DEVELOPMENT OF A TEXT TO SPEECH SYSTEM FOR DEVANAGARI KONKANI

A Thesis submitted to Goa University

for the award of the degree of

Doctor of Philosophy

in

Computer Science and Technology

By

Nilesh B. Fal Dessai

GOA UNIVERSITY

Taleigao Plateau

May 2017

Development of a Text to Speech System for Devanagari Konkani

A Thesis submitted to Goa University

for the award of the degree of

Doctor of Philosophy

in

Computer Science and Technology

By

Nilesh B. Fal Dessai

GOA UNIVERSITY

Taleigao Plateau

May 2017

Statement

As required under the ordinance OB-9.9 of Goa University, I, Nilesh B. Fal Dessai, state that the present Thesis entitled, ‘Development of a Text to Speech System for Devanagari Konkani’ is my original contribution carried out under the supervision of Dr. Jyoti D. Pawar, Associate Professor, Department of Computer Science and Technology, Goa University and the same has not been submitted on any previous occasion. To the best of my knowledge, the present study is the first comprehensive work of its kind in the area mentioned. The literature related to the problem investigated has been cited. Due acknowledgments have been made whenever facilities and suggestions have been availed of.

(Nilesh B. Fal Dessai)

Certificate

This is to certify that the thesis entitled ‘Development of a Text to Speech System for Devanagari Konkani’, submitted by Nilesh B. Fal Dessai, for the award of the degree of Doctor of Philosophy in Computer Science and Technology is based on his original studies carried out under my supervision. The thesis or any part thereof has not been previously submitted for any other degree or diploma in any University or Institute.

Dr. Jyoti D. Pawar
Department of Computer Science and Technology
Goa University, Taleigao Plateau
Goa, India - 403206.

Date: 30th May 2017
Place: Department of Computer Science and Technology, Goa University, Goa.

Development of a Text to Speech System for Devanagari Konkani

By

Nilesh B. Fal Dessai

Abstract

The most dynamic form of communication in everyday life is speech. The artificial production of speech using a computer is called as Speech Synthesis. Text to Speech Synthesis (TTS) Systems is a computer based system that converts text to speech and has several applications in real life. The TTS systems are especially important to the visually impaired and also to those who have lost their ability to speak. Naturalness and intelligibility are the most important qualities of a TTS system which describes the closeness of the speech generated by a TTS system to human speech and the ease with which this speech can be understood.

The main objective of this research work has been to develop a TTS system for Konkani language. As part of this research work, a study of different tools and techniques used for Indian and non-Indian languages for TTS development has been carried out. There are numerous techniques that can be used for synthesizing speech from input text.

During this research work, we have focused on study of the Formant Synthesis and the Concatenative Synthesis techniques. We initially explored the use of formant synthesis using eSpeak, an open source tool. It was tested for only Konkani numbers and we found that the sound produced is very unnatural. The performance evaluation of the TTS systems is carried out using a subjective evaluation metrics also known as volunteer based evaluation. This evaluation method is based on the feedback of volunteers who are made to listen to the speech for naturalness and intelligibility and give their score ratings to calculate the Mean Opinion Score (MOS). Twenty volunteers (male and female speakers of Konkani) in the age group of 20 to 25 years who did not have any prior exposure to speech synthesis experiments were used as evaluators.

Concatenative synthesis technique produces natural speech but requires a good speech corpus in the language for which it is to be implemented. Initially we attempted to develop a basic Concatenative TTS system for Konkani using a sample speech corpus consisting of words, diphones, phonemes and numbers.

Most of the Indian languages are syllabic in nature and syllable based approach is reported to perform better for Indian languages. Hence next we designed syllabification algorithms to generate syllables. The forward and backward algorithms designed to generate syllables were tested for Devanagari script. Experimental results indicate that syllable formation using the backward approach is best suited for syllabification of text written in Devanagari script. Given the appropriate syllable patterns for a language which uses Devanagari script, the algorithm generates the syllables for any input text. The TTS system for Konkani developed using the syllable based Concatenative approach as an enhancement to the basic Concatenative approach further improved the quality of the output speech in terms of naturalness.

The study of phonology rules for Konkani, suggested that the quality of the speech output can be improved by using the phonology rules. Hence the backward syllabification algorithm was enhanced to addresses the Konkani phonology rules like schwa deletion, jodaksharas, nasal words, diphthongs and vowel harmony. A comparison between the quality of speech synthesized by the syllable based Concatenative approach and the syllable based Concatenative approach with phonology using the Mean Opinion Score(MOS) indicate that the output speech quality is better with the syllable based Concatenative approach with phonology. The system is also able to handle some non-standard words like date **(२१\१२\१०१२)**,etc.

Festival Framework is widely used for TTS implementation. Various aspects of the Festival and Festvox Framework in Linux environment were also studied to implement a TTS system for Konkani. The synthesized speech obtained using the Festival Framework exhibits good quality based on the MOS scores obtained. Concatenative technique demands a good speech corpus to generate natural speech. A sample text of 2000 words and 200 syllables along with numbers and some phonemes were cut from the Konkani News Channel data to create a Konkani News Speech corpus to test the syllable based Concatenative TTS with phonology and the TTS system built using Festival Framework. The results have been satisfactory.

The quality of these TTS systems will further improve if we use a sufficiently large Speech Corpus. We expect to get in the very near future good quality Konkani speech corpus created out of more than thirty hours of recorded speech from the

Linguistics Data Consortium for Indian Languages (LDC-IL), Mysuru to work towards building a robust TTS for Konkani Language.

There is further scope to study and implement more language specific requirements, word and sentence stress, sentiments, emotions etc. to improve the naturalness and intelligibility of the developed TTS systems.

Acknowledgements

It gives me immense pleasure and great sense of satisfaction to express my heartfelt gratitude to those who made this research work possible.

I am grateful to the Almighty Power for being with me through this enriching journey.

I would like to express my deepest gratitude towards my guide, Dr. Jyoti D. Pawar for all her invaluable guidance, patience, encouragement, timely advice and support. She has been a constant source of inspiration and motivation in solving problems for this research work. Her holistic approach and devotion with meaningful information and thoughts has been a driving force to walk along this research path. I feel proud to have worked under her guidance.

I wish to thank the members of the FRC committee, all the Deans during my research period for their constructive feedback and guidance from the research problem definition stage to the thesis submission stage.

I am grateful to Late Dr. Madhavi Sardessai, for all the linguistic guidance, support and feedback during the initial stages of this research work. I am thankful to Dr. Rafael Fernandes for agreeing to become the VC's nominee and giving valuable suggestions to drive this research work. Without their help this thesis would not have taken this shape.

I am thankful to Prof V. V. Kamat, Head of Department, Prof. P. R. Rao, Former Head of Department, Shri. Ramdas Karmali, Shri. Ramrao Wagh and all the teaching, administrative, project staff and students of the Department of Computer Science and Technology, Goa University for their valuable support.

I am grateful to all the administrative staff of the Goa University for their support and help.

My sincere thanks are due to Dr. V. N. Shet, Principal, my Teachers and friends at Goa College of Engineering (Government of Goa), Shri. Vivek B. Kamat, Director and all my well-wishers at the Directorate of Technical Education (Government of Goa), Shri. Subhash U. Phal Dessai, Ex. Chairman, and all my friends at the Info Tech Corporation of Goa Ltd (a Government of Goa Undertaking) and the Department of Information Technology (Government of Goa) and for their constant support and guidance.

I would like to thank Shri. Chetan Desai, Dr. Gajanan S. Gawde, Dr. Anant J. Naik, Shri. Shankar Naik, Ms. Sangeeta Jadhav, Dr. Shilpa Dessai, Dr. Dimple Paul and all the Research Scholars of the department for their help from time to time.

I am indebted to the Director, Directorate of Art and Culture, Govt. of Goa for the usage of the recording studio for building the speech corpus. Thanks to Shri. Bhavesh Fulari and Shri. Daulat Palyekar for all the help and co-ordination.

My thanks are due to Shri. Prasad Velip, Shri. Satchitanand Chatim, Ms. Ashlesha Bhangui, Ms. Sumati Patil, Ms. Shweta Narvekar, Shri. Omkar Kundaikar and Shri. Vikram Gaonkar for all the help.

I am grateful to Ms. Sonia Kuncalienker, General Manager, Prudent Media (Goa) Pvt. Ltd. for providing the Konkani news voice data for this research work.

I thank the entire NLP community of India and worldwide, conference organizers, unknown reviewers, family members of my guide, friends and relatives, well-wishers, each and every individual and thing that has directly or indirectly contributed towards my research work.

A special thank you to Shri. Gaurav A. Naik, Assistant Manager, Info Tech Corporation of Goa Ltd who has been a part throughout my journey.

I express my deep gratitude to my Mother Nisha, Wife Sarika, Children Swanish and Nihika, Nieces / Nephews, Tanvi, Kaushik, Nimisha, Rohinish, Saanya and Sainish, all my family members, relatives, friends and well-wishers for their patience, care and love which drew me with inspiration to carry out my research.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	**Description**
TTS	Text-To-Speech
NLP	Natural Language Processing
DSP	Digital Signal Processing
CEERI	Central Electronics Engineering Research Institute
TIFR	Tata Institute of Fundamental Research
IIIT	International Institute of Information Technology
HCU	Hyderabad Central University
CoE	College of Enginering
IIT	Indian Institute of Technology
IISc	Indian Institute of Bangalore
CDAC	Centre for Development of Advanced Computing
CTTS	Concatenative Text-to-Speech
STTS	Statistical Text-to-Speech
HMM	Hidden Markov Model
HTTS	Hybrid Text-to-Speech
HNM	Harmonic plus Noise Model
LPC	Linear Predictive Coding
GUI	Graphical User Interface
XML	eXtensible Markup Language
API	Application Program Interface
FIFO	First In First Out
MOS	Mean Opinion Score
DMOS	Degradation Mean Opinion Score
PWD	People with Disability
NSW	Non-Standard Word
C	Consonant
V	Vowel
PU	Phoneme Unit
ASR	Automatic-Speech-Recognition
TD-PISOLA	Time Domain Pitch Synchronous Overlap and Add
LDC-IL	Linguistics Data Consortium for Indian Languages
MHRD	Ministry of Human Resource Development
HTS	Hidden Markov Model Based Text to Speech Synthesis
CSTR	Center for Speech Technology Research
CMU	Carnegie Mellon University
Nitech-HTS	Nagoya Institute of Technology
HSMM	Hidden Semi-Markov Model

Dedicated to

Swanish and Nihika

Chapter 1

Introduction

1.1. Text-to-Speech

Speech is the most fundamental form of communication for every human being. Flow of air through various articulators such as vocal track, lips, tongue, and the nose produces speech. The vocal track begins at the opening of the vocal chords and ends at the lips. The nasal track begins at the vellum and ends at the nostrils. When the vellum is lowered, the nasal track is acoustically coupled to the track to produce nasal sounds of speech [1]. Text to Speech (TTS) technology is in great demand today in view of its growing use of applications.

TTS technology aims to produce synthetic voice from textual information, thus serving as a more natural interface in human machine interaction. Progress in this technological domain has developed high quality synthesizers but focus of current research is towards improving naturalness [2]. The TTS systems are used in reading / communication aid, industrial applications, telecommunication services, hands-busy / eyes-busy applications, voice enabled emails, document readers, talking toys and books, entertainment production, e-Governance service, vocal monitoring of control systems, interactive voice response systems, etc. [3].

1.2. General Architecture of a TTS System

The general architecture of a TTS system is depicted in Fig. 1.1. Speech synthesis mainly uses two processing components; the NLP (Natural Language Processing) and the DSP (Digital Signal Processing) [4] [5]. This schematic applies for every data driven (corpus-based) TTS system, regardless of the underlying technology (e.g., unit selection or parametric). The NLP component accounts for every aspect of the linguistic processing of the input text, whereas the DSP component accounts for the speech signal manipulation and the output generation.

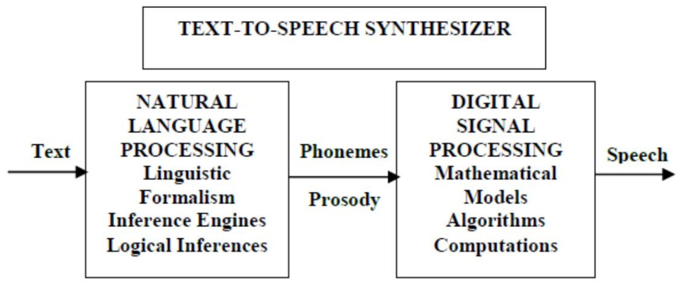


Fig. 1.1 : The General Architecture of a TTS System

1.2.1. Natural Language Processing

The responsibility of the NLP component is to provide the appropriate feed to the DSP component. It takes care of parsing, analysing and transforming the input text into an intermediate symbolic format. Furthermore, it provides all the essential information regarding prosody, i.e. pitch contour, phoneme durations and intensity. It is usually composed of a text parser, a morpho-syntactic analyser, a text normalizer, a letter-to-sound converter and a prosody generator. All these units are essential for disambiguating and expanding abbreviations and acronyms, for producing correct pronunciation, and also for identifying prosody related anchor points.

Text normalization unit relies on a rule-based approach combined with lexicon resources and complemented by exception dictionaries when necessary. The letter-to-sound converter transforms the normalized text into an intermediate symbolic form providing text’s phonetic description.

1.2.2. Digital Signal Processing

The DSP component includes all the essential units for the proper manipulation of the speech signal i.e. prosodic analysis and modification, speech signal representation, processing and generation.

The unit selection part of the DSP component performs the selection of the speech units from the speech database using explicit matching criteria. The DSP component also comprises of a signal manipulation unit.

1.3. Components of a TTS System

This section briefly discusses the main components of a TTS System. The following components make a TTS System:

1.3.1. Text Processing Front End

Indian languages are written using ‘aksharas’, which represent speech sounds. The pronunciation of the scripts in Indian Languages is almost straightforward as they use a common phonetic base. This means that there is correspondence between script and the speech. The consonant and Vowel set represents a universal phone

1.3.2. Format of Input Text

It is important that the input text in ISCII, UNICODE and in transliteration scheme of various fonts is conveniently separated from the synthesis engine. The text processing front end can provide appropriate conversion of various formats of input text into the transliteration scheme.

1.3.3. Mapping of Non-Standard Words to Standard Words

Involves mapping of the non-standard words to a set of standard words and depends on the context in which it is used. In a TTS conversion process, the input text normally is made up of a mix of standard and non-standard words. e.g. text email comprising of dictionary words (standard words) and abbreviations, digits, symbols (non-standard words).

1.3.4. Standard Words to Phoneme Sequence

Generation of a sequence of phoneme units for a given standard word is referred to as letter to sound rules. The complexity of these rules and their derivation depends on the nature of the language. Understanding of these phenomenon is important to build a good text processing module and thus to generate natural sounding speech synthesis for any language.

1.4. Challenges in TTS Systems

Development of speech synthesis system is a challenging task. The major challenges are:

(i) Design and Development of TTS systems require knowledge about speech production and details of the languages under implementation.

(ii) It is difficult to find a proper way to disambiguate homographs as the input text sometimes cannot be represented semantically by a TTS system [6]. One can only apply Heuristic techniques to guess the same.

(iii) The process of normalizing text is not straightforward as text contains numbers, abbreviations, etc. which require expansion into a phonetic representation. Also there may be many spellings for a word in any given language which is pronounced differently based on the context.

(iv) Naturalness and intelligibility contribute to the most important qualities of a speech synthesis system [7]. An ideal speech synthesizer should be both natural and intelligible [8].

(v) When the given word is not available in the dictionary, the dictionary-based approach which is supposed to be quick and accurate, fails completely. On the other hand, the rule-based approach works for all types of input, but the complexity of the rules grow for irregular pronunciations. Thus each approach has its advantages and disadvantages.

(vi) The TTS system under implementation should ideally have phonemic orthography, i.e. there should be a complete one-to-one correspondence between the graphemes and the phonemes of the language. This would mean that the spelling of a word would unambiguously and transparently indicate its pronunciation. Conversely the speaker knowing the pronunciation of a word would be able to infer its spelling without any doubt.

1.5. Current Status of TTS Systems for Indian Languages

Institutions like CEERI – Delhi, TIFR – Mumbai, IIIT – Hyderabad, HCU – Hyderabad, IIT – Chennai, CoE – Chennai, IIT – Mumbai, IIT – Delhi, CDAC – Mumbai, CDAC – Noida, CDAC – Kolkota, etc. are involved in research covering

TTS for various Indian Languages. Table 1.1 shows the current research scenario of TTS systems for Indian Languages [9].

Table 1.1 : Research Scenario for TTS on Indian Languages

Institute	**Language Cover**	**Synthesis Strategy**	**Unit / Database**	**Text/Speech Segment processing/ Tools**	**Prosody**
CEERI, Delhi	Hindi Bengali (partly)	Formant (Klatt – type) Synthesis	Syllables & Phonemes (Parameter Data Base)	Manual (Rules for smoothing) Parsing rules for Syllabification	Manual + Some rules
TIFR Mumbai	Hindi Bengali Marathi Indian English (Partly)	Format (Klatt type) Synthesis	Phonemes and other units	Automatic parsing rules for phonemization, Rules for smoothing prosody	Prosody rules
IIIT (Hyd.)	Hindi Telugu Other languages	Concatenative	Data base in required languages as per festival norms	For unit as per festival system As per requirements of festival System	Prosody studies in required language done and implemented
HCU (Hyd.)	Telugu	Concatenative	Diphone	MBROLA based	Prosody rules
IIT, Chennai	Hindi Tamil	Concatenative diphone synthesis (1400 diphones)	Syllabus (Mainly)	Automatic segmentation using group delay functions for unit selection festival System	Pitch tracks determined and implementation
CoE Chennai	Tamil	Concatenative	Diphone	Phonetic segmentation	Prosody rules
IIT Mumbai	Marathi Hindi	Concatenative	Di Syllables, -phones,	Prosody modeling using CART	Prosody rules
IIT Delhi	Hindi	Concatenative	Unit selection	Rule and corpora based method	Prosody rules
CDAC, Mumbai	Marathi Odia	Concatenative	Unit selection	speech synthesis festival based	Prosody rules
CDAC, Pune	Hindi, Indian English	Concatenative	Phonemes, other units	speech synthesis festival based	Prosody rules
CDAC, Noida	Hindi	Concatenative	Multi form units, Diphones Syllables, frequent words, phrases etc.	Parsing for syllables, Statistical processing of text for formation of phonetically rich sentences and other units	Study of intonation patterns,

CDAC Kolkata	Bengali	Concatenative	Phonemes & Sub – Phonemes (Size 1 MB)	Cool – edit Phonemic/ Segmentation	TDPSOLA /ESNOLA
CDAC Trivandrum	Malayalam	Concatenative	Phonemes	Phonemic/ Segmentation	ESNOLA
Bhrigus Software Ltd. Hyderabad	Hindi, Telugu & Others	Concatenative	Phonemes, Using festival requirements	Fest VOX tools festival	Intonation using (CART for Prosody modelling)
Prologix Software, Lucknow	Hindi	Concatenative	Di-phone data base	festival based- Fest VOX tools	-
Webel Mediatronics, Kolkata	Bengali, Hindi	Formant type	Phonemes (Parameters of phonemes)	Rules for concatenation and smoothing of parameters Text processing Rules	Intonation rules being implemented
RIT Islampur (MS)	Konkani	Concatenative	Units	Rules for Concatenation	Prosody rules
Utkal University Bhuvnesh War	Oriya	Concatenative	Phonemes	Processing of Text parsed in C & V	Prosody rules
Tapar University Patiala	Panjabi	Concatenative	Diphone, sub-syllabic	Phonetic Segmentation	Prosody rules
IISC Bangalore	Bengali Hindi Gujarati Kannada Malayalam Marathi Oriya Punjabi Tamil Telugu Pashto	Concatenative	Phones of C & V, syllables	Phonetic Transcription	Prosody rules

1.6. Overview of Konkani Language

Language is not only a rule-governed system of communication but also a phenomenon that structures our thought to a great extent and defines our social relationships in terms of both power and equality [10].

