible, even at very high speeds, eliminating acoustic errors that often affect concatenative systems. High-speed, visually impaired speech is used to easily access devices using a screen reader. Formant synthesizers are usually smaller than Concatenative speech synthesis. [7]

D. HMM based synthesis

Hidden Markov Model (HMM) based synthesis is a synthesis process based on secret Markov models, and is also known as the statistical parametric synthesis. In this method, the frequency spectrum (vocal tract), the fundamental frequency (voice source), the length of the source (prosody) of speech are simultaneously modeled by HMMs. Speech waveforms are generated from HMMs themselves on the basis of the maximum probability criterion.[12]

E. Articulatory synthesis

Articulatory synthesis refers to computational techniques for speech synthesis based on models of the human vocal tract and the articulation processes that occur there. The first articulatory synthesizer to be used routinely for laboratory experiments was created at Haskins Laboratories in the mid-1970s by Philip Rubin, Tom Baer and Paul Mermelstein. This synthesizer, known as ASY, was based on the vocal tract models created by Paul Mermelstein, Cecil Coker, and colleagues at Bell Laboratories in the 1960s and 1970s. Articulatory speech synthesis refers to make speech based on models of human vocal tract and articulation process.[8]

Articulation synthesis comprises of:

1) Generation of vocal tract movements
2) Convert movement information into continuous succession of vocal tract geometries
3) Generate acoustic signals from articulatory information

F. Hybrid Text to Speech synthesis

The Hybrid TTS approach is a combination of the two main approaches of synthesis: Concatenative synthesis and Statistical Synthesis. The hybrid TTS combines the characteristics of smooth transitions between adjacent speech segments of a Statistical TTS with the naturalness of a Concatenative TTS. This is achieved by interweaving natural speech segments and statistically generated speech segments. The statistical segments are positioned so as to smooth discontinuities in the synthesized speech, while enabling as far as possible natural speech sequences as they appear in the training inventory disadvantages of this system are the degradation in speech quality when TTS speech inventory is small and more signal processing requirement [9].

G. Statistical Parametric Synthesis

Statistical parametric synthesis makes use of averaged acoustic inventories that are extracted from the speech corpus.The extracted parameters of speech are the spectral parameters such are cepstral coefficients or line spectral pairs, and excitation parameters such as fundamental frequency. Statistical Parametric synthesis has the advantages of requiring less memory to store the parameters of the model, rather than the data itself and it allows more Variation in the speech produced for example, an original voice can be converted into another voice. The most commonly used statistical parametric speech synthesis technique is the Hidden Markov Model (HMM) synthesis and Unit Selection synthesis [10].

F. Efficient speech

A new model for audio synthesis WaveRNN is introduced in [4]. A single layer RNN with a softmax layer WaveRNN matches current SOTA of Wavenet models. Yet it's possible to reduce the N (no of layers) and operations in GPU. A technique of weight pruning is applied to reduce number of weights in WaveRNN. A sparse WaveRNN makes it possible to sample high fidelity audio on a mobile CPU in real time. A new generation scheme based on sub scaling that folds a long sequence into batches of shorter sequences and allows one to generate multiple samples at a time. The subscale WaveRNN produces 10 samples per step without loss of quality and offers an orthogonal method for increasing sampling efficiency.

WaveRNN demonstrates that a simple recurrent neural network for sequential modelling of high fidelity audio, and demonstrated a high performance implementation of this model on GPUs. It's shown that large sparse models have much better quality than small dense models with the same number of parameters and we have written high performance block-sparse matrix-vector product operations to demonstrate that sampling time is proportional to parameter count. It showed that high fidelity audio generation is now achievable on widely available low-power mobile CPUs. Finally, this paper introduced the subscale dependency scheme that lets sequential models generate many samples per step while preserving the output quality of the original model. The underlying ideas of the methods we introduce are not specific to audio, and the results of sparse models have implications for inference in all types of neural networks.