Brahmi script is the ancient script from which Indian languages have been originated. The basic units of the writing system are referred to as ‘aksharas’ which are divided into consonants(C) and vowels (V) [11] as shown in Fig. 1.2. Devanagari is a Brahmi script which was used for writing Sanskrit and other Indian languages [12]. Here the basic writing units represent syllables of various kinds (e.g. CV, CCV, CCCV, CVC, VC) instead of representing individual consonant (C) and vowel (V) sounds. The V and C component of Brahmi symbols are clearly distinguishable, since they are syllabic and hence Brahmi script is called as an alpha-syllabic writing system.

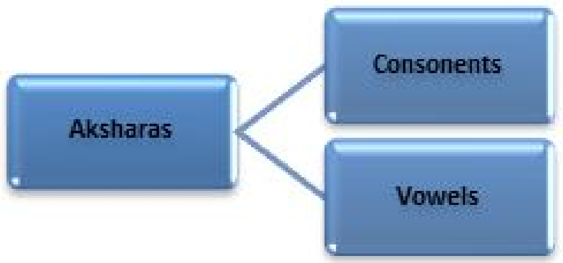


Fig. 1.2 : Comparion of Aksharas

a) A consonant is a sound in spoken language (letter of the alphabet denoting such a sound) that has no sounding voice (vocal sound) of its own, but must rely on a nearby vowel with which it can sound. A consonant is characterized by closure at one or more points along vocal tract such as lips, tongue and teeth. Each consonant has the specialty of having an inherent vowel.
b) A vowel is a sound in spoken language that has a sounding voice (vocal sound) of its own. It is produced by comparatively open configuration of the vocal tract and can be sounded on its own.

Vowels are always voiced sounds produced with the vocal cords in vibration. They play a major role in the pronunciation of any word and their duration in any word is also significant based on which they are classified as long and short vowels. The consonants may be either voiced or unvoiced [13] [14].

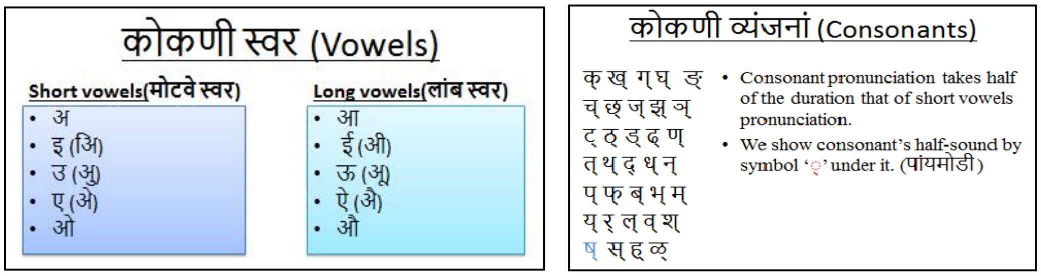
Konkani, the official language of the State of Goa in India is also the minority language in the States like Karnataka, Kerala and Maharashtra in India. Konkani is being spoken by about 3.6 million people. Goans mostly use Devanagari or Roman script and accordingly the expression 'Devanagari Konkani' refer to the writing of

Konkani in Devanagari script and ‘Romi Konkani’ refer to the writing of Konkani in Roman script. In Karnataka, Konkani speaking people use Kannada script or Devanagari script. The Konkani community in Kerala use Malyalam script or Devanagari script. There is also usage of Arabic script to write Konkani in Bhatkal Taluka of Karnataka. In the context of dialects, Antruz Konkani, Bardeskari Konkani, Pednekari Konkani, Xaxtti Konkani, etc. together is referred to as Goan Konkani ie those are the languages primarily spoken in the State of Goa. Malvani Konkani, Karwari Konkani, Mangalorean Konkani, etc. are some of the dialects spoken outside the State of Goa. The dialects are traditionally written in the respective script e.g. Antruz Konkani is written in Devanagari script whereas, the Bardeshi and Xaxtti variants are written in Romi script [15]. Our research work focuses on Devanagari Script.

Words can be formed in Konkani language using its diphone. Splitting words into diphone for a written language is different from a spoken language [16].

Konkani has 12 vowels. The independent form and the dependent form are the two forms of a vowel. Accordingly they are referred to as 'swaras' and 'matras'. Swaras are used to pronounce vowels when they are in isolation, not associated and not attached to any consonant. Whereas the matras are used when the vowels are always attached, associated and not in isolation.

Konkani has 36 consonants, which are divided into groups and then subgroups. The first 30 consonants form a grouping of five sounds and further subgrouping of 3 sounds as voiced, unvoiced and nasal. In each group the last letter represents a nasal pronunciation called as 'anunasik'. Fig. 1.3 shows the Konkani vowels and consonants.


कोकणी स्वर (Vowels)

Short vowels(मोटवे स्वर)
- अ
- इ (अि)
- उ (अु)
- ए (अे)
- ओ

Long vowels(लांब स्वर)
- आ
- ई (अी)
- ऊ (अू)
- ऐ (अै)
- औ

कोकणी व्यंजनां (Consonants)

क् ख् ग् घ् ङ्
च् छ् ज् झ् ञ्
ट् ठ् ड् ढ् ण्
त् थ् द् ध् न्
प् फ् ब् भ् म्
य् र् ल् व् श्
ष् स् ह् ळ्

- Consonant pronunciation takes half of the duration that of short vowels pronunciation.
- We show consonant’s half-sound by symbol ‘्’ under it. (पांयमोडी)

Fig. 1.3 : Konkani Vowels and Consonants

A text to phoneme rule defines generation of a sequence of phonetic units for a given standard word and is also referred to as letter to phoneme rule. In Konkani, the basic sound units called aksharas, consists of consonants (व्यंजनां) and vowels (स्वर). Vowel are sound production is in terms of highest point of tongue and the position of lips. None of the articulators come into play here, where as consonant sound is produced by obstruction to the passage of airstream by articulators like lips, tongue and teeth. All consonants inherently occur with the vowel "अ" which is called as schwa (discussed later). In order to depict consonants in their pure representation, characters are shown with broken leg (पांयमोडी).

1.7. Motivation

India is a Multi-lingual country with variety of scripts and hundreds of spoken dialects. It is desired that information along with ICT based services are delivered to a large portion of the population in their own language in the form of voice. Lot of research work is currently carried out in the area of text to speech processing inclusive of many Indian languages and the synthesis systems are in great demand.

Naturalness and intelligibility are the most important qualities of a TTS system which describes the closeness of the speech generated by a TTS system to human speech and the ease with which this speech can be understood. And hence ideal speech synthesizers usually try to maximize both these characteristics [7].

No concrete work is carried out for Konkani in the area of TTS and hence this research work attempts to develop a TTS system for Konkani.

1.8. Thesis Contributions

The main goal of this research work is development of a TTS system for Konkani. During the process of design and implementation, the following tasks were carried out.

(i) Implementation of a TTS prototype for Konkani using eSpeak.

(ii) Development of a Konkani TTS system using concatenative synthesis.

(iii) Comparison of syllable based TTS systems.

(iv) Design and implementation of syllabification algorithms for Devanagari languages.

(v) Identifying and applying Konkani phonology rules to improve naturalness of the syllable based concatenative TTS system.

(vi) Design and implementation of algorithms for implementing phonology in Konkani TTS

(vii) Development of a TTS system for Konkani using Festival Framework.

1.9. Organization of the Thesis

This thesis comprises of nine chapters detailed as below:

(i) Chapter 1 gives the introduction to TTS with a brief on Konkani.

(ii) Chapter 2 discusses the background of TTS outlining the techniques for speech synthesis, tools for speech synthesis, tools for recording and labeling and the performance evaluation method for TTS systems.

(iii) Chapter 3 discusses the implementation of a TTS system for Konkani using eSpeak.

(iv) Chapter 4 discusses the development of a TTS system using concatenative speech synthesis. The implementation details are presented along with the experimental results.

(v) Chapter 5 gives an overview of syllable based TTS systems along with a comparative study of the syllable based TTS systems studied for various languages.

(vi) Chapter 6 discusses the proposed syllabification Algorithms for Devanagari languages. The implementation details are presented and experimental results reported.

(vii) Chapter 7 details the syllabification using Konkani phonology rules towards improving naturalness of the TTS system. The implementation details are presented and experimental results reported.

(viii) Chapter 8 discusses the implementation of a TTS system for using festival framework. The implementation details are presented along with the experimental results.

(ix) Chapter 9 gives an overview of the HMM-based speech synthesizers.

(x) Chapter 10 concludes the thesis with directions for future work.

Chapter 2

Background

2.1. Speech Synthesis Techniques.

Todays general purpose speech synthesis systems are designed to exhibit high quality and naturalness. Systems built for specific domain and application requirements target to achieve quality results and performance. There is always a tradeoff between the quality of speech and synthesis speed depending on the application context. The typical trade-off between the speech quality and synthesis speed and responsiveness, or processing requirements and storage space, can significantly deviate. Speed optimization and response latency minimization are two crucial issues identified in this research from the viewpoint of visually impaired users.

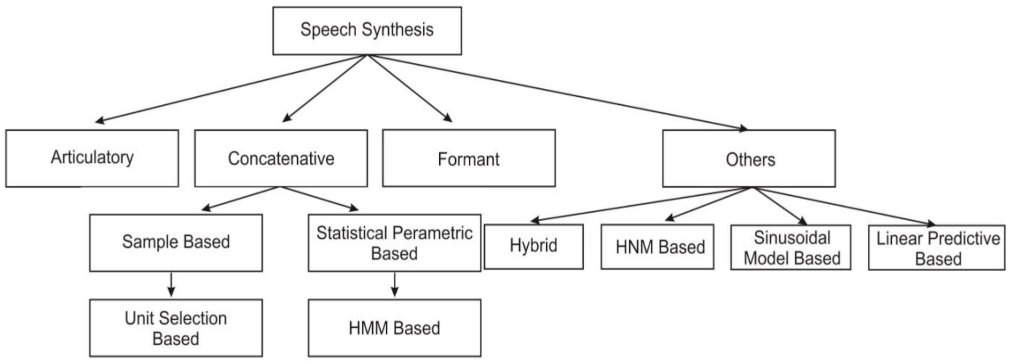


Fig. 2.1 : Classification of the Speech Synthesis Techniques

It is desired that for a visually challenged user the delay between action and response is not noticeable. Many users suggest including an option of allowing degraded speech quality in exchange of increased speed. In order to meet these requirements, two strategies are followed: database size reduction and pre-recording and pre-synthesis of words and phrases. Research in this context is focusing on efficient techniques for database coding and compression, as well as database reduction and speed optimization for TTS systems with minimal speech quality degradation. There are several methods available to synthesize speech with their respective strengths and weaknesses and suits a specific language while does not suit others. e.g . a TTS system for Arabic language [17] using neural networks. Speech synthesizers come in varying qualities for different languages for various products [18]. A classification of the speech synthesis techniques is given in Fig. 2.1. Following are some of the commonly used speech synthesis techniques.

2.1.1. Articulatory Synthesis

The computational techniques which are based on models of the human vocal tract are called articulatory synthesis techniques. They also model the articulation processes occurring in the vocal tract. Philip Rubin, Tom Baer, and Paul Mermelstein developed the first articulatory synthesizer at the Haskins Laboratories 1970s which was mostly used for laboratory experiments [3] [19].

2.1.2. Formant Synthesis

Formant synthesis uses an acoustic model to generate artificial speech. This synthesis does not involve recorded speech data. In formant synthesis the output artificial speech is generated by varying certain parameters over time such as fundamental frequency, noise levels, etc. Systems based on formant synthesis generate a robotic-sounding speech which is artificial in nature where naturalness of the speech quality is not expected to be at its best.

An important aspect of the synthesized speech generated is reduction in the number of acoustic glitches even at high speeds thus making it intelligible over concatenative synthesis. Formant synthesizers do not use recorded speech corpus and hence are smaller in size and find their use mostly in embedded systems. Prosody, intonations, emotions, etc. can be easily implemented in formant synthesis with limited memory and computing power [3] [20].

2.1.3. Concatenative Synthesis

Concatenative synthesis uses recorded speech data to generate a natural speech output using concatenation. The disadvantage of this speech synthesis technique is the presence of audible glitches in the generated speech due to the usage of small size speech segments. Concatenative synthesis techniques are further classified as: Unit Selection based Synthesis and Hidden Markov Based (HMM) synthesis [3] [21] [22].

2.1.3.1. Unit Synthesis

Recorded speech comprising of individual phones, half-phones, diphones, syllables, morphemes, words, phrases, and sentences make the database used by Unit selection synthesis. Since Unit selection synthesis applies very little signal processing to the recorded speech, it provides the greatest naturalness and may be indistinguishable from real human voices if tuned properly. It is possible to detect the unnatural segments in unit selection TTS systems. Various automated methods are proposed by researchers in this regards [23] [24].

Diphone synthesis uses a minimal speech corpus and is a type of Unit Synthesis. It uses the all the diphones (sound-to-sound transitions) present in a language. The DSP technique used in the synthesis method, targets the required prosody at runtime by superimposing it on these minimal speech units. Diphone synthesis finds its use in research as it has a number of implementations and most of them are available without any cost. However its use in commercial applications is dropping down because of the drawbacks of concatenation. [24] [25] [26].

2.1.3.2. HMM based Synthesis

It is based on Hidden Markov Models (HMMs) wherein based on the Maximum likelihood criterion, the Speech waveforms are generated from the HMMs. It is the HMM which models the speech frequency spectrum (vocal tract), Fundamental frequency (vocal source), and duration (prosody) simultaneously. It involves a training part and a synthesis part. The spectrum and excitation parameters are extracted from speech database and modelled by context dependent HMMs in the training part and the context dependent HMMs are concatenated according to the text to be synthesized in the synthesis part [32].

2.1.3.3. Domain-Specific Synthesis

Prerecorded words and phrases are concatenated to generate speech from text in domain-specific synthesis. In applications like talking clocks, talking calculators, railway announcements, banking counter, a domain-specific synthesis would be more suitable where the output speech is limited to a particular domain, The technology is very simple to implement, and has been in commercial use for a long time. In terms of naturalness of the system, they match the original recordings. An important thing to note about the domain specific synthesis is that, they cannot be used for any general purpose TTS, as the database is limited to words and phrases. [26].

2.1.4. Comparison of Synthesis Techniques

A brief comparison of the commonly used methods i.e. articulatory, concatenative and formant synthesis [27] [28] is given in Table 2.1.

Table 2.1 : Comparison of Speech Synthesis Techniques

.	**Synthesis Techniques**		
	Articulatory	**Concatenative**	**Formant**
Naturalness	Excellent	Excellent	Satisfactory (Robotic)
Speed of Processing	Moderate	Slower	Faster
Implementation Ease	Difficult	Easier	Easy
Challenging Aspects	Critical in operation of articulator and vocal cords	Choice of unit and space required for its storage	Set of parameters controlling speech

The quality of synthetic speech produced by a synthesizer in terms of naturalness is with the use of concatenative synthesis technique. Concatenative synthesis maintains a waveform repository of basic speech units that encapsulate the sounds in a particular language along with co-articulation, prosody and transitions exhibited by the system [7]. One of the most important aspects of concatenative synthesis is to find the correct unit length representing the language being modeled. The basic types of units are phonemes, diphones, syllable and polysyllables [29]. As Indian languages are syllabic, syllables are best suited as the basic units over phonemes and diphones in this context. Also the concatenation points are reduced and the syllable boundaries are characterized by low energy regions with syllable as the basic unit [30].

2.1.5. Other Speech Synthesis Techniques

We also reviewed the following other speech synthesis techniques.

2.1.5.1. Hybrid Synthesis

It involves both concatenative (CTTS) and statistical (STTS) text to speech synthesis. CTTS synthesize speech from a recorded speech and contains discontinuities. On the other hand, STTS systems, synthesize speech that is free from such discontinuities. In order to gain advantages of each of the two approaches, CTTS and STTS combines into a hybrid TTS (HTTS) system. The quality of this hybrid system depends on the quality of the baseline CTTS and STTS systems [31].

2.1.5.2. HNM based Synthesis

In Harmonic plus Noise Model (HNM) synthesis speech signals are represented as a time-varying harmonic component and a modulated noise component. The harmonic part responds to the quasi-periodic components of the speech and the noise part responds to non-periodic components [33]. The parametric representation of speech using HNM provides a straightforward way of smoothing discontinuities of acoustic units around concatenation points. HNM provides high-quality speech synthesis while outperforming other models for synthesis.

2.1.5.3. Sinusoidal Model based Synthesis

This synthesis is based on the assumption that the speech signal can be represented as a sum of sine waves with time varying amplitude and frequencies. A sinusoidal model is characterized by the amplitudes, frequencies and phases of the component sine waves. The resulting synthetic waveform preserves the general waveform shape and is essentially perceptually indistinguishable from the original speech [34].

2.1.5.4. Linear Predictive based Synthesis

The linear predictive coding (LPC) is based on the source-filter-model of speech. The digital filter coefficients are estimated automatically from a frame of natural speech. This speech synthesizer is developed from speech coders [35].

2.2. Tools for Speech Synthesis

There are limited resources available for Konkani Language for speech processing. Some readily available tools like eSpeak [36], Dhvani [37] [38], Vani [39] [40] and MAGE [41] were studied. We implemented a Konkani TTS system using eSpeak which uses formant synthesis. An extensive study was also carried out on Festival Framework [42] for building a TTS system. Festival is one of the most popular frameworks for building a TTS system and is used by many researchers in the TTS domain particularly for Indian languages.

2.2.1. eSpeak

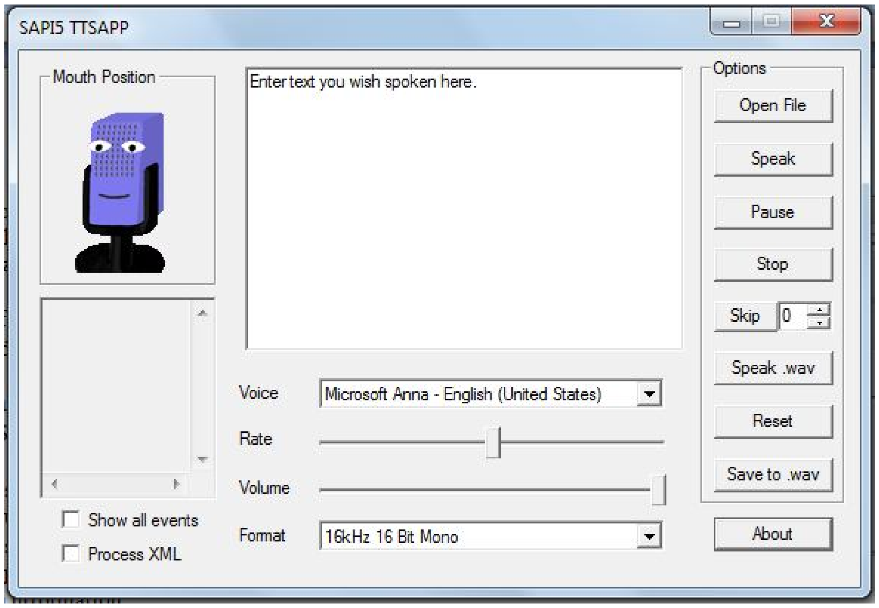
eSpeak is an open source tool for converting text to speech. eSpeakedit, an editing tool of eSpeak has a Graphical User Interface (GUI) as shown in Fig. 2.2, where the text can be played. eSpeak has the following features [36]:

(i) Includes different voices that can be altered.

(ii) Translates text into phoneme codes which helps to compile other languages.

(iii) Using eSpeakedit one can generate phoneme data.

(iv) One can generate phoneme data and tune it as per the requirement using the available tools.



Enter text you wish spoken here.

Voice: Microsoft Anna - English (United States)

Skip: 0

Format: 16kHz 16 Bit Mono

Fig. 2.2 : The GUI of eSpeak

2.2.2. Dhvani

Dhvani is a TTS system for Indian languages developed at the Indian Institute of Science, Bangalore [38]. It uses diphone based concatenation and is implemented for Indian languages like Hindi, Marathi, Tamil, Telgu, Gujarati, Bengali, Kannada, Malayalam and Punjabi. For every language a Unicode parser is designed based on direct grapheme to phoneme mapping. The architecture of Dhvani includes: text parser, text to Dhvani phonetic script, grapheme to phoneme conversion, sound dataset and speech synthesis as shown in Fig. 2.3.

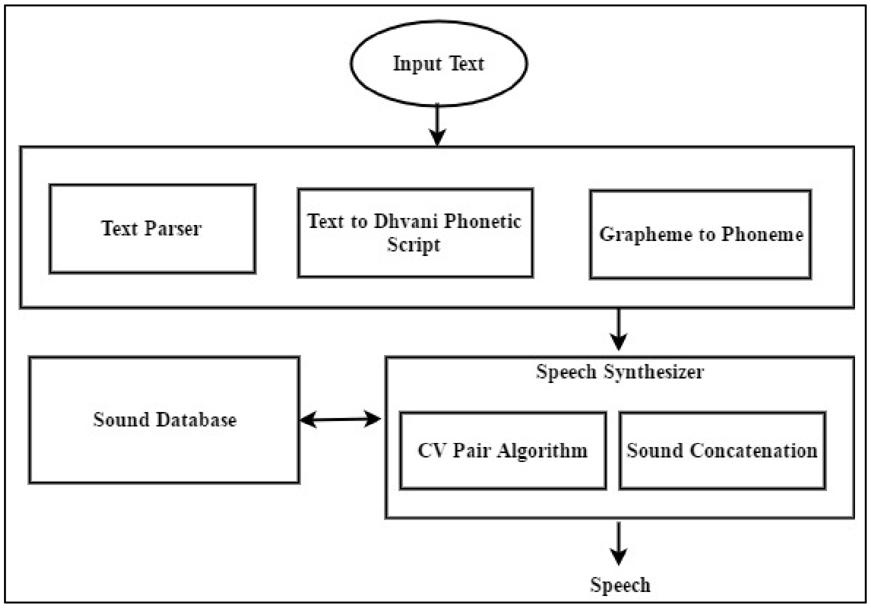


Fig. 2.3 : Architecture of Dhvani

2.2.3. Vani

Vani is TTS synthesizer for Indian Languages and uses an encoding scheme known as vTrans [39] [40]. A vTrans file allows a user to encode the text to be read and the way it is to be read. A signal processing module is used to make appropriate variations to the sound database. vTrans is an XML (eXtensible Markup Language) document containing a head section and a body part. The head section defines the parameters and styles and the body section contains several nested tags. The text is put only in the innermost of the tags. The text within the tags is iTrans encoded. The major components of Vani system are iTrans Data, DSP Data, Design module, Vani Data and Concatenizer as shown in Fig. 2.4.

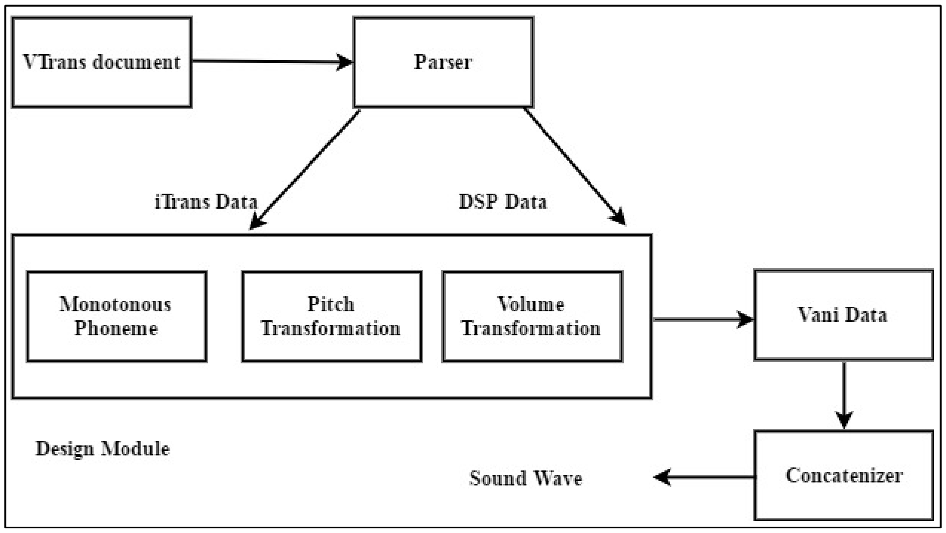


Fig. 2.4 : Architecture of Vani

2.2.4. MAGE

MAGE is a platform for reactive speech synthesis wherein both phonetic content and prosody of synthetic speech can be controlled by the user in real-time [41]. It works based on the Hidden Markov Model based Text to speech synthesis (HTS) engine and synthesizes speech using a limited number of labels preceding and succeeding the currently synthesized phoneme.

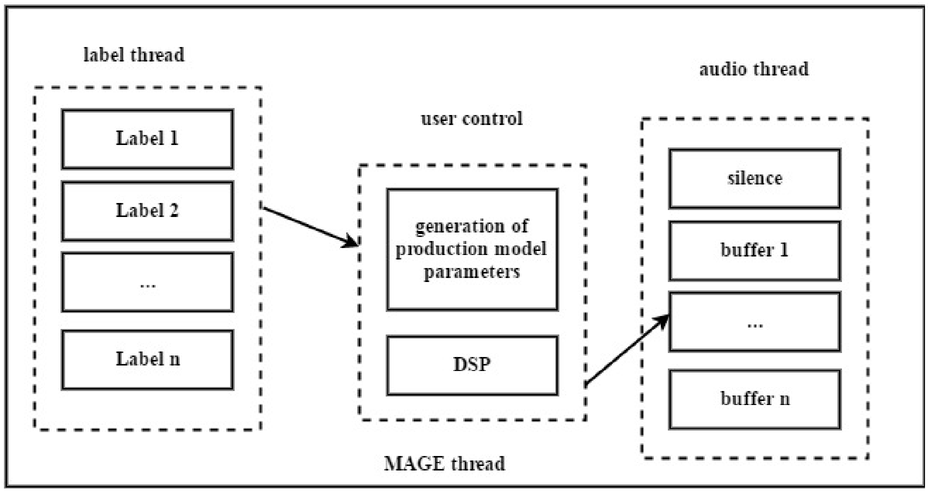
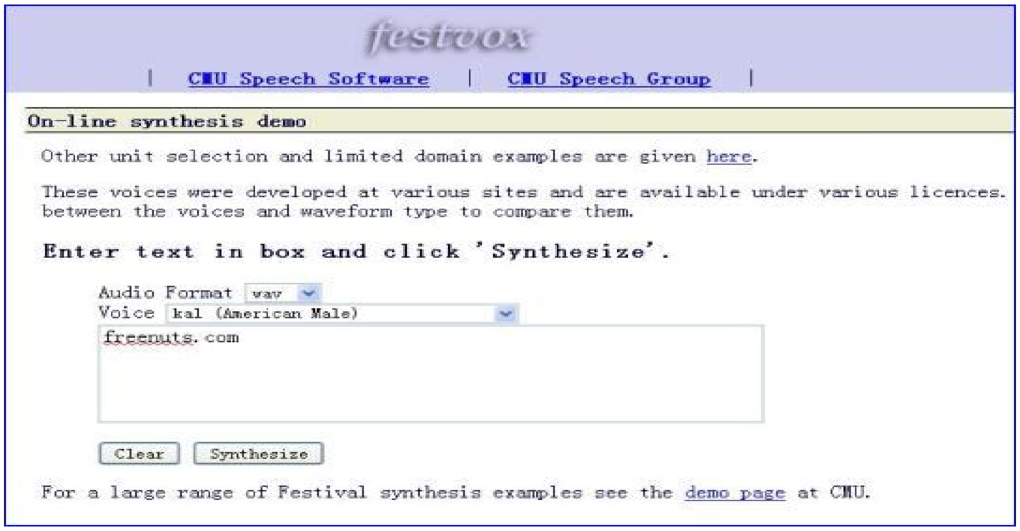


Fig. 2.5 : Architecture of MAGE

MAGE has user-friendly (Application Program Interface) API which helps the developers for easy integration of reactive speech synthesis to their applications. The multithread architecture of MAGE is shown in Fig. 2.5. The labels are added in its own thread in a First-In-First-Out (FIFO) ordering. The parameter trajectories and the speech samples are generated after the MAGE pops labels out of the FIFO. Samples are sent to the audio thread as consecutive buffers and each buffer contains the audio samples of one label. As there is delay noticed in one-label, the first audio buffer always corresponds to silence.

2.2.5. Festival

Festival is multi-lingual TTS system designed to support multiple languages including Indian languages. The framework is designed for at least three levels of users. The first level is for users wanting high quality speech from arbitrary text. The second level is for the users developing systems for specific languages and wishes to include synthesis output. The third level is for the users developing and testing new synthesis techniques. The Konkani TTS that we have implemented in festival and discussed in chapter 8 comes under the second level users. In festival, each module in the system is given an utterance that manipulates it and then pass on to the next module. An utterance consists of a set of items which are related by a set of relations [43]. Festvox is a component of festival used to build synthetic voice. A snapshot of an online demo of Festvox is shown in Fig. 2.6.



On-line synthesis demo

Other unit selection and limited domain examples are given here.

These voices were developed at various sites and are available under various licences. between the voices and waveform type to compare them.

Enter text in box and click 'Synthesize'.

Audio Format wav
Voice kal (American Male)

freenuts.com

For a large range of Festival synthesis examples see the demo page at CMU.

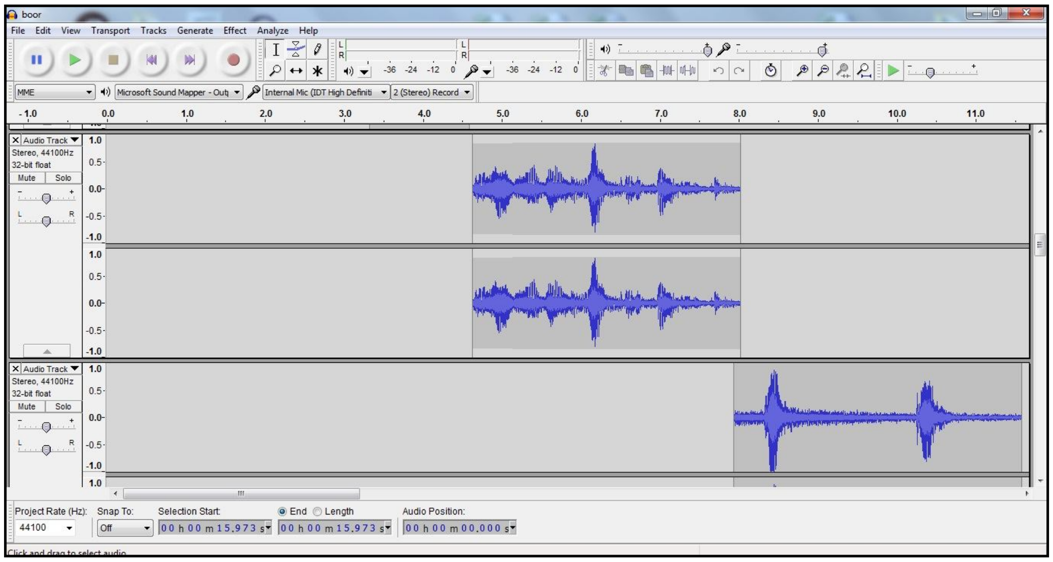
Fig. 2.6 : Online Demo of Festvox

2.3. Tools for Sound Recoding

Building of a corpus based TTS systems involves recorded speech. In our work due to non-availability of Konkani speech, we have created a speech corpus on experimental basis in a laboratory as well as in studio environment as per our needs and requirements. The laboratory recording was carried out at a sampling rate of 44100 Hz per second, compress bit rate of 128 kbps. The sound files were recorded in wav format. The studio recording was carried out at a sampling rate of 48000 Hz per second, bit depth of 16 bit, mono channel. The sound files were also recorded in wav format. The following tools we used for recording and editing of the speech files.

2.3.1. Audacity

Used for recording, editing and analyzing voice files. It is an open source tool, easy-to-use and multilingual audio editor and recorder for Windows, Mac OS X, GNU/Linux and oth*e*r operating systems [44]. Audacity can record live audio as well as capture streaming audio using a microphone or a mixer. It allows importing of sound files, editing them, and combining them as new recordings. It allows exporting your recorded files in many formats. It supports 16-bit, 24-bit and 32-bit (floating point) samples. It allows easy editing with Cut, Copy, Paste and Delete and Unlimited sequential Undo (and Redo) to go back any number of steps. The working of Audacity tool is shown in Fig.2.7.


boor
File Edit View Transport Tracks Generate Effect Analyze Help
MME
Microsoft Sound Mapper - Outp
Internal Mic (IDT High Definiti
2 (Stereo) Record
Audio Track
Mute
Solo
Click and drag to select audio.

Audio Track: Stereo, 44100Hz, 32-bit float

Project Rate (Hz): 44100
Snap To: Off
Selection Start: 00 h 00 m 15.973 s
End: 00 h 00 m 15.973 s
Audio Position: 00 h 00 m 00.000 s

Fig. 2.7 : Working of Audacity Tool

2.3.2. Wavesurfer

WaveSurfer is an open source tool for sound visualization and manipulation of wave files [45] [46]. It allows users to create their own configurations. It allows new functionality to be added through plugin architecture and can be used as a widget in custom applications. It supports on Linux, Windows and OSX platforms.

Wavesurfer support transcription for file formats which reads, and writes HTK (and MLF), TIMIT, ESPS/Waves+, and Phondat. Refer Fig. 2.8 for a snapshot showing the working of a Wavesurfer tool.

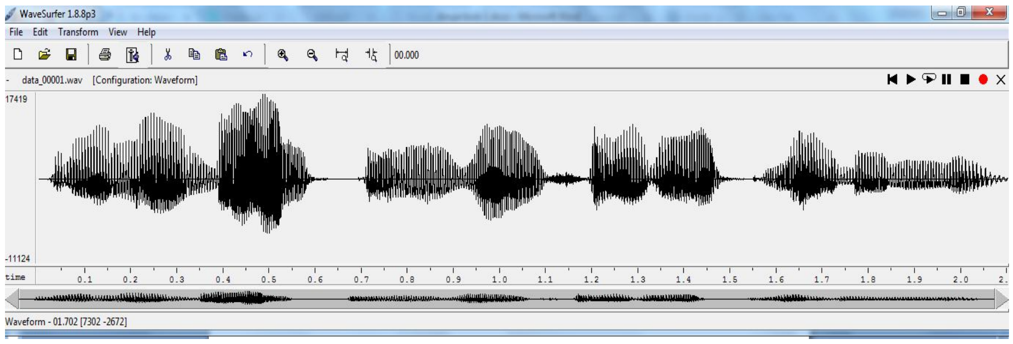

WaveSurfer 1.8.8p3
File Edit Transform View Help
00.000
data_00001.wav [Configuration: Waveform]
17419
-11124
time
Waveform - 01.702 [7302 -2672]

Fig. 2.8 : Working of Wavesurfer Tool

2.4. Performance Evaluation of the TTS System

The testing process of a TTS system involves the following:

(i) Testing techniques: consisting of module testing, system testing, performance testing and field testing.

(ii) Subjective evaluation metrics: consisting of naturalness test, intelligibility test, accuracy test and comprehensibility test.

(iii) Speech quality evaluation procedures: consisting of evaluators selection criteria, evaluators training guidelines, evaluators training overview, pilot test and test environment setup and test execution.

(iv) Analysis of speech quality: consisting of segmental evaluation, language specific features evaluation and prosody evaluation.

Evaluation of overall speech quality is utmost important from the end users perspective, as they are usually interested in the performance of a system as a whole that accepts text as input and generates the corresponding speech. Testing and evaluation of a TTS system aims at judging the speech quality in terms of its similarity to the human voice and by its ability to be understood. We access the reliability of what is being tested in performance testing, where the specifications are verified and performance in terms of responsiveness and stability is determined [47].

The performance evaluation of the TTS systems is carried out using a subjective evaluation metrics also known as volunteer based evaluation or evaluator based evaluation [48]. This evaluation method is based on the feedback of volunteers who are made to listen to the speech for naturalness, intelligibility, accuracy and comprehensibility (discussed later in the chapter) and give their score ratings. An example score rating on a scale from 1 to 5 is shown in the Table 2.2 to calculate the Mean Opinion Score (MOS) for naturalness and intelligibility respectively.

Table 2.2 : Speech Perception Rating

Scores for Naturalness			**Scores for Intelligibility**		
Rating	Quality	Description	Rating	Quality	Description
5	Excellent	Imperceptible	5	Excellent	Complete relaxation possible; no effort required
4	Good	Just perceptible but not annoying	4	Good	Attention necessary; no appreciable effort required
3	Fair	Perceptible and slightly annoying	3	Fair	Moderate effort required
2	Poor	Annoying but not objectionable	2	Poor	Considerable effort required
1	Bad	Very annoying and objectionable	1	Bad	No meaning understood with any feasible effort

The Mean Opinion Score (MOS) test is the simplest method to evaluate the quality of a speech synthesis system. It gives a numerical indication of the quality of the synthesized speech. Using the above scores, the MOS for a particular parameter for the speech under evaluation is computed as follows:

$$\text{MOS} = \sum_{j=1}^{M} \left(\sum_{i=0}^{N} \text{Si} \;/\text{N} \right) / \text{M}$$

Where S_i = Score of the i^{th} evaluator, N = Number of evaluators, M = Number of sentences, J = Sentence index.