G. Wavenet Based Models

In September 2016, Deep Mind proposed WaveNet, a deep generative model of raw audio waveforms [3]. This shows the community that deep learning-based models have the capability to model raw waveforms and perform well on generating speech from acoustic features like spectrograms or spectrograms in Mel scale, or even from some preprocessed linguistic features. The model is fully probabilistic and self-regressive, with the predictive distribution of each audio sample conditioned on all previous ones. Data with tens of thousands of samples per second of audio can be efficiently trained. Applied to text-to-speech, it produces state-of - the-art performance, with human listeners rating it as significantly more natural than the best sound parametric and concatenative systems can produce for both English and Mandarin. Wavenet model triggered a huge development of Text to speech architectures based on Deep Learning which is mentioned in [11].

H. Fast Speech

Compared with traditional concatenative and statistical parametric approaches, neural network based end- to-end models suffer from slow inference speed, and the synthesized speech is usually not robust (i.e., some words are skipped or repeated) and lack of controllability (voice speed or prosody control). In this work [5], a novel feed-forward network based on Transformer is used to generate Mel-spectrogram in parallel for TTS. Specifically attention alignments are extracted from an encoder-decoder based teacher model for phoneme duration prediction, which is used by a length regulator to expand the source phoneme sequence to match the length of the targeted Mel-spectrogram sequence for parallel Mel-spectrogram generation. Experiments on the LJ-Speech dataset show that our parallel model matches autoregressive models in terms of speech quality, nearly eliminates the problem of word skipping and repeating in particularly hard cases, and can adjust voice speed smoothly. Most importantly, compared with autoregressive Transformer TTS, our model speeds up Mel-spectrogram generation by 270x and the end-to-end speech synthesis by 38x. Therefore, this method is called Fast speech.

Concatenative Synthesis
Diphone Synthesis
Formant Synthesis
HMM based Synthesis
Articulatory Synthesis
Hybrid Text to Speech Synthesis
Statistical Parametric Synthesis
Efficient Speech
Wavenet Based Models
Fast Speech

Table 1- Popular TTS Techniques

III. Related Works

A. Text Normalization for Bangla, Khmer, Nepali, Javanese, Sinhala and Sudanese TTS [13]

The main contributions of this [13] paper are as follows:

1. Describe a general method for working with native speakers in identifying patterns and grammars needed to normalize text
2. Making available text normalization grammars and their test cases for a wide range of common semiotic classes for 6 low resourced languages
3. Provide a recipe for utilizing these grammars and for integrating them into actual text-to-speech systems.

Text normalization takes plain text as input for any language. It takes input words being separated each other using whitespaces while for languages which do not use whitespaces to separate works - like Khmer, the input is the output after passing text into word segmenter. Text normalization is divided into two phases. First, input text is analyzed and NSWs are classified into semiotic classes. In this phase, some input tokens may be grouped together. For example, input text "15 km" may be grouped together and are classified as a measurement token. Then, verbalizer grammar for each semiotic class will convert the classified NSWs into standard text accordingly

After identifying the semiotic classes, they reached out to Native speakers in each language with a questionnaire. The questionnaire contains a set of questions for each semiotic class. The questions were designed to capture the writing and verbalizing system of the language. They took into consideration of various language-specific characteristics that affect text normalization. For each language there are particular characteristics for the same. In case of Bangla: different inflection cases are handled and 4 different time indicators similar to "am/pm". Besides the above considerations, all languages in this set, other than Javanese and Sudanese, have their own alphabets and digits. Bangla uses the Bengali alphabet. Khmer uses the Khmer script. Nepali uses Devanagari and Sinhala uses the Sinhala alphabet. Khmer writing contains inconsistent usages of the zero-width space character. Bengali, Nepali and Sinhala also utilize zero-width nonjoiner (U+200C). Various unit test cases was created for the grammars so as to solicit both common and corner cases. So our test cases verify if everything works fine.

On obtaining test cases and explanations about how to classify and verbalize different semiotic classes, create grammar rules to do the classification and verbalization. The grammars that created is Thrax grammars, which consist of mostly regular expressions and context-dependent rewrite rules. The grammars can then be compiled with the Thrax grammar compiler, which turns them into archives of finite state transducers. First, the input text will be classified into different semiotic classes. The output of this classification step is a protocol buffer which contains information about the original token and its semiotic class. Then, the protocol buffer will be passed to the verbalizer component, which converts the token into one or more standard words based on the verbalization rules for the semiotic class associated with the token. For example, the classification of input”100 m” will output the following protocol buffer: measure decimal part:”100” units:”meter”

The usage of text normalization with open source tools by Google is mentioned properly in this paper [13]. Moreover tools are made open source and presented text normalization grammars, both classifiers and verbalizers, for six low resource languages and along with test cases are available free to public. The process of testing and utilizing this grammar is mentioned in Sparrow hawk text normalization system. TTS build using this techniques can be widely adopted for low-resource languages and process is universally applicable in any language. These text normalization techniques helped Google in successfully creating the Project Unison, the first text to speech system in Bangla [19]. More details on the experiences have been mentioned in detail in [20] and about how to tackle the issue of solving Text-to-speech systems for low resource languages.

B. Corpus Driven Malayalam Text-to-Speech Synthesis for Interactive Voice Response System [14]

In paper [14] written by Arun Soman, etc., a corpus-driven Malayalam text-to-speech (TTS) system based on the concatenative synthesis approach is explained. The most important qualities of a synthesized speech are naturalness and intelligibility. In this system, words and syllables are used as the basic units for synthesis. The corpus consists of speech waveforms that are collected for most frequently used words in different domains. The speaker is selected through subjective and objective evaluation of natural and synthesized waveform. The proposed Malayalam text-to-speech system is implemented in Java multimedia framework (JMF) and runs on both in Windows and in Linux platforms. The proposed system provides utility to save the synthesized output. The output generated by the proposed Malayalam text-to-speech synthesis system resembles natural human voice. Text to speech reader software converts a Malayalam text to speech wav file that has high rates of intelligibility and comprehensibility.

The corpus was collected by a single best speaker wave form. The best speaker means the speech produced by that speaker have capabilities with respect to energy profile, speaking rate, pronunciation and into nation. Input Malayalam text was first text normalized which made the input text words in non- standard form like Numbers, dates, etc. and punctuations were also removed. Then process of sentence splitting took place in the paragraphs and words were separated out. Romanization is the representation of written word with a roman alphabet. In this system Romanized form of Malayalam words/syllables are generated. For representing the written text the method used for Romanization is transliteration and for spoken word, the method is transcription. The final stage is the concatenation process.

All the arranged speech units are concatenated using a concatenation algorithm. The concatenation of speech files is done in java media framework. The main problem in concatenation process is that there will be glitches in the joint. The concatenation process combine all the speech file which is given as a output of the unit selection process and then making in to a single speech file. This can be played and stopped any where needed.

C. Orthography with Maratha [15]

Speech synthesis models are typically built from a corpus of speech that has accurate transcriptions. Paper [15] consider many of the languages of the world do not have a standardized writing system. This paper is an initial attempt at building synthetic voices for such languages. It may seem useless to develop a text-to-speech system when there is no text available. A novel method to build synthetic voices from only speech data is shown here. Experimental results and oracle studies shows that it is possible to automatically devise an artificial writing system for these languages, and build synthetic voices that are understandable and usable.

The speech data in a target language with no well-defined orthography for transcriptions is given. A simple method to deal with this situation is to run an automatic speech recognizer over available speech data and use its output as transcriptions. The caveat with using a speech recognizer is that because our target language does not have a text form, a speech recognizer will not exist in that language. We hence have to use a speech recognizer in another language: a language that has an orthography, and large corpora to train speech recognizers. This presents another caveat: that is recognizing in a different language than the models are trained for. Using the default language model is thus not ideal, and we need to adapt it so that it is suitable for our target language. It also uses phonetic decoding instead of word level decoding.