Degradation of Mean Opinion Score (DMOS) is another test that can be used to test the speaker similarity. In the DMOS test, the volunteers are made to listen to the target speaker’s natural speech and a test sample (same sentence) and they are asked to give their rating. A similarity score of five-point scale (5: exactly the same to 1: completely different) is used for the purpose of giving the score.

2.4.1. Naturalness Test

Here the focus of the evaluator should be on naturalness of synthetic speech ie closeness to human voice. In terms of test data, we use sentence level or short paragraph data that should cover all the possible variations mainly: numerals, abbreviations and acronyms, symbols, English words written in Latin and in Indian scripts and different types of sentences like declarative, negative, and exclamatory. Sentences should be selected from different areas namely news, stories, sports etc.

2.4.2. Intelligibility Test

Intelligibility is one of the important factors affecting speech quality. The Intelligibility refers to the accuracy with which each word is pronounced so that normal listener can understand the spoken word or phrase. In this method, the focus of evaluator should be on intelligibility of synthetic speech.

2.4.3. Accuracy Test

For accuracy calculation proper selection of test data is crucial. All such data whose expected output is well defined can be considered for accuracy test. Categorization of test data is as follows:

(i) Number Handling: Digits (Phone number ९८५०४०९५७९), Fractions (2/3), Numbers (1004), Numerals (1st, XII)

(ii) All Date formats (01/12/14, 9 जून, 2014, ०२/०६/२००७)

(iii) Foreign words (Latin script)

(iv) English words transliterated in Indian script (इंटरव्यूव)

(v) Acronyms (etc., Prof., डॉ., Rs.)

(vi) Abbreviations (SBI, DRDO, ऩीडब्लूडी)

(vii) Names

(viii) Addresses

(ix) Homographs

(x) Punctuations and Brackets (, , ; , “, ”, –, [,], (,), {, })

(xi) Special Symbols ($, @, %)

Punctuation handling is application dependent e.g. if the application is designed for people with disability (PWD) then the system must readout all the punctuation marks. For e.g. (True) should read as ‘round bracket open’ ‘True’ ‘round bracket closed’; applications designed for people other than PWD appropriate pauses for punctuations should be checked.

2.4.4. Comprehensibility Test

Comprehension means up to what extent the message received is understood. In the intelligibility test, the evaluator will emphasize only on recognition of each word without concentrating on the meaning of the sentence. Comprehensibility test should be carried out when the system achieves the intelligibility up to acceptable level. In Comprehensibility test, evaluator will be asked to listen to a paragraph or story (100 – 150 words) and based on that a series of questions will be posed. Questions should be framed in such a way that whether the evaluator has understood the paragraph heard or not can be observed. A two point scale (0, 1) is recommended with 0 indicating incorrect response and 1 indicating correct response.

There are two kinds of questions that can be prepared for comprehension test.

1. Open ended questions: the evaluator will have to write down the answer.
2. Closed ended questions: the evaluator will have to choose an appropriate answer from multiple options, generally 2 or 4 options are provided.

The performance evaluation of a TTS system can also be performed using objective tests. These tests are used to measure how the synthetic voice differs from the natural utterance. They can be used as diagnostic tools for refining the synthetic speech. In our experiments, we have used only subjective - MOS tests for naturalness and intelligibility.

2.5. Summary

To understand the fundamentals of building a TTS for Konkani, the underlying techniques and tools in use were studied. The study of this background has given us insights of the TTS system development.

Chapter 3

TTS System using eSpeak

3.1. Introduction

The eSpeak and SpeakVolumes [49] are examples of two speech synthesis tools used for English and other languages for Linux and Windows environment. eSpeak is a free, open-source software while SpeakVolumes is commercial. eSpeak is mostly used by researchers to implement and understand the formant synthesis. Continuing with the study of different techniques for speech synthesis, Indian Language scripts, techniques for generating prosody, eSpeak, a readily available TTS, was studied and implemented for Konkani.

The eSpeak tool generates speech using acoustic model of speech generation [36]. The eSpeak tool can prepare and compile phoneme data to analyse a certain number of frame-sequences in editor through imported or generated files, as well as allows the selection of the types of sounds. It takes input character in UTF-8 encoding format. Although a high speed, small storage and clear speech is generated by eSpeak, it is low on naturalness compared to the speech corpus based synthesizers. The eSpeak speech synthesizer supports several languages including some of the Indian languages.

3.2. Related Work

eSpeak provides rules and phoneme files for more than 30 languages. Following are the two related papers reviewed:

(i) An implementation of a TTS implementation for Albanian, an Indo-European language using eSpeak [50] explores the possibility of adaptation of the existing text to speech converters into Albanian language. eSpeakedit is used for generating and improving words in order to adapt them to Albanian. Praat software is used for the analysis and reconstruction of acoustic signals.

(ii) Another work in Indian context discusses improvements in the formant based text to speech synthesis system for Punjabi text input [51]. Here after analysis of the Punjabi input some faults are identified and corrected using eSpeakedit.

3.3. Modification to eSpeak

It was observed that TTS is already implemented using eSpeak for many Indian languages like Hindi, Kannada, Malayalam, Nepalese, Punjabi and Tamil. However it was not implemented for Konkani language. Thus for implementing a TTS for Konkani we selected Hindi language in eSpeak for modification as the script for both the languages is same.

With an objective to produce a prototype of Konkani speech with eSpeak, we worked on the following files:

(i) The phoneme definition file: contains the phoneme definitions for the vowels and consonants for Devanagari languages as shown in Table 3.1 and Table 3.2.

(ii) The rules file: contains the spelling-to-phoneme translation rules.

(iii) The list file: contains pronunciations for numbers, letter and symbol names, and words with exceptional pronunciations.

These files were required to be updated to suit the Konkani implementation.

Table 3.1 : Konkani Vowels with Phoneme Definition

Sr. No.	Devanagari Vowels	Linguistics Representation	Phoneme Definition
1	अ	***A***	#X1a:
2	◌ा	*Ā*	#X2a:
3	ि◌	*I*	#X1I
4	◌ी	*Ī*	#X2i:
5	◌ु	*U*	#X1U
6	◌ू	*Ū*	#X2u:
7	◌े	*E*	#X2e:
8	◌ै	*Ai*	#X2E:
9	◌ो	*O*	#X2o:
10	◌ौ	*Au*	#X2O:

Table 3.2 : Konkani Consonants with Phoneme Definition

Sr. No.	Devanagari Consonants	Linguistics Representation	Phoneme Definition
1	क	*Ka*	k@
2	ख	*Kha*	kh@
3	ग	*Ga*	g@
4	घ	*Gha*	gh@
5	ङ	*ṅa*	N@
6	च	*Ca*	c@
7	छ	*Cha*	ch@
8	ज	*Ja*	J@
9	झ	*Jha*	Jh@
10	ञ	*Ña*	n^@
11	ट	*ṭa*	t.@
12	ठ	*ṭha*	th.@
13	ड	*ḍa*	d.@
14	ढ	*ḍha*	dh.@
15	ण	*ṇa*	n.@
16	त	*Ta*	t@
17	थ	*Tha*	th@
18	द	*Da*	d@
19	ध	*Dha*	dh@
20	न	*Na*	n@
21	प	*Pa*	p@
22	फ	*Pha*	ph@
23	ब	*Ba*	b@
24	भ	*Bha*	bh@
25	म	*Ma*	-
26	य	*Ya*	j@
27	र	*Ra*	r@
28	ल	*La*	l@
29	व	*Va*	v@
30	ष	*ṣa*	s.@
31	श	*Śa*	S@
32	स	*Sa*	s@
33	ह	*Ha*	l.@
34	ळ	*ḷha*	S.@
35	क्ष	*kṣa*	N.@
36	ज्ञ	*Jña*	-

3.4. Implementation

The hindi list file was modified in eSpeak to define pronunciation for Konkani numbers in order to produce speech for Konkani numbers (0 - 100). eSpeak edit was used for compiling the master phoneme file and dictionary files to generate the phoneme translation for the given text. Table 3.3 shows sample Konkani numbers (0 - 20) with phoneme definition.

Table 3.3 : Konkani Numbers with Phoneme Definition

Sr. No.	Numbers (Konkani Devanagari)	Numbers (English)	Phoneme Definition
1	०	Zero	*S'u:n.jV*
2	१	One	'e:k
3	२	Two	d'o:n
4	३	Three	*t'i:n*
5	४	Four	c'a:r:
6	५	Five	p'a:nc
7	६	Six	s'@
8	७	Seven	s'a:t
9	८	Eight	'a:th.
10	९	Nine	n'O:
11	१०	Ten	d'a:
12	११	Eleven	'e:kr@2
13	१२	Twelve	b'a:r@2
14	१३	Thirteen	t'E:r@2
15	१४	Fourteen	ch'O:d@2
15	१५	Fifteen	p'Vndr@2
17	१६	Sixteen	s'o:l@2
18	१७	Seventeen	s'Vt:r@2
19	१८	Eighteen	Vth.:'a:r@2
20	१९	Nineteen	'e:kUnn'Is
21	२०	Twenty	V'i:s

3.4.1. Data Sets used

Devanagari number / numbers between [0 - 100] appearing in any sequence and the following Konkani sentences:

(i) मात शारांचें वातावरण तिजेर परिणाम करूंक पावलें नाशिल्लें. (Maat Sharache Vatavaran Tijer Parinam Karunk Pavle Nashil.).

(ii) रमाबायचें लग्न भुरगेपणांतूच जाल्लें. (Ramabaiche Lagn Bhurgepanatuch Jalle.).

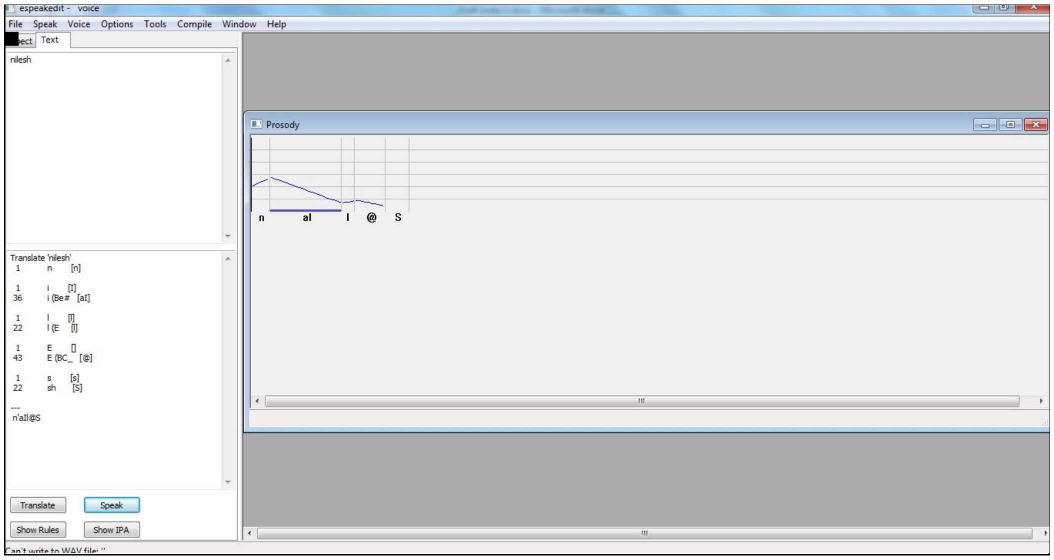
(iii) देवळ्ळालागीं घर जाल्ल्यान येतो-वतो आशिल्लोच. (Devlalagi Ghar Jallyaan Yeto-Vato Aashilloch.).

(iv) आनी एके बाबतींत तिची कुचमण जाताली. (Aani Eke Babtint Tichi Kuchman Jatali.).

(v) फकत आलायतीं कामां तिजेर पडटालीं. (Fakat Alaiti Kama Tijer Padtali.).

3.4.2. Experimental Results

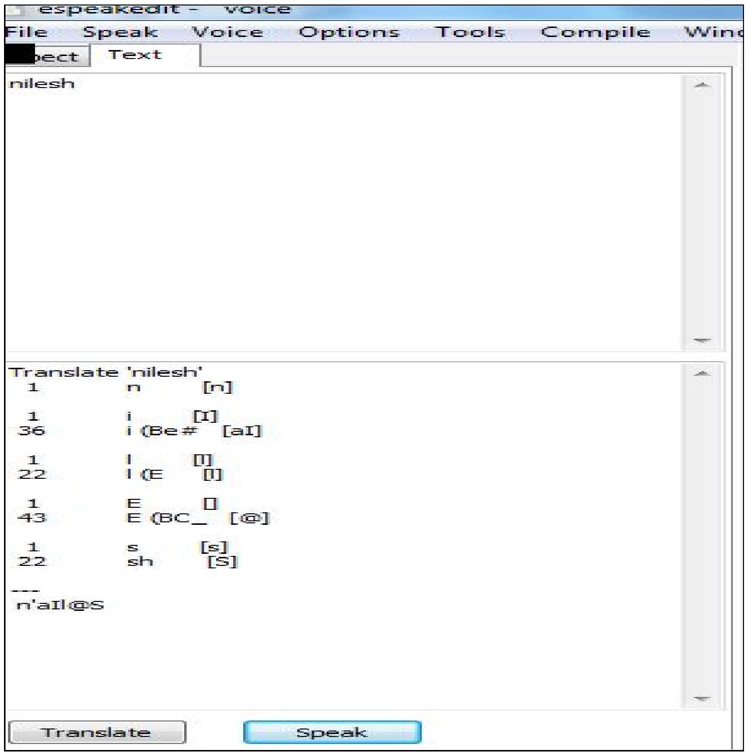
Fig. 3.1 and 3.2 shows the interface of eSpeakedit demonstrating phoneme definition along with its prosody for the sample word “nilesh”.


espeakedit - voice
File Speak Voice Options Tools Compile Window Help
ect Text
Prosody
n
aI
I
@
S
Translate
Speak
Show Rules
Show IPA
Can't write to WAV file: ''

```
nilesh
```

```
Translate 'nilesh'
  1        n       [n]

  1        i       [I]
 36        i (Be#  [aI]

  1        l       [l]
 22        l (E    [l]

  1        E       []
 43        E (BC_  [@]

  1        s       [s]
 22        sh      [S]

---
 n'aIl@S
```

Fig. 3.1 : Screenshot of eSpeakedit (1)



```
nilesh
```

```
Translate 'nilesh'
  1        n       [n]

  1        i       [I]
 36        i (Be#  [aI]

  1        l       [l]
 22        l (E    [l]

  1        E       []
 43        E (BC_  [@]

  1        s       [s]
 22        sh      [S]

---
 n'aIl@S
```

Fig. 3.2 : Screenshot of eSpeakedit (2)

3.5. Summary

We implemented a TTS prototype using the eSpeak tool to produce speech for Konkani numbers (0 - 100). The same can be further extended to implement more features of Konkani language using the existing Devanagari framework. It is observed that eSpeak produces robotic sounding speech lacking the required Indian accent and thus low on the required naturalness of a text to speech system.

Chapter 4

Concatenative Speech Synthesis

4.1. Introduction

Concatenative speech synthesis systems involve the selection of appropriate units from the speech inventory, algorithms that join the selected units and some signal processing so as to make the concatenation boundaries smooth. The speech database can have speech units of different sizes such as phones, diphones, syllables, words or sentences.

With an objective to improve the naturalness for Konkani language, the concatenative approach was adopted to develop a basic TTS system for Konkani.

4.2. Related Work

We reviewed the following papers relating to concatenative speech synthesis to understand the technical details:

(i) A TTS synthesis system based on phonetic concatenation, where the input text is first converted into phonetic transcription using Letter-to-Sound rules. The system on selection of the audio phoneme units (PUs) proceeds to modify its duration using spelling based rules. The modified PUs are then concatenated by synchronizing pitch-periods at juncture and smoothening the transitions in order to remove the audible discontinuity and spectral mismatches [52].

(ii) A method to design a Text to Speech conversion module using simple matrix operations of Matlab where the recorded sounds are sampled and the sampled values are separated into their constituent phonetics. The separated syllables are then concatenated to reconstruct the desired words [53].

(iii) A system using concatenative speech synthesis to transform Marathi text retrieved from a search engine into spoken words where the synthesizer is based on artificial neural networks and uses different sizes of speech units like words, diphones and tri-phones to produce speech [54].

(iv) Using a database of Konkani words, the system reads these words directly. The synthesizer used for the purpose uses concatenative technique for around 1000 words [13].

(v) Design and implementation of TTS system for English and Konkani language using image recognition technology (Optical Character Recognition) where a cost effective text image to speech conversion system is developed for Konkani using MATLAB. This involves pre-processing the text for recognition which is further segmented to separate out the characters from each other followed by extraction, resizing and storing in a text file. This text is then converted into speech [55].

4.3. The TTS Model

The approach used for the development of the TTS is depicted in the block diagram shown in Fig. 4.1.

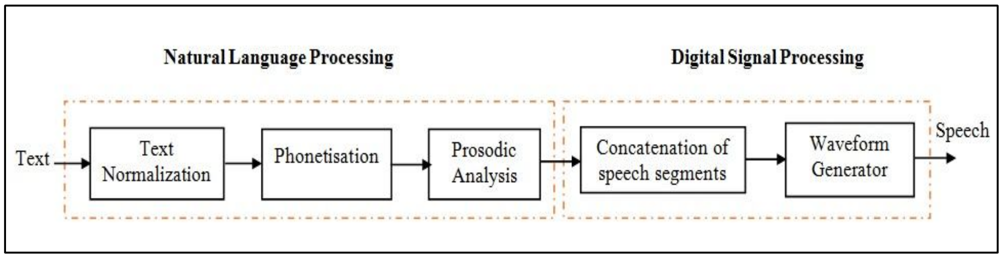

Natural Language Processing
Digital Signal Processing
Text
Text Normalization
Phonetisation
Prosodic Analysis
Concatenation of speech segments
Waveform Generator
Speech

Fig. 4.1 : Concatenative Speech Synthesis

Concatenative speech synthesis is used to generate user specific sequence of sounds from a database built by recording sounds. The recorded sound files database is saved in one of the directories. The input text is scanned and checked for numbers, syllabic, diaphonic and phonemic units in order. The Text parser does this, thus separating the units with the help of grapheme to phoneme rules. Each text unit is then linked to its respective sound file from the sound files database by providing the required linking path. Once this is done, the wave files found are concatenated by the concatenation synthesis algorithm thus giving out the output in the form of speech.

4.4. Implementation

The conversion of TTS is not a single step process and comprises of a number of stages [56]. The process flow of concatenative speech synthesis is depicted in Fig. 4.2. We have implemented the following stages: Splitting the Text (further split into 3 steps: split the token, type identifier and token expander), extracting pronounceable words [57], controlling the prosody [58] and voice generation.