The paper [15] solution proposes the following:

1) Choose an appropriate acoustic model for speech recognition
2) Choose a language that has an orthography and is phonetically close to our target language, and then build a phonetic language model on text in the language.
3) Run phonetic decoder on our target speech data with these two models and obtain transcripts
4) Build voice using the speech data and the phonetic transcripts just obtained.

The paper has addressed a novel problem of building speech synthesizers for languages without an orthography. In the solution proposed automatically developing a writing system for the language, using a speech recognition system. The iterative method to build targeted acoustic models yield very good improvements in synthesis quality. The objective and subjective results, as well as oracle results on Marathi, which show that direction to building synthesis models without written text is promising. We also showed similar results on Hindi and Telugu, thus showing that our methods are language Independent.

D. Maithili Text to Speech system [16]

This paper describes the method of creating a text-to-speech system for Maithili, which is a similar variant of Hindi. As most Indian languages, Maithili is syllabic in nature and concatenative method is used for purpose of speech generation taking speech syllable as the basic unit. The concatenate Unit Selection Synthesis (USS) technique is used. Naturalness of USS for small amount of data is better compared to other methods.

The workflow of this TTS made by Amit Kumar Jha, etc is:

1) Input Maithili text written in Devanigiri script using UTF-16
2) Inputted text is normalized with the help of three
3) Inputted text is segmented into sentence level. Afterwards, it is segmented into word level using white space.
4) A word level search is done in database and if it is found then corresponding speech file is added into playlist. Else, the word is broken into corresponding syllables and corresponding syllables files are searched and added in playlist.
5) Found speech units are concatenated in playlist using digital signal processing.
6) Add prosodic features to the speak file according to the types of sentence.
7) Play the sound of playlist.

The process of text normalization for Maithili language is mentioned in detail in [16]. To increase the naturalness of TTS system, 1055 most frequently occurring word have been recorded and stored in separate lexicon. The system support UTF-16 for text input and a C#.NET interface is used for developing TTS system in Maithili. The speech database consists of 930 syllable (C*V) in total. Each position has 300 syllables and 10 independent vowels. Each position has 300 syllables and 10 independent vowels. 930 units of speech data is build all three positions. This is the first TTS system which exists for Maithili language till date

E. Speaking Style Adaption in Text-to-Speech Synthesis using sequence-to-sequence models with attention [17]

Neural networks which are data driven is good in Text to Speech synthesis. The paper [17] is aimed for challenging speaking styles like Lombard speech (speaking in loud voice) where it's difficult to generate large corpora. A new transfer learning method which adopt Lombard style from Normal speaking style. They use a learning method to adopt sequence to sequence based TTS system of normal speaking style to Lombard style. Moreover on evaluation results indicated that an adaption system with Wavenet Vocoder clearly outperformed conventional deep neural networks based on TTS.

Here TTS system uses a sequence to sequence model with attention. The model accepts either mono-phonemes or graphemes as inputs and emits acoustic parameters as outputs. It consists of three main components: 1) Encoder 2) attention and 3) decoder. The encoder takes text sequence x of length L as input, which represented either in the character or phoneme domain as one-hot vectors. The encoder learns a continuous sequential representation h using various neural network architectures such as LSTMs or CNN. At each output time step t, both the attention and decoder modules work together. The decoder takes the previous hidden state and current context vector as inputs and generates the current output. The process runs until the end of the utterance is reached. The accuracy of recognizing Lombard speech by our TTS is 95%.

F. Indonesian TTS using Diphone based speech TTS [18]

Paper [18] shows a novel approach to create a text to speech system for Indonesian language. This approach first creates a database of Indonesian diphone at first. Indonesian diphone synthesis uses speech segment of recorded voice from text to speech and save it as audio file like WAV, MP3. First a diphone databases is created and then convert text to speech from input of numbers, words. They used diphone concatenative synthesis in which recorded segments were collected.

The two main contributions of this paper are: First developed a diphone database including creating a list of words consisting of diphones organized by prioritizing diphones in this system. Second develop system using Microsoft visual Delphi 6.0, includes: the conversion system from the input of numbers, acronyms, words, and sentences into representations diphone. There are two kinds of conversion (process) alleged in analyzing the Indonesian text-to-speech system. One is to convert the text to be sounded to phoneme and the other convert the phoneme to speech. Method used in this research is called Diphone Concatenative synthesis, in which recorded sound segments are collected. Every segment consists of a diphone (2 phonemes). This synthesizer may produce voice with high level of naturalness. The Indonesian Text to Speech system can differentiate special phonemes like in 'Beda' and 'Bedak' but sample of other specific words is necessary to put into the system. This Indonesia TTS system can handle texts with abbreviation, there is the facility to add such words. Indonesian Text-To-Speech System using diphone concatenative synthesis can produce speech or language the natural approach. Speech or resulted sound may not be perfect for several causes such as poor recording and distorted phoneme segmentation process where front, middle and end borders are not synchronized.

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

IV. CONCLUSION

On studying the various papers we realized that the process of synthesizing speech from text has evolved rapidly over the years. Studying TTS on various non-English languages it is realized the accuracy and naturalness of TTS systems in these languages are not as good as English TTS. After a detailed study of TTS on various low resourced non-English languages, we have concluded the following:

The paper [14] which worked on concatenative synthesis was used for one of the earliest text to speech system for Malayalam. The experiments in the paper give good MOS score for small and easy sentences (MOS score of 4.0) while for harder sentences the accuracy is giving an MOS score of 2.72. In [15] Text-to speech for languages like Martha based on techniques mentioned in paper[3] on evaluation using MCD score gives considerably good result. The tests were conducted by taking speech data of Telugu and Hindi as well with the assumption that both languages have no orthography and no transitions were provided to them and the accuracy is considerably good. The iterative models mentioned in [15] to build targeted models yield very good improvements in synthesis quality. In case of Maithili Text-to-speech [16], evaluation of TTS depends on three key approaches: 1. User Testing 2. Feature comparisons, and 3.objective measures. The basic approach to measure the speech output was decided by talking people from ten different backgrounds, and calculating Mean opinion score. The output score in average for all the speakers is satisfactorily close to 84% for speech generated by [16]. The intermixing of Hindi and English strings in Maithili language helped to give a better result for this system. In [17] compared different TTS models and vocoders to adapt the speaking style of speech synthesis from normal to Lombard. The study proposes using an adaptation method based on fine-tuning combined with sequence-to-sequence based TTS models and the WaveNet vocoder conditioned using Mel spectrograms. Listening tests show that the proposed method outperformed the previous best method that was developed using a LSTM-RNN based adapted systems. According to results mentioned in paper by Sultarman etc. [18] Indonesian Text- To-Speech System using diphone concatenative synthesis can produce speech or language the natural approach which can differentiate special phonemes like Beda and Bedak and can handle text with abbreviations as well. [13] Which provided a new text normalization system helps in creating better TTS systems for low resource languages in the future.

Based on this survey, we aim to create an accurate and steady Text-to Speech system for Malayalam as a future work. Most of the TTS systems in Malayalam are not so good now as it isn't able to produce speech with decent quality and add prosody features to the sound. There has been no development of TTS in Malayalam for the past 10 years, we hope to make some positive new change in this direction.