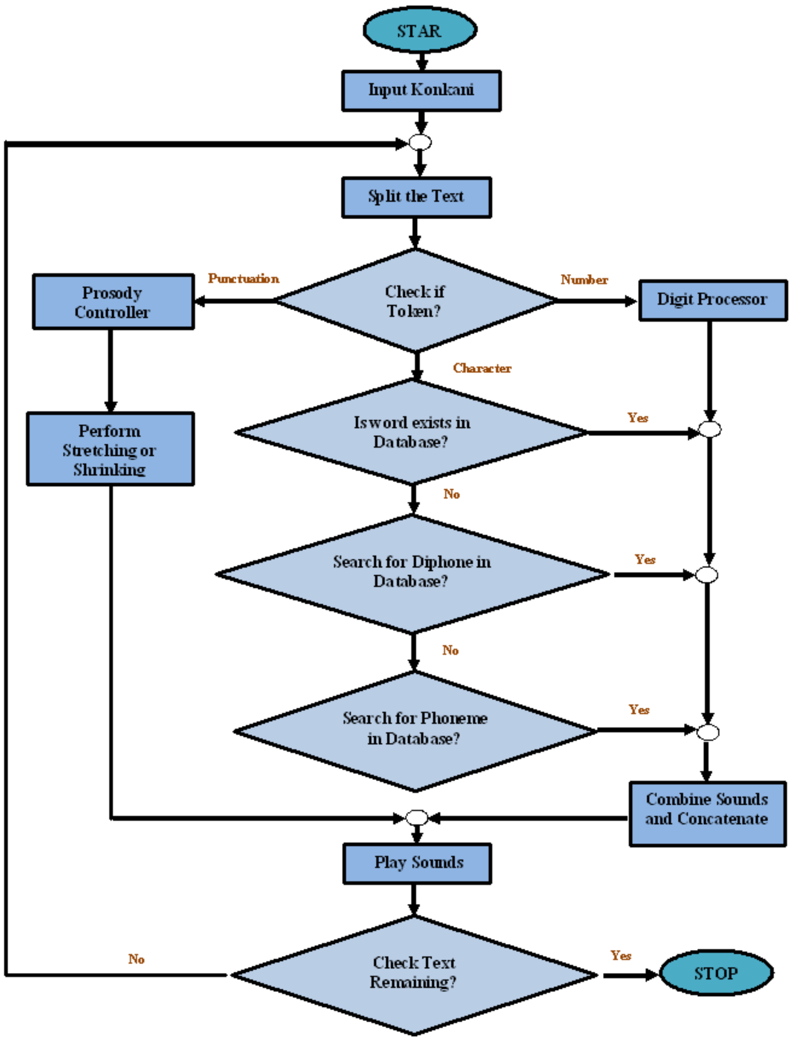


Fig. 4.2 : Flowchart of Concatenative Speech Synthesis

4.4.1. Split the Text

In text analysis the input text is converted to a Phonetic Transcription. The input text is a stream of characters, punctuations and non-standard words (NSW) like numbers, abbreviations, date, etc. Table 4.1 shows examples of non-standard words and their corresponding pronunciation [59]. This stage is further sub divided into 3 parts.

Table 4.1 : NSW Categories

Type of NSWs	**Examples**
Fraction	३/५
Ratios	३:५
Decimal numbers	३. ७५
Time	११.३५
Tele Phone no.	२२ २७२ २२
Date	२१\१२\१०१२
Percentage	३५.३७%
Abbreviations	डॉ. रू.

(i) Split the token (based on whitespace and punctuation): Whitespace is most commonly used between words and is extensively used for tokenization. One of the issues in tokenization is the end of utterance detection. This is done by detecting punctuations [comma (,), question mark (?), exclamation mark (!), etc.].

(ii) Type Identifier: Every token will then have to go through a token identification process that identifies its token type/category. Each NSW can be identified as a separate token by token identifier rules. It also involves identification of numbers in a language.

(iii) Token Expander: After identification of all the NSW, we can convert these to standard words by appropriate pronunciation characters. If the text encounters a number, then it is processed by a digit processor and the system places its corresponding sound unit. The list of characters is further split for each word and query is fired to the database.

4.4.2. Extracting Pronounceable Words

In this stage, standard input words are processed. If the text entered is a word then it is searched in the word database. If the word is not in the database then it is cut into diphones and the diphones are searched in the diphone database. If the corresponding diphone is not in the database then the word is formed by concatenating phonemes from the phoneme database and played [57].

4.4.3. Controlling the Prosody

Prosody plays a vital role for human interaction and therefore it is essential to include prosodic components into the speech-based systems. We have learnt that speech prosody is not only a component of linguistics, but forms a link to non-linguistic communication as well and above all, non-verbal communication modes like gesturing [58].

Prosody is implemented by inserting a silence between every word and sentence and by searching for the punctuations at the end of sentence. In our work, we have considered six punctuations that are frequently used as shown in Table 4.2.

Table 4.2 : Punctuation Controlling Prosody

Sr. No.	Name of Prosody	Symbol
1	Full Stop	.
2	Comma	,
3	Exclamation Mark	!
4	Question Mark	?
5	Colon	:
6	Double Inverted Comma	“ ”

Controlling prosody usually involves stretching and shrinking of wave signal (speech signal). The signal stretching and shrinking is illustrated in Fig. 4.3 with the help of an example Konkani word “kal (काल)”. Once the phoneme to speak is received by the engine, it loads the digital audio from the database, along with pitch, time and volume changes as output voice. A sample input sentence traced through different stages to produce speech output is illustrated in Fig. 4.4.

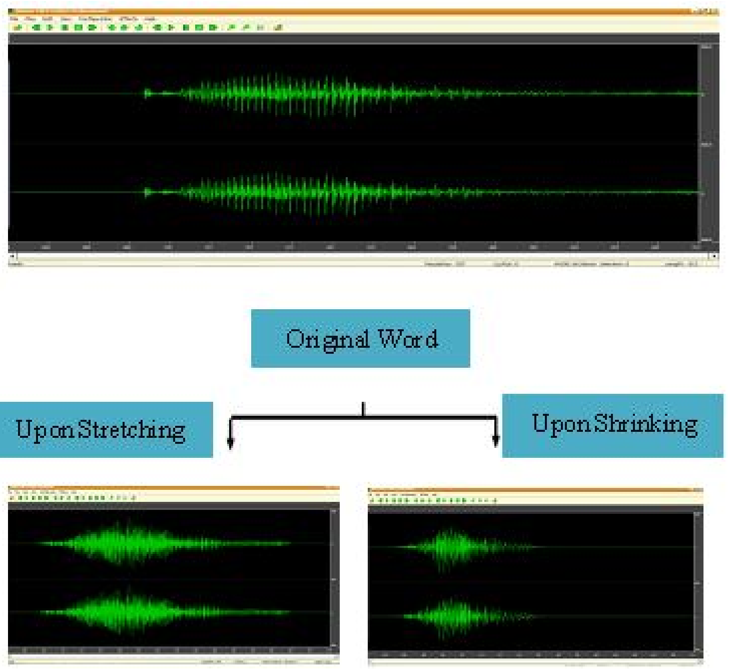


Fig. 4.3 : Original, Stretched and Shrinked Wave file

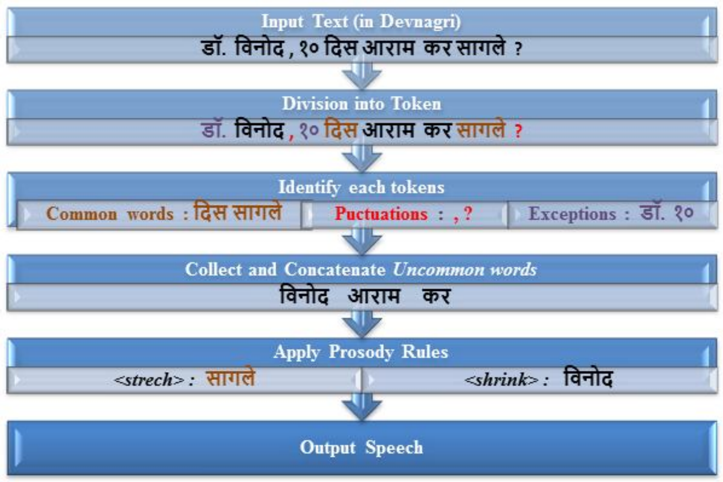
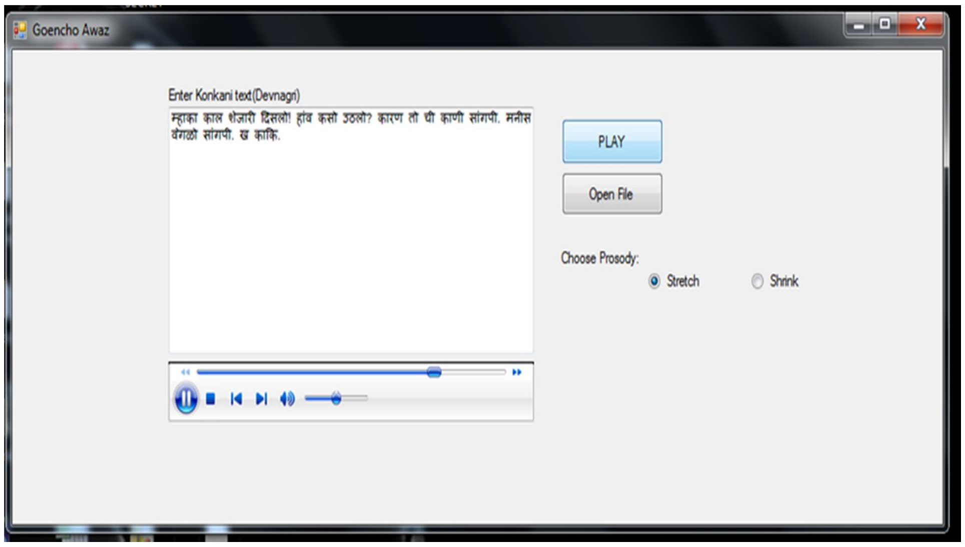


Fig. 4.4 : Sampled Input Sequence

4.4.4. Front End and Voice Data

The text processing Front-End is implemented in VISUAL BASIC.NET. The input text can be entered either through Konkani (Devanagari) enabled keyboard or by accessing the Devanagari text file. The TTS interface is shown in Fig. 4.5. The text input is either non-standard words or standard words.



म्हाका काल शेजारी दिसलो! हांव कसो उठलो? कारण तो धी काणी सांगपी. मनीस वेगळो सांगपी. ख काकि.

Choose Prosody: [✓ Stretch] Shrink

Fig. 4.5 : TTS Interface of Speech Synthesis

4.4.5. Sound Inventory

A female voice was used for recording voice in a laboratory environment to create the speech database. The recording was carried out with a sampling rate of 48000 Hz. Sample three and four letter words, diphones, phonemes and numbers (0-100) were recorded in .wav format as “CV”, “V”, “VC” and “0C” (half-consonant), where C represents consonant and V represents vowel. Considering 36 consonants and 12 vowels, there are 36*12 “CV” pairs, 12*36 “VC” pairs, 30 half sounds and around 200 words. The size of the recorded database is around 150 MB.

4.4.6. Data Sets used

The system was tested with the set of sentences that covered possible variations mainly: numerals, abbreviations & acronyms, symbols in Konkani along with the Konkani sentences listed at 3.4.1.

4.4.7. Experimental Results

A volunteer based evaluation was carried out on a set of sentences to compare the TTS system using concatenative synthesis with eSpeak for Konkani. The utterances were played to twenty native male and female speakers each of Konkani in the age group of 20 to 25 years who did not have any prior exposure to speech synthesis experiments.

The feedback of the volunteers for the TTS systems using eSpeak and concatenative synthesis in terms of the Mean Opinion Score (MOS) is depicted in Fig. 4.6.

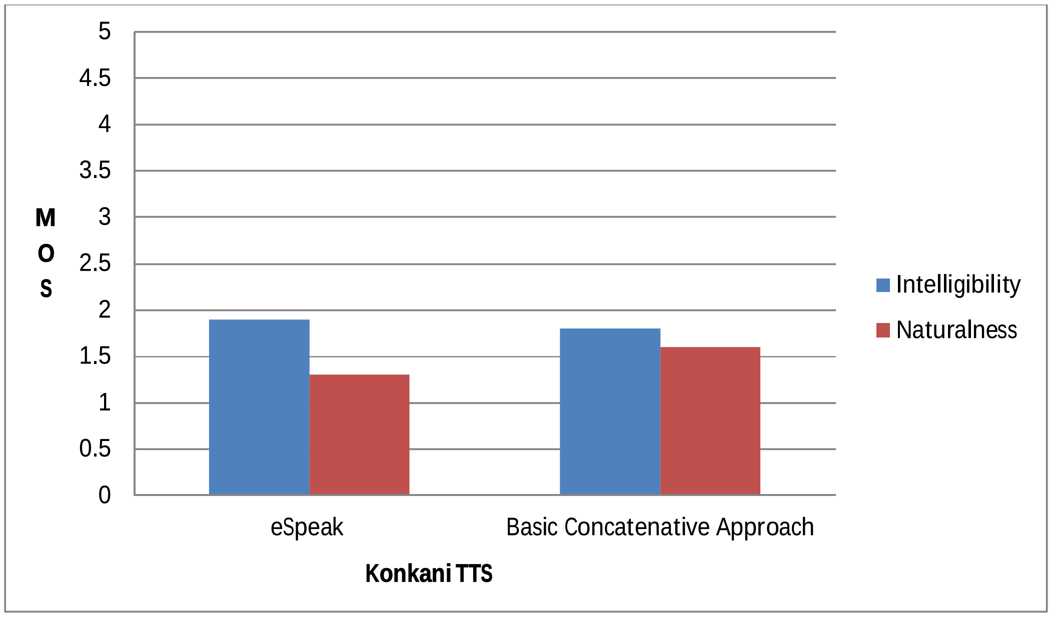


Fig. 4.6 : Feedback of the Volunteers for the TTS System using eSpeak and Basic Concatenative Approach

4.5. Summary

A basic concatenative TTS system was developed, implemented and tested for Konkani language using a small speech corpus created for the purpose. The tests indicate that the word units perform better than diphone or phoneme units. Based on the outcome of the evaluation, it is observed that the basic concatenative TTS system performs better than eSpeak for Konkani in terms of playing the natural human recorded sounds. This system can be used for a domain specific application by recording the domain specific voices.

Chapter 5

Overview of Syllable Based TTS Systems

5.1. Introduction

Rhythm and Intonation play an important role into making of the spoken component of a statement in a language, without which meaning remains ambiguous and is not conveyed to the listener properly e.g. interrogative statement versus a normal statement. The rhythm of the languages is classified as stress-timed (e.g. English, German, Russian, Arabic, Thai, Brazilian, Portuguese, Newari, Chepang, Gurung, Tamang) and syllable-timed (e.g. French, Spanish, Yoruba, Telugu, Hindi, Tamil, Indonesian, Japanese, Italian) patterns. Indian languages are syllable-timed, where there is a regular time interval between each syllable [59].

Syllabification algorithms have been proposed for various languages. According to the researchers and philologists in the domain of Indian languages, all Indian languages are phonetic in nature and thus they require a common syllabification algorithm. But practically, we find that each language has its own uniqueness and thus leading to independent work being carried out for the respective language.

Speech in Indian language is based on inherently syllable units. With different choices of speech units: syllable, diphone and phoneme, the syllable unit gives better results than the rest and thus better representation for Indian languages [61]. Study in this context refers that, there is no agreed definition of a syllable among linguist. It is observed that identifying syllables is comparatively easy, but difficult to consider syllables as being made up of segments (consonants and vowels). An approach to do this is by defining the syllable in terms of inherent ‘sonority’ of each sound. The sonority of a sound is its loudness relative to other sounds with the same length, stress and speech. By virtue of Sonority Theory of a sound, syllable is its loudness relative to other sounds with the same length, stress and speech [62].

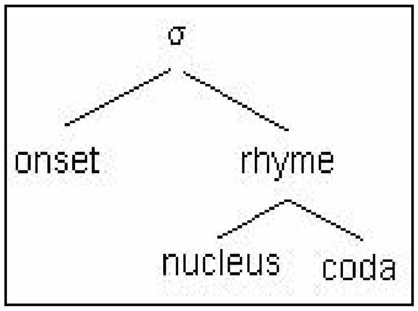
5.1.1. Syllable Structure

Syllable is a unit of spoken language consisting of a single uninterrupted sound formed generally by a vowel and preceded or followed by one or more consonants [63][64]. Syllables define the sequence of speech sounds. Words in a language can be divided into syllables (the word water is composed of two syllables: wa and ter*)*. Sonority theory distinguishes syllables on the basis of sounds. By theory of phonology, a syllable structure which is combination of allowable segments and typical sounds sequences is language specific.

Generally, a syllable (σ) consists of three segments: Onset (ω), Nucleus (ν) and Coda (κ) (refer Table 5.1).

Table 5.1 : Syllable Structure Description

Segments/Parts	**Symbol**	**Description**	**Optionality**
Onset	**ω/ on**	➢ Initial segment of a syllable. ➢ This sound(s) occurs before the nucleus and is usually consonant(s).	optional
Rhyme/Rime	P	➢ Core of a syllable. ➢ Consisting of nucleus and coda.	Obligatory
Nucleus	**ν / nu**	➢ Central segment of a syllable ➢ It is usually the vowel in the middle of a syllable.	Obligatory
Coda	**κ/co**	➢ Closing segment of a syllable. ➢ This sound(s) is literally 'tail') to the nucleus and is usually consonant(s).	optional



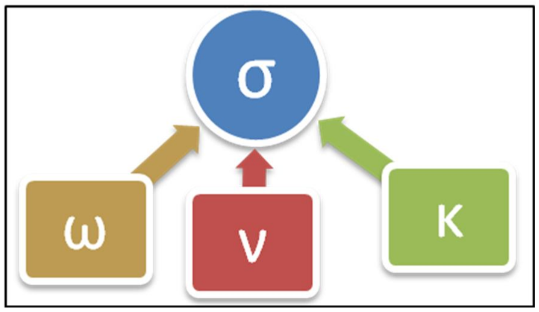


Fig. 5.1 : Structural Diagram of Syllable

Syllable structure is usually represented in phonetics as a tree diagram (refer Fig.5.1). Syllable is divided into onset (any consonants preceding the vowel) and rhyme (all phonemes from the vowel to the end of the syllable). The rhyme is further divided into the nucleus and coda.

Let us consider an example in English for the word “cat”. Here nucleus is a (the sound that can be shouted or sung on its own), the onset c, the coda t, and the rhyme at. This syllable can be abstracted as a consonant-vowel-consonant syllable, abbreviated CVC.

In the prior work on sonority theory, it was not possible to syllabify some words. To overcome this, linguists propose Maximal Onset Principle to determine syllable division. This phonological principle determines the placement of syllable boundaries in case of doubt. It states that intervocalic consonants are maximally assigned to the onsets of the syllable in conformity with universal and language-specific conditions. For example English word diploma can be divided in several ways: dip.lo.ma versus di.plo.ma. However, the only division that is in conformity with the maximal onset principle is di.plo.ma [65].

5.1.2. Syllabification

Syllabic patterns are made up of basic units of writing called Aksharas which consist of consonants and vowels. Generally, linguists define the syllable structure as C*VC*, where C is consonant and V is vowel. In other words, syllabic pattern can be of the form zero or more consonants followed by a vowel further followed by zero or more consonants. The rules of sonority theory are applied as technique to extract syllables i.e. we require to have an algorithm to split syllables from words in input text to synthesizer. Thus the partitioning of a word into syllables, whether in spoken or written text is referred to as Syllabification or Syllabication.

Syllable is widely used by many researchers in the domain of Automatic-Speech-Recognition (ASR) and Text-to-Speech (TTS) for their research work. A coarse classification of syllable in Indian language is a monosyllable (one syllable), a disyllable (two syllables) or a polysyllable (two or more syllables). Although syllable structure need not remain the same, we have considered monosyllables in our work.