REFERENCES

[1] History and Development of Speech Synthesis, Helsinki University of Technology (http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html)

[2] Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine ("Mechanism of the human speech with description of its speaking machine", J. B. Degen, Wien). (in German)

[3] Oord, Aaron & Dieleman, Sander & Zen, Heiga & Simonyan, Karen & Vinyals, Oriol & Graves, Alex & Kalchbrenner, Nal & Senior, Andrew & Kavukcuoglu, Koray. (2016). Wavenet: A generative model for Raw Audio (https://arxiv.org/abs/1609.03499)

[4] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, etc. “Efficient Neural Audio Synthesis”, 25 June 2018

[5] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, Tie-Yan Liu. “FastSpeech: Fast, Robust and Controllable Text to Speech.” 29 May, 2019

[6] Rubeena A. Khan , J. S. Chitode, “Concatenative Speech Synthesis: A Review, International Journal of Computer Applications” (0975 – 8887). Volume 136 – No.3, February 2016.pg-1 to 4.

[7] Sami Lemmetty. “Review of Speech Synthesis Technology”. Helsinki University of Technology, Department of Electrical and Communications Engineering. March 30, 1999.

[8] Pertti Palo. “A Review of Articulatory Speech Synthesis”. Espoo, June 5, 2006

[9] Stas Tiomkin, David Malah, Slava Shechtman, and Zvi Kons, “A hybrid text-to-speech system that combines concatenative and statistical synthesis units” IEEE Transactions on Audio, SPEECH, and Language Processing, vol. 19, no. 5, JULY 2011 pp 1278-1288.

[10]Heiga Zen, Keiichi Tokuda, Alan W. Black ,“Statistical parametric speech synthesis”, Speech Communication vol.51,no.11,2009,pp. 1039–1064.

[11]http://www.erogol.com/text-speech-deep-learning-architectures/

[12]Raitio, Tuomo, et al. "HMM-based speech synthesis utilizing glottal inverse filtering." Audio, Speech, and Language Processing, IEEE Transactions on vol.19, no.1, 2011, pp. 153-165.

[13]Keshan Sodimana, Pasindu De Silva , Richard Sproat, Theeraphol Wattanavekin , Chenfang Li,Alexander Gutkin, Supheakmungkol Sarin, Knot Pipatsrisawat, “Text Normalization for Bangla, Khmer, Nepali, Javanese and Sudanese, Text-to-speech systems”. The 6th Intl. Workshop on Spoken Language Technologies for Under-resourced Languages

[14]Arun Soman, Sachin Kumar S., Hemanth V. K., M. Sabarimalai Manikandan, K. P.Soman. “Corpus Driven Malayalam Text-to-Speech Synthesis for Interactive Voice Response System”. International Journal of Computer Applications (0975 – 8887)Volume 29– No.4, September 2011.

[15]Sukhada Palkar, Alan W Black, Alok Parlikar.”Text-To-Speech for Languages without an Orthography”

[16]Maithili Text-to-Speech System, Amit Kumar Jha, Piyush Pratap Singh, Pankaj Dwivedi

[17]Bajibabu Bollepalli, Lauri Juvella, Paavo Alku. “SPEAKING STYLE ADAPTATION IN TEXT-TO-SPEECH SYNTHESIS USING SEQUENCE-TO-SEQUENCE MODELS WITH ATTENTION”

[18]Sultarman. “Indonesian Text-to-speech system using Diphone concatenative synthesis”

[19]“Text-to-Speech for Low Resource Languages: But can it say Google? “(https://ai.googleblog.com/2016/02/text-to-speech-for-low-resource.html)

[20]One Down, 299 to go (https://ai.googleblog.com/2018/09/text-to-speech-for-low-resource.html)

[21]Dessai Sidhi, etc Survey on Various Text to speech synthesis

References

[1] History and Development of Speech Synthesis, Helsinki University of Technology (http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html)

[2] Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine ("Mechanism of the human speech with description of its speaking machine", J. B. Degen, Wien). (in German)

[3] Oord, Aaron and Dieleman, Sander and Zen, Heiga and Simonyan, Karen & Vinyals, Oriol & Graves, Alex & Kalchbrenner, Nal and Senior, Andrew & Kavukcuoglu, Koray. (2016). Wavenet: A generative model for Raw Audio (https://arxiv.org/abs/1609.03499)

[4] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, etc. "Efficient Neural Audio Synthesis", 25 June 2018

[5] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, Tie-Yan Liu. "FastSpeech: Fast, Robust and Controllable Text to Speech." 29 May, 2019

[6] Rubeena A. Khan , J. S. Chitode, "Concatenative Speech Synthesis: A Review, International Journal of Computer Applications" (0975 – 8887). Volume 136 – No.3, February 2016.pg-1 to 4.

[7] Sami Lemmetty. "Review of Speech Synthesis Technology". Helsinki University of Technology, Department of Electrical and Communications Engineering. March 30, 1999.

[8] Pertti Palo. "A Review of Articulatory Speech Synthesis". Espoo, June 5, 2006

[9] Stas Tiomkin, David Malah, Slava Shechtman, and Zvi Kons, "A hybrid text-to-speech system that combines concatenative and statistical synthesis units" IEEE Transactions on Audio, SPEECH, and Language Processing, vol. 19, no. 5, JULY 2011 pp 1278-1288. Heiga Zen, Keiichi Tokuda, Alan W. Black ,"Statistical parametric speech synthesis", Speech Communication vol.51,no.11,2009,pp.

[10] Heiga Zen, Keiichi Tokuda, Alan W. Black ,"Statistical parametric speech synthesis", Speech Communication vol.51,no.11,2009,pp. 1039–1064.

[11] http://www.erogol.com/text-speech-deep-learning-architectures/

[12] Raitio, Tuomo, et al. "HMM-based speech synthesis utilizing glottal inverse filtering." Audio, Speech, and Language Processing, IEEE Transactions on vol.19, no.1, 2011, pp. 153-165.

[13] Keshan Sodimana, Pasindu De Silva , Richard Sproat, Theeraphol Wattanavekin , Chen-fang Li,Alexander Gutkin, Supheakmungkol Sarin, Knot Pipatsrisawat, “Text Normalization for Bangla, Khmer, Nepali, Javanese and Sudanese, Text-to-speech systems”. The 6th Intl. Workshop on Spoken Language Technologies for Under-resourced Languages

[14] Arun Soman, Sachin Kumar S., Hemanth V. K., M. Sabarimalai Manikandan, K. P.Soman. “Corpus Driven Malayalam Text-to-Speech Synthesis for Interactive Voice Response System”. International Journal of Computer Applications (0975 – 8887)Volume 29– No.4, September 2011.

[15] Sukhada Palkar, Alan W Black, Alok Parlikar.”Text-To-Speech for Languages without an Orthography”

[16] Maithili Text-to-Speech System, Amit Kumar Jha, Piyush Pratap Singh, Pankaj Dwivedi

[17] Bajibabu Bollepalli, Lauri Juvella, Paavo Alku. “SPEAKING STYLE ADAPTATION IN TEXT-TO-SPEECH SYNTHESIS USING SEQUENCE-TO-SEQUENCE MODELS WITH ATTENTION”

[18] Sultarman. “Indonesian Text-to-speech system using Diphone concatenative synthesis”

[19] “Text-to-Speech for Low Resource Languages: But can it say Google? “(https://ai.googleblog.com/2016/02/text-to-speech-for-low- resource.html)

[20] One Down, 299 to go (https://ai.googleblog.com/2018/09/text-to-speech-for-low-resource.html)

[21] Dessai Sidhi, etc Survey on Various Text to speech synthesis

[22] Pytorch Documentation - Getting started (https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html

[23] OpenSLR dataset(https://www.openslr.org/63/)

[24] Malayalam Speech Corpus(https://gitlab.com/smc/msc/)

[25] A feasibility study of Malayalam TTS(https://telegra.ph/A-small-feasibility-study-of-Malayalam-TTS-04-15)

[26] https://github.com/kurianbenoy/MTTS