5.2. Related Work

We reviewed the following syllable based TTS systems to understand the approaches used for building TTS for other languages.

5.2.1. Development of an Arabic Text-To-Speech System

This speech synthesizer is a combination of formant and Concatenation techniques [2]. The design of synthesizer is split into three phases: Pre-Processing phase, Natural Language Processing and Digital Signal Processing. Pre-Processing phase prepares raw text for processing i.e. it detects spaces, punctuation marks or non-Arabic symbols in each word of the sentence.

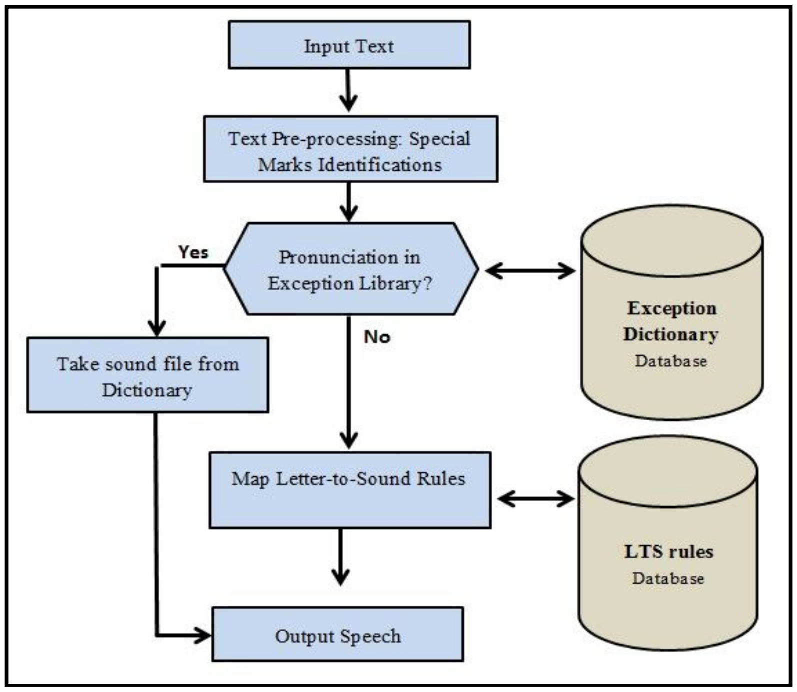

Input Text
Text Pre-processing: Special Marks Identifications
Pronunciation in Exception Library?
Yes
No
Take sound file from Dictionary
Exception Dictionary Database
Map Letter-to-Sound Rules
LTS rules Database
Output Speech

Fig. 5.2 : Arabic Speech Synthesizer

The Natural Language Processing (NLP) phase performs mapping of each word to its exact phonetic representation in three parts. As seen in Fig. 5.2, words are searched in the Exception Dictionary containing list of all words whose pronunciations are explicitly given rather than determined by the pronunciation rules in the first part. In the second part, Arabic pronunciation rules are applied. These rules specify phonemes which are used to pronounce each letter or sequence of letters. To find the pronunciation of the word, the rules are searched with the highest score depending on how many letters are matched. In the third part, all phonemes are defined in Arabic language. Words are applied with special patterns for syllable lexical stress. Finally in the Digital Signal Processing (DSP) phase, the resultant phonemic representation of input text with special stress is transformed into a proper utterance wave file.

5.2.2. Indonesian Text-to-Speech using Syllable Concatenation for PC-based Low Vision Aid

This TTS synthesizer uses syllable concatenation [16]. The list of all syllables required to make all words in Bahasa (Indonesian national language) are recorded with the same characteristics.

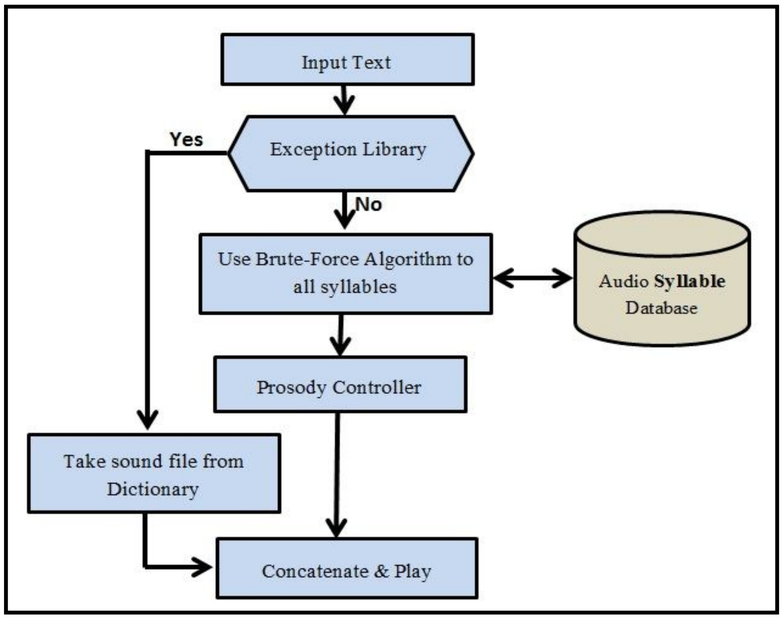


Fig. 5.3 : Indonesian Speech Synthesizer

The input special terms are handled by exception library and numeric characters are normalized to their sounds. The input word is seen as a list of characters wherein, the word is processed to obtain the proper combination of syllable. i.e. first the system will check whether there are reserved syllable patterns for the particular word. If reserved pattern is available, then the system will use that pattern. Otherwise, the system uses brute force algorithm to obtain the syllables required to form the word. This is followed by matching the word by its longest pattern and checking for its corresponding sound in the library. If no matching sound is available, then check for shorter pattern. This algorithm is repeated until the word is processed as shown in Fig. 5.3.

In this synthesizer the prosody is generated by looking for punctuation at the end of the sentence. In this study, we concentrate on two general punctuation that are commonly used: dot (.) and question mark (?). If a sentence ends by a dot (.), then the pitch of the syllable before the mark is lowered by a semi tone. If it ends by a question mark (?), then the pitch is raised by a semi tone.

5.2.3. Marathi Language Speech Synthesis using Concatenative Synthesis Strategy

This paper presents a Concatenative speech synthesis technique for Marathi language [57] using different choice of units: words, syllables and phonemes as shown in Fig. 5.4.

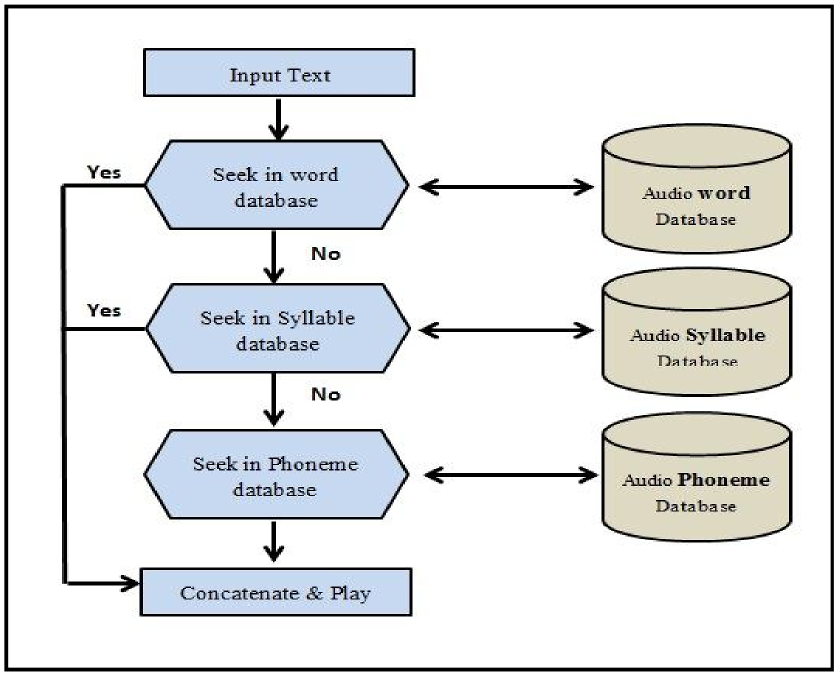


Fig. 5.4 : Marathi Speech Synthesizer

The text input is either a standard word or a non-standard word. If the input text is a number then it is handled by a digit processor. The TTS system is able to read any dates, addresses, telephone numbers and bank account numbers. If input text is a word then it searched in the word database. If the word does not exist in the database then it is cut into syllables and syllables are searched in the syllable database. If the corresponding syllable does not exist in the database then the word is formed by concatenating phonemes in the phoneme database and played.

5.2.4. Text To Speech Synthesis System for Punjabi Language

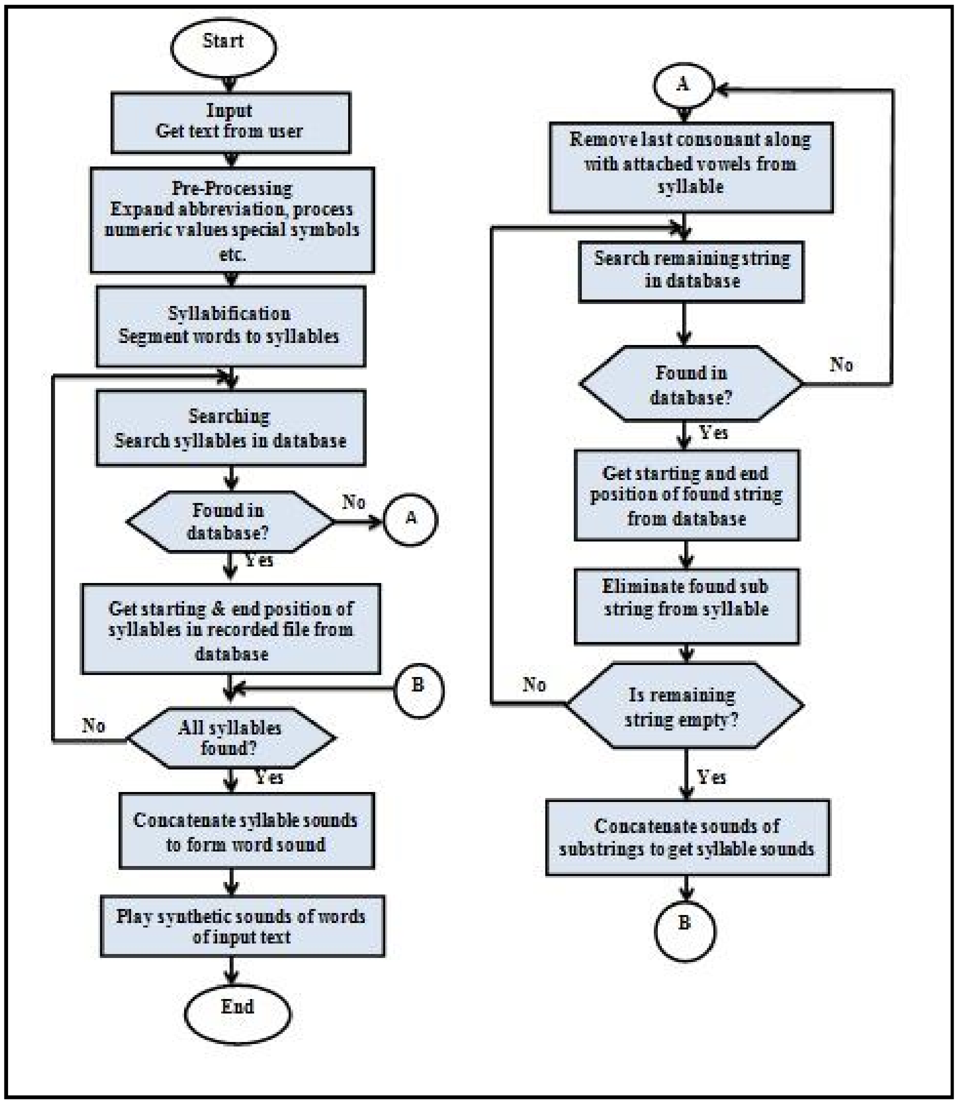

Start
Input
Get text from user
Pre-Processing
Expand abbreviation, process numeric values special symbols etc.
Syllabification
Segment words to syllables
Searching
Search syllables in database
Found in database?
No
A
Yes
Get starting & end position of syllables in recorded file from database
B
No
All syllables found?
Yes
Concatenate syllable sounds to form word sound
Play synthetic sounds of words of input text
End
A
Remove last consonant along with attached vowels from syllable
Search remaining string in database
Found in database?
No
Yes
Get starting and end position of found string from database
Eliminate found sub string from syllable
No
Is remaining string empty?
Yes
Concatenate sounds of substrings to get syllable sounds
B

Fig. 5.5 : Punjabi Speech Synthesizer

The TTS system for Gurmukhi Punjabi language uses a syllable based concatenation approach of speech synthesis [66]. The system uses a speech corpus which contains labelled syllables specifying the beginning and end position in the file. The syllabification algorithm developed generates syllables based on grammatical rules as shown in Fig. 5.5. The output speech is generated in this TTS system by searching the appropriate sound in the speech database. The quality of the output speech depends on how carefully the speech units are labeled in the recorded sound file.

5.2.5. Text to Speech Synthesis System for Tamil Language

This system uses a concatenative -based approach to synthesis [67] wherein, concatenation happens at the word level and syllable level. As shown in Fig. 5.6, the following stages are used to convert Tamil text to speech: Text Normalization, Sentence Splitting, Speech corpus, Concatenation and Speech synthesis.

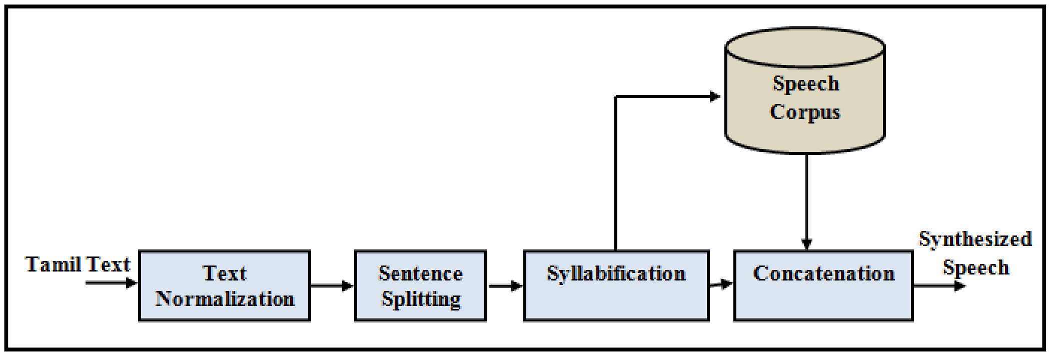


Fig. 5.6 : Tamil Speech Synthesizer

In text normalization, punctuations such as double quotes, full stop, comma, etc. are all eliminated to obtain plain sentences. In sentence splitting, the given paragraph is split into sentences using white spaces as delimiter. The quality of synthesized speech waveform depends on the number of realization of various units present in the speech corpus. The concatenation of speech files is carried out as the final stage process in MATLAB.

The speech synthesis follows two approaches. First, word level synthesis; when all the words in the input text are already present in the speech corpus and hence the synthesized output naturalness is very high. Second, syllable level synthesis; when the input word is not present in the speech corpus, the word is synthesized using syllable level concatenation and hence naturalness drops in comparison to word level synthesis.

5.2.6. Text To Speech System for Telugu Language

This system uses Concatenative method of synthesis with syllables as the basic unit [68]. The process of speech synthesis starts with text processing where the input Telugu text is converted to Unicode irrespective of the platform and size. This is followed by differentiating Graphemes and phonemes in order to provide the correct output speech signal to the user as shown in Fig. 5.7.

Phonemes can be further differentiated on the basis of CV structure (consonant & vowel**)**. The input text it is made to go through the process of prosodic modeling and acoustic synthesis so as to generate the correct audible accent for the output speech. Selection of a syllable unit is carried out on the basis of preceding and succeeding context of the syllables and the position of the syllable.

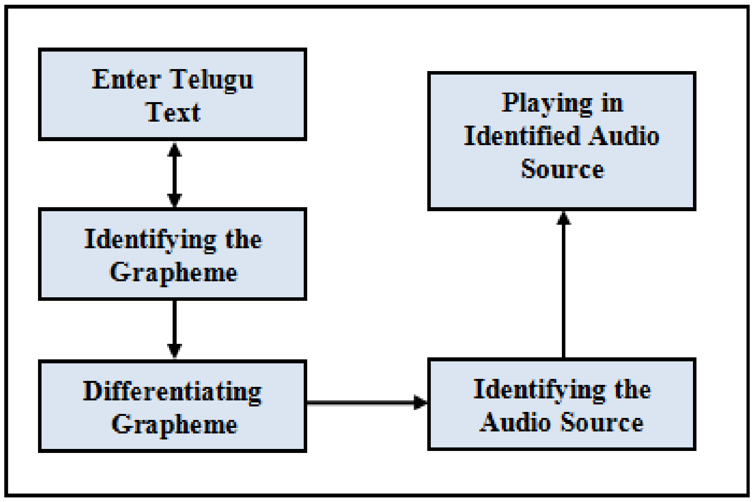


Fig. 5.7 : Telugu Speech Synthesizer

5.3. Comparison of Syllable based TTS Systems

The objective of this study is to understand the syllabification methods used by TTS synthesizers to generate speech. In general the TTS systems use a variety models, techniques and tools to suite their requirements. In our work for the purpose of comparison we have considered systems using similar synthesis techniques for development.

The basis for comparison considered is nature of the script, syllable pattern, prosody modelling.

5.3.1. Parameters for Comparison

The above reviewed speech synthesizers were examined to draw comparative inferences considering the following parameters of a syllabic speech synthesizer: Front end processing, Symbolic prosody control, Syllable patterns distribution, grapheme to phoneme conversion [69].

5.3.1.1. Front End Processing

It refers to the first step wherein raw text (such as Non-standard words) requiring normalization are transformed into pronounceable words [60]. After tokenizing and token identification, expansion rules are applied to resolve ambiguity and phrase detection.

5.3.1.2. Symbolic Prosody Control

Prosody is also referred to as supra-segmental features and deals with two major factors:

(i) Breaking the sentence into prosodic phrases, possibly separated by pauses.

(ii) Assigning labels, such as emphasis to different syllables or words within each prosodic phrase.

Words are normally spoken continuously, unless there is a specific linguistic reason to signal discontinuity. The term juncture refers to prosodic phrasing i.e. where words cohere, and where prosody breaks (pauses and/or special pitch movements occur).

5.3.1.3. Syllable Patterns Distribution

Syllabification is a process by which sound units are produced corresponding to the syllables. It is based on the syllable distribution patterns in the language.

Common supportive syllable patterns for Indian languages are V, CV, CCV, VC, and CVC. Given general patterns of syllable, they exhibit differently based on their language origin.

5.3.1.4. Grapheme to Phoneme Conversion

There are three approaches available for text to phoneme conversion namely, dictionary based, rule based and lexical accent based [69]. A dictionary based approach implements a large dictionary. Like a dictionary it will contain containing all the words of a language along with their correct Phonetic Transcriptions. Although this approach is quick and accurate, it fails if look- up word is not present in the dictionary. In the rule based approach, pronunciation rules are applied to the words to determine their pronunciations based on their spellings. In the lexical accent approach, the type of pronunciation or sound of a word depends on its context.

5.3.2. Observations

In syllable based speech synthesis, while forming a new word, the position of the syllable is considered in-order to apply proper rules of concatenation to generate a more natural output. A brief comparison of the speech synthesis techniques for various TTS [2] [16] [57] [66] [67] [68] is shown in the Table 5.2. It is observed that most Indian languages use syllable based approach to speech synthesis.

Based on analysis of mythologies followed in each of the paper following are the inference drawn from the above study:

(i) Indian script features the phonetic nature. There is more or less one to one correspondence between what is written and what is spoken.

(ii) Syllables are best suitable units for Indian Synthesizer implementation.

(iii) The Basic Indian syllable patterns are recognized by each candidature algorithm. Typically patterns are C, V, CV and CVC, where C is a consonant and V is a vowel. But there also exist additional patterns.

(iv) There is no special need of prosody modeling when syllables are used as patterns.

Table 5.2 : Comparison of Syllable Based Candidate TTS Systems

Parameter	**TTS systems**					
	Arabic Speech Synthesizer	***Indonesian Speech Synthesizer***	***Marathi Speech Synthesizer***	***Punjabi Speech Synthesizer***	***Tamil Speech Synthesizer***	***Telugu Speech Synthesizer***
Languages Supported	Arabic	Bahasa	Marathi	Punjabi (Gurmukhi)	Tamil	Telugu
Front-End Processing	Detects spaces, punctuations and other non-Arabic symbols	Detects units of physics, length, mass, etc. Handled by Exceptional Library.	Detects numbers, dates, address, telephone nos. & bank account nos.	abbreviations, numeric values, special symbols are analyzed	remove all types of punctuations and process pure sentences	No specific pre-processing are considered
Type of Approach	Rule Based with Dictionary Based supportive	Dictionary Based	Rule Based	Rule Based	Rule Based	Rule Based
Choice of unit for Synthesis	Syllables with words	Syllables	Syllable with words and phonemes	Syllables	Words and Syllables	Syllables
Syllable Patterns	CV, CVV, CVC, CVCC, CVVC, CVVCC	V, VC, VCC, CV, CVC, CVCC, CCV, CCVC, CCVCC, CCCV, CCCVC	C, V, CV, CCV, VC, CVC	V, VC,CV, VCC, CVC, CCVC, CVCC	V, VC,CV, VCC, CVC, CCVC, CVCC, CC, CCV	C, V, CV, CCV, CVC
Prosody	Performs lexical stress at syllables	Pitch control by semi tone raised/lowered for punctuations	No specific prosodic features are considered	No specific prosodic features are considered	No specific prosodic features are considered	Prosody considered while implementing

5.4. Summary

Each of the strategic TTS systems studied shows its importance for the specific language for which it is designed. Syllable based concatenative approach is most suitable for Indian languages as Indian languages are phonetic in nature.

Chapter 6

Syllabification Algorithms for Devanagari Languages

6.1. Introduction

For a TTS synthesizer, the basic unit of synthesis is phoneme, diphone, syllable, word, phrase or even sentence. Table 6.1 outlines a brief comparison of the unit of synthesis with respect to concatenation points, database size and quality of speech generated. It is observed that with the increase in the unit size there is an increase in the database size, decrease in the concatenation points and improvement in the produced speech quality.

Table 6.1 : Comparison of the Unit of Synthesis.

Parameters	**Unit of Synthesis**		
	Phonemes/ Diphones	***Syllables***	***Words***
Concatenation points	Large	Moderate	Few
Database size	Small	Moderate	Large
Quality of Speech Generated	Good	Better	Excellent

6.2. Related Work

We reviewed the following work in the context of syllable based concatenative speech synthesis:

(i) A TTS system using concatenative synthesis techniques for Marathi language with unit selection speech databases discusses the issues when using words, phonemes and syllables as the unit of speech synthesis for Marathi Language [70].

(ii) In a Marathi Text-to-Speech system, this work is about a prosodic modification to generate expressive synthetic speech, where the input speech is a neutral speech. The major focus of this work is the modification technique and keeping the acceptable rate and naturalness with a reduced database. A method to modify fundamental frequency contour is proposed for Marathi TTS system. The emotion content in the speech is highly correlated to phonetic description and prosodic features such as fundamental frequency and time duration of that phone. Speech modification is carried out in time domain by a process of compressing the speech signal in time domain without changing its spectral features. For prosody generation, a primary pitch curve for the word is obtained based on the location followed by punctuation marks. Question mark and exclamation mark in the text are studied to modify prosody. TD-PSOLA (Time Domain Pitch Synchronous Overlap and Add) algorithm can improve the prosody of the synthesized speech. Prosodic changes can be possible by modifying different parameters like pitch level, pitch contour and pitch variation [71].

6.3. The TTS Model

Based on the study of the syllable based TTS systems, we use a combined top-down approach for the development of a TTS system using different choice of units: words (recorded), syllables (recorded) and syllables (formed using syllabification algorithm as shown in Fig. 6.1.

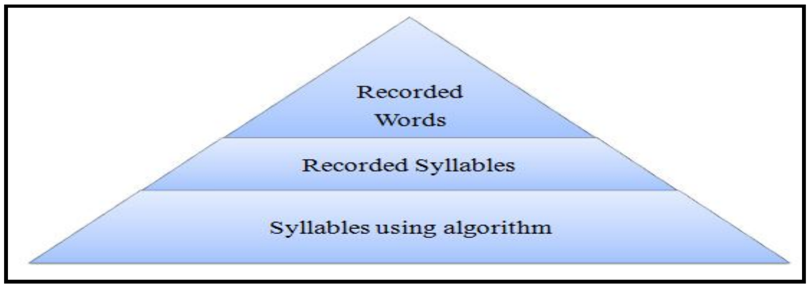


Fig. 6.1 : Top-down Flow to Play Input Text.

The input word is first searched in the database and played if present. If the word does not exist in the database then it is cut into syllables then syllables are searched in the recorded syllable database and played if present. If the corresponding syllable does not exist in the database then syllable is formed using syllabification algorithm by concatenating phonemes / diphones from the database and played as shown in Fig. 6.2.

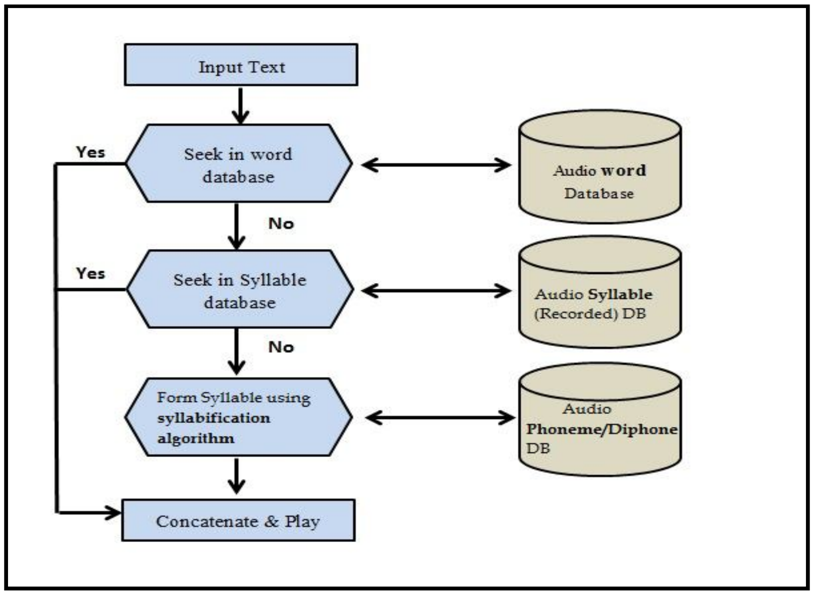


Fig. 6.2 : Flowchart for Speech Generation for Input Text.

6.4. Syllable Generation

In the preceding sections, we discussed some of the TTS systems in the context of Indian and non-Indian languages. It is observed that there are common patterns of syllables in Indian languages.

Syllables can be formed scanning words either from left to right or from right to left. Most of the languages are written / spoken from left to right and literature speaks of the forward approach and backward approach. Also study of few other TTS algorithms reveal that better syllabification can be obtain through the backward approach, even though that language is written and spoken form left to right [72].

6.4.1. The Backward Approach

The backward syllabification approach can be well understood in three phases. In phase I, it scans each token from right to left i.e. from rightmost character to the leftmost character (Refer Fig. 6.3). If the character to be traversed is ‘C’, i.e.

consonant, then it follows to the phase II (Refer Fig. 6.4). On encounter of specific symbols, it increments the counter of symbols scanned. Phase II can identify patterns of the type CC, CVC, VC and C. Similarly if the character to be traversed is ‘V’, i.e. vowel, then it follows to the phase III (Refer Fig. 6.5). On encounter of specific symbols, it increments the counter of symbols scanned. Phase III can identify patterns of the type CV and V.

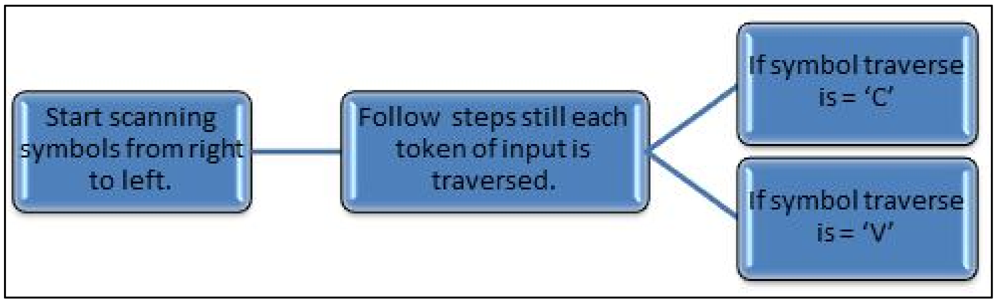

Start scanning symbols from right to left.
Follow steps still each token of input is traversed.
If symbol traverse is = ‘C’
If symbol traverse is = ‘V’

Fig. 6.3 : Backward Syllabification approach Phase I

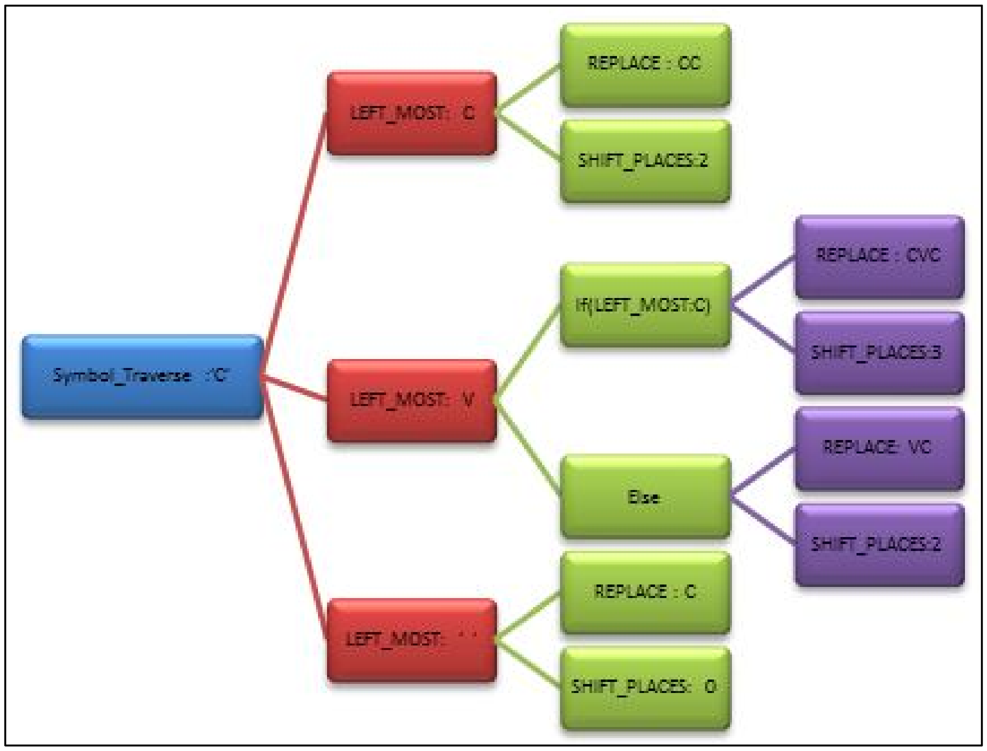


Fig. 6.4 : Backward Syllabification Approach Phase II

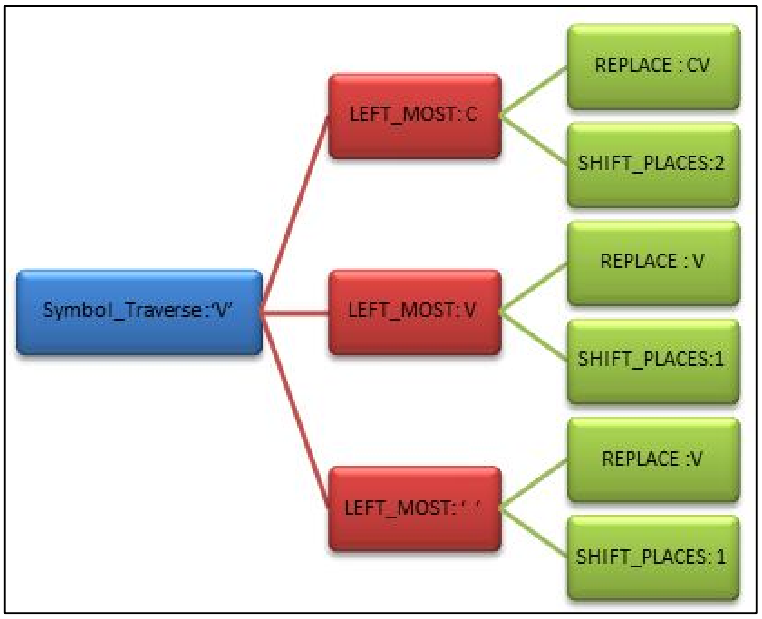


Fig. 6.5 : Backward Syllabification Approach Phase III

The syllabification approach recognizes syllable patterns of the form V, C, CV, VC, CVC and CC for Konkani. Thus the output will be string of Cs &Vs. The syllables of the type 'C' and 'CC' are having presence of inherent schwa present with each consonant [14].

6.4.2. The Forward Approach

In order to compare and implement the system for forward approach of syllabification, the processing of the steps is similar to backward approach, except that here we scan the characters from left to right. The backward approach can be easily understood from the flow diagrams shown in Fig. 6.6, 6.7 and 6.8 respectively.

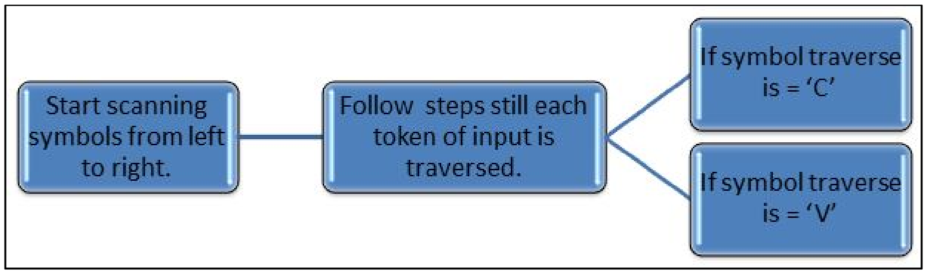

Start scanning symbols from left to right.
Follow steps still each token of input is traversed.
If symbol traverse is = 'C'
If symbol traverse is = 'V'

Fig. 6.6 : Forward Syllabification Approach Phase I

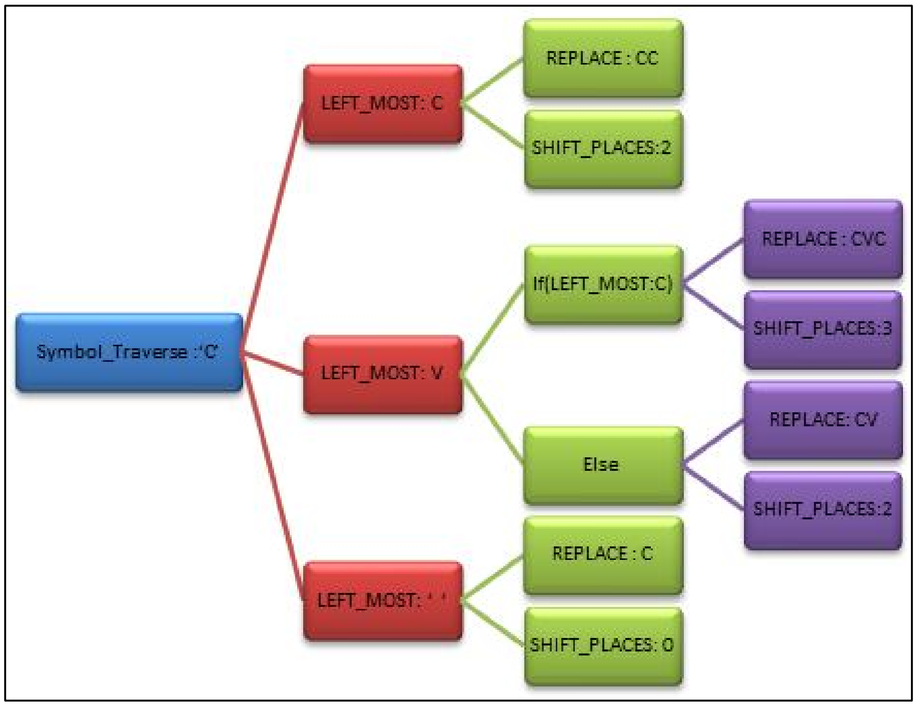


Fig. 6.7 : Forward Syllabification Approach Phase II

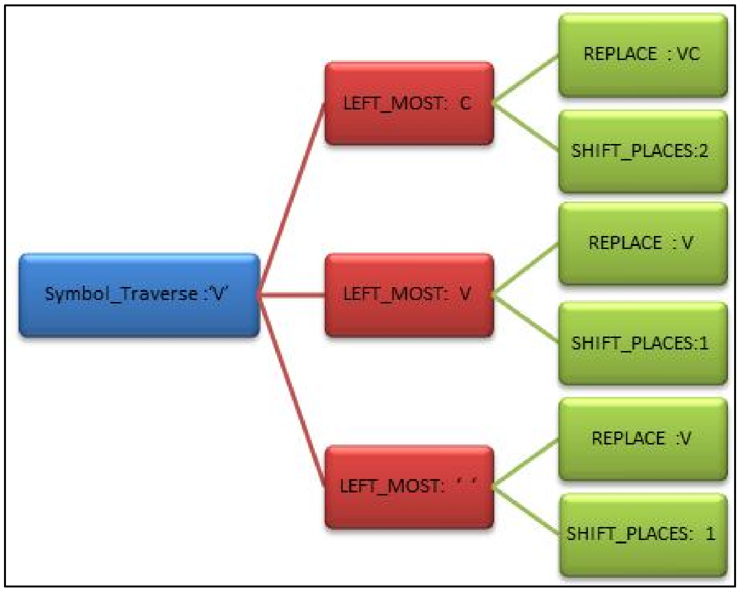


Fig. 6.8 : Forward Syllabification Approach Phase III

6.5. Testing Syllabification

In this section we manually trace some Konkani words to generate syllables using the forward and the backward approach to test the syllabification process.

6.5.1. The Backward Trace

We test the backward syllabification algorithm for Konkani text to speech conversion for simple words; “रमाबाय” (Ramabai), “परत” (Parat) and “बारकायेन” (Barkayen). The sound classes are successfully syllabified (syllable types) as shown in Table 6.2, 6.3 and 6.4.

Table 6.2 : Backward Syllabification for the word “रमाबाय”

Input String	Syllable Found	Description of Rules Applied
CCVCV$\underline{\text{C}}$	**CVC**	The symbol traversed is ‘C’, its left most sound unit is ‘V’, so we further traverse. Symbol traversed left most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places left.
CC$\underline{\text{V}}$CVC	**CV**	The symbol traversed is ‘V’ and its left most symbol is ‘C’, hence a class of “CV” will be formed. Pointer will be shifted by 2 places left.
$\underline{\text{C}}$CVCVC	**C**	The symbol traversed is ‘C’, and left most is the only character remaining and a class of “C” will be formed. Pointer will be shifted 1 place left.
रमाबाय **CCVCVC**	$\Rightarrow$	**र – मा – बाय** **C - CV – CVC**

Table 6.3 : Backward Syllabification for the word “परत”

Input String	Syllable Found	Description of Rules Applied
CC$\underline{\text{C}}$	**CC**	The symbol traversed is ‘C’, its left most sound unit is ‘C’, so a class of “CC” will be formed. Pointer will be shifted 2 places left.
$\underline{\text{C}}$CC	**C**	The symbol traversed is ‘C’ is the only character remaining and a class of “C” will be formed. Pointer will be shifted 1 place left.
परत **CCC**	$\Rightarrow$	**प - रत** **C - CC**

Table 6.4 : Backward Syllabification for the word “बारकायेन”

Input String	Syllable Found	Description of Rules Applied
CVCCVCV$\underline{\text{C}}$	CVC	The symbol traversed is ‘C’, its left most sound unit is ‘V’, so we further traverse. Symbol traversed left most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places left.
CVCC$\underline{\text{V}}$CVC	CV	The symbol traversed is ‘V’ and its left most symbol is ‘C’, hence a class of “CV” will be formed. Pointer will be shifted by 2 places left.
CV$\underline{\text{C}}$CVCVC	CVC	The symbol traversed is ‘C’, its left most sound unit is ‘V’, so we further traverse. Symbol traversed left most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places left.
बारकायेन **CVCCVCVC**	$\Rightarrow$	**बार - का - येन** **CVC - CV – CVC**

6.5.2. The Forward Trace

On similar lines, we test the forward syllabification algorithm for Konkani text to speech conversion for the same words; “रमाबाय” (Ramabai), “परत” (Parat) and “बारकायेन” (Barkayen). The results are shown in Table 6.5, 6.6 and 6.7. It is observed that the Konkani words “रमाबाय” (Ramabai), and “परत” (Parat) are not syllabified correctly while the word “बारकायेन” (Barkayen) is syllabified correctly.

Table 6.5 : Forward Syllabification for the word “रमाबाय”

Input String	Syllable Found	Description of Rules Applied
$\underline{C}$CVCVC	**CC**	The symbol traversed is ‘C’, its right most sound unit is ‘C’ hence a class of “CC” will be formed Pointer will be shifted by 2 places right.
CC$\underline{V}$CVC	**VC**	The symbol traversed is ‘V’ and its right most symbol is ‘C’, hence a class of “VC” will be formed. Pointer will be shifted by 2 places right.
CCVC$\underline{V}$C	**VC**	The symbol traversed is ‘V’ and its right most symbol is ‘C’, hence a class of “VC” will be formed. Pointer will be shifted by 2 places right.
रमाबाय CCVCVC	$\Longrightarrow$	**रम - ◌ाब – ◌ाय CC - VC – VC**

Table 6.6 : Forward Syllabification for the word “परत”

Input String	Syllable Found	Description of Rules Applied
$\underline{C}$CC	CC	The symbol traversed is ‘C’, its right most sound unit is ‘C’, so a class of “CC” will be formed. Pointer will be shifted 2 places right.
CC$\underline{C}$	C	The symbol traversed is ‘C’ to the right is the only character remaining and a class of “C” will be formed. Pointer will be shifted 1 place right.
परत CCC	$\Longrightarrow$	**पर - त CC - C**

Table 6.7 : Forward Syllabification for the word “बारकायेन”

Input String	Syllable Found	Description of Rules Applied
$\underline{C}$VCCVCVC	**CVC**	The symbol traversed is ‘C’, its right most sound unit is ‘V’, so we further traverse. Symbol traversed right most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places right.
CVC$\underline{C}$VCVC	**CVC**	The symbol traversed is ‘C’, its right most sound unit is ‘V’, so we further traverse. Symbol traversed right most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places right.
CVCCV$\underline{C}$VC	**VC**	The symbol traversed is ‘V’, its right most sound unit is ‘C’, so a class of “VC” will be formed. Pointer will be shifted 2 places right.
बारकायेन CVCCVCVC	$\Longrightarrow$	**बार - का - येन CVC - CV – CVC**

Around 50 more Devanagari words were manually traced using the forward and backward approach to generate syllables in order. It is observed that in the manual tracing process, the backward approach generated correct syllables for all the considered words whereas the forward approach did not syllabify all the words.

6.6. Implementation

To validate the manual tracing results for Konkani words, we have implemented both the backward and forward approaches, which are explained in the following sections.

6.6.1. The Backward Syllabification Algorithm

The Finite State Automata for the backward Syllabification as shown in Fig. 6.9 demonstrates the syllabification process where each node represents the state and C and V stands for consonant and vowel respectively.

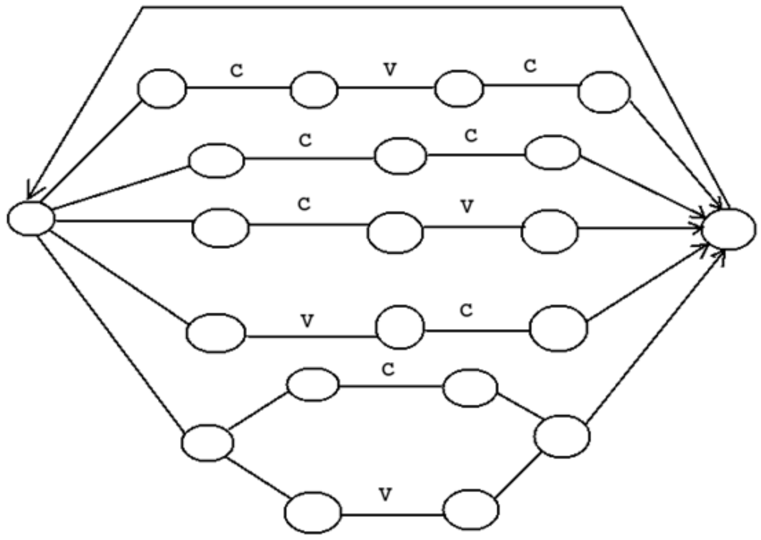


Fig. 6.9 : Finite State Automata for Backward Syllabification

The regular expressions for the finite state automata which correspond to the backward syllabification algorithm are shown in Table 6.8.

Table 6.8 : Regular Expressions for the Backward Syllabification Patterns

Sr. No.	Regular Expression
1	L1 = { CVC } +
2	L2 = { CC } +
3	L3={CV}+
4	L4={VC} +
5	L5 = {C+ V}+

Therefore L = ∑ {L1 + L2 + L3 + L4 + L5}, defines the language L For the regular expressions above which will recognize all the patterns during the syllabification process. The backward syllabification algorithm is detailed as below:

Backward_ Syllabification_Algorithm (Word, Syllable)

```
Input :- Word
Output :- Syllable

i=1;
While ( i < len (Word)
{
if( Word[i] == 'C' && Word[i+1] == 'V' && Word[i+2] == 'C')
{
i= i+3 ;
Syllable=Word[i]+Word[i+1]+Word[i+2];
}
if( Word[i] == 'C' && Word[i+1] == 'C')
{
i= i+2 ;
Syllable=Word[i]+Word[i+1];
}
if( Word[i] == 'V' && Word[i+1] == 'C')
{
i= i+2 ;
Syllable=Word[i]+Word[i+1];
}
if( Word[i] == 'C' || Word[i] == 'V')
{
i= i+1 ;
Syllable=Word[i];
}
}
return Syllable;
```

First the text is pre-processed to remove punctuations. We then pass the string of ‘C’ and ‘V’ patterns to the function for syllabification. The patterns are recursively traversed in reverse order. Further, the text file is processed in order to find the word present. Assuming ‘ch’ to hold each character read and ‘ChCount’ to hold the number of characters already traversed, the steps involved in detecting the syllables are as follows:

1. If word length is greater than 3 then spilt the input sting.
2. Examine to trace the longest consonant and vowel pattern.

a. Case 1 (for CVC): If the word length detected is 3 i.e. ChCount = 3, then the algorithm compares ch(i) = ‘C’, ch(i+1) = ‘V’ and ch(i+2) = ‘C’, increment the pointer by 3 and extract the corresponding audio file and concatenate.
b. Case 2 (for CC): If the word length detected is 2 i.e. ChCount = 2, then the algorithm compares ch(i) = ‘C’ and ch(i+1) = ‘C’, increment the pointer by 2 and extract the corresponding audio file and concatenate.
c. Case 2 (for CV): If the word length detected is 2 i.e. ChCount = 2, then the algorithm compares ch(i) = ‘V’ and ch(i+1) = ‘C’, increment the pointer by 2 and extract the corresponding audio file and concatenates.
d. Case 3 (for VC) If the word length detected is 2 i.e. ChCount = 2, then the algorithm compares ch(i) = ‘C’ and ch(i+1) = ‘V’, increment the pointer by 2 and extract the corresponding audio file and concatenate.
e. Case 4 (for C or V): If the word length detected is 1 i.e. ChCount = 1, then the algorithm compares the input string ch = ‘V’ or ch = ‘C’, increment the pointer by 1 and extract the corresponding audio file and concatenate.

The above steps are repeated for the whole text.

6.6.2. The Forward Syllabification Algorithm

In the preceding subsection we have discussed the backward syllabification algorithm. The implementation approach for forward syllabification algorithm is similar. In order to implement the forward approach for syllabification, the steps remain the same as followed for the backward approach; however the traversing of patterns is from left to right. The forward syllabification algorithm is detailed below:

Forward_Syllabification_ Algorithm (Word, Syllable)

```
Input :- Word
Output :- Syllable

i=len(Word);
While ( i >= 0)
{
if( Word[i] == 'C' && Word[i-1] == 'V' && Word[i-2] == 'C')
{
i= i-3 ;
Syllable=Syllable + Word[i]+Word[i-1]+Word[i-2];
}
if( Word[i] == 'C' && Word[i-1] == 'C')
{
i= i-2 ;
Syllable=Syllable+ Word[i]+Word[i-1];
}
if( Word[i] == 'V' && Word[i-1] == 'C')
{
i= i-2 ;
Syllable=Syllable + Word[i]+Word[i-1];
}
if( Word[i] == 'C' || Word[i] == 'V')
{
i= i-1 ;
    Syllable=Syllable + Word[i];
}
}
return Syllable;
```

6.6.3. Sound Inventory

In view of our specific requirement for our implementation, we recorded a speech of around 500 Konkani sentences in a studio environment. The recording was carried out with Sampling Rate: 48000 Hz, Bit Depth: 16 bit, Channels: Mono. These recorded sentences were cut into words and further into syllables of which we used around 2000 words and 500 syllables for testing. We also recorded voices for abbreviations (20), months (12), numbers (0 to 100), vowels (14), consonants (36), diphones (300) and nasal consonants (4) as part of the developed speech corpus. In addition, specific syllables were also recorded for the purpose of testing. Audacity tool and WaveSurfer tool was used to record, cut and label the sound files.

The ultimate goal is to have a large number of syllables from the available text. The text for recoding was selected keeping in mind the minimal presence of large polysyllables. For this purpose we have used the Konkani (Devanagari) text data available online at the Linguistics Data Consortium for Indian Languages (LDC-IL), Mysuru, Ministry of Human Resource Development (MHRD), Government of India [73] as well as short Konkani stories provided by the Konkani Linguists. Also, the text used contains very minimal non-Devanagari text and non-standard words [74].

The syllables in the words can be identified and processed manually or automatically. The cutting of the word into syllables must be very accurate to ensure correct results. A tool called SOUND FORG can be used to automatically form regions according to the settings and each region gives the syllables present in the word. The length of the syllable, their start and end location which is required for concatenation can be obtained using the view region list option [57]. In our implementation we have used the manual process of cutting the syllables to ensure accuracy.

6.6.4. Data Sets used

A sample text file to generate speech from the TTS system is as shown below:

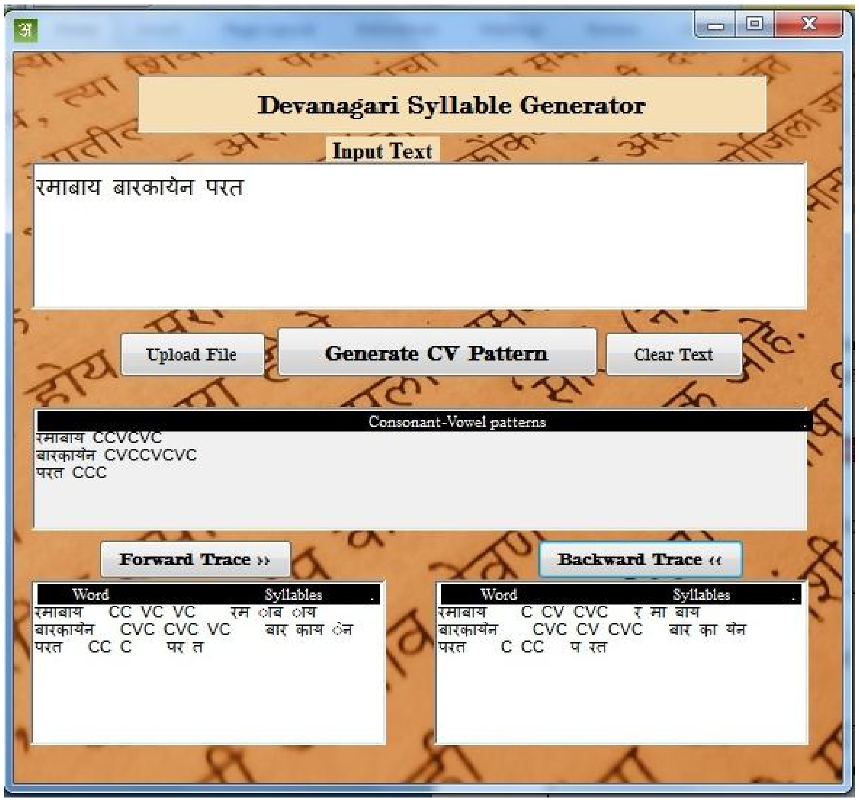
मराठी चार पुस्तकां जातकच एकतर घरा बसून शिवण सूत करप वा घर संवसार शिकप.
व्हेल्ल्या घरा त्रास जावचे न्हय म्हूण तेन्नाच्यो आवयो आपल्या चलयांक घर संवसाराचीं कामां मुद्दाम शिकयताल्यो.

Some sample sentences (Hindi, Marathi and Konkani) of Devanagari script used in the experiment for syllabification are: **वहा शायद कुछ पताचले।** (*Vaha Shayad Kuch Patachale*), **बाघारू ठेच लागून पडतो.** (*Bhagharu Thech Laagun Padto*), **रमाबाय पणजे शारात वाडिल्ली.** (*Ramabai Panaje Shaarat Vadilee*) along with the Konkani sentences listed at 3.4.1

6.6.5. Experimental Results

The syllabification results of the manual trace for backward and forward approaches were tested with the developed syllable based TTS system. The test results indicate that the backward approach correctly syllabify all the Devanagari words. The same is shown in the screenshot of the developed TTS system is shown in Fig. 6.10 for the given syllable patterns.

Table 6.9 depicts sample Hindi, Marathi and Konkani sentences with syllabified text using the backward and forward syllabification algorithms. The incorrect syllabified words are shown with highlighted background. The simulation results show 100% accuracy using the backward approach for syllabification.



रमाबाय बारकायेन परत

Consonant-Vowel patterns

```
रमाबाय CCVCVC
बारकायेन CVCCVCVC
परत CCC
```

Forward Trace

```
Word          Syllables
रमाबाय     CC VC VC      रम ाब ाय
बारकायेन    CVC CVC VC      बार काय ेन
परत     CC C      पर त
```

Backward Trace

```
Word          Syllables
रमाबाय     C CV CVC     र मा बाय
बारकायेन     CVC CV CVC     बार का येन
परत     C CC     प रत
```

Fig. 6.10 : Devanagari Syllable Generator

Table 6.9 : Syllabified Words with Syllabification Algorithms.

Devanagari Language	**Sentence**	**CV Structure**	**Syllables Generated using Forward Approach**	**Syllables Generated using Backward Approach**
Hindi	**वहा शायद कुछ पताचले।** (*Vaha Shayad Kuch Patachale*)	CCV - CVCC - CVC - CCVCCV	वह ा – शाय द – कुछ - पत ाच ले	व हा – शा यद – कुछ - प ताच ले
Marathi	**बाघारू ठेच लागून पडतो.** (*Bhagharu Thech Laagun Padto*)	CVCVCV - CVC - CVCVC - CCCV	बाघ ार ू – ठेच – लाग ून - पड तो	बा घा रू – ठेच – ला गून – पड तो
Konkani	**रमाबाय पणजे शारात वाडिल्ली.** (*Ramabai Panaje Shaarat Vadilee*)	CCVCVC- CCCV- CVCVC- CVCVCCV	रम ाब ाय – पण जे – शार ात – वाड ि ल ली	र मा बाय – पण जे – शा रात – वा डिल ली

Table 6.10 shows the hit count for the recorded words, recorded syllables, recorded diphones and phonemes for sample Hindi, Marathi and Konkani sentences presented to the developed syllabification system using the backward approach.

Table 6.10 : Sample Sentences Tested for the Backward Syllabification

Devanagari Language	**Sentence**	**Syllabification (Backward tracing)**		
		No. of hits for recorded words	***No. of hits for recorded syllables***	***No. of hits for diphones and phonemes***
Hindi	**आज-कल तो स्कूलो की छुट्टी है ।** (*Aaaz-Kal To Scholo Ki Chutti Hai*)	**2**	**2**	**4**
Marathi	**पण पुढच्याच क्षणी आणखी एका धक्क्यात बहादुर गाडी पुढे नेतो.** (*Pan Phudchya kshani Aankhi Eka Dhakyat Bhahadur Gaddi Pudhe Neto*)	**3**	**4**	**5**
Konkani	**विनोदाचीं अशे तरेची मता जावपाक एक कारण आशिल्ले.** (*Vinodache Ashe Tareche Mata Javapak Ek Karan Ashile*).	**2**	**4**	**6**

A volunteer based evaluation was carried out on a set of sentences to compare the basic concatenative TTS system with the syllable based concatenative TTS system for Konkani. The feedback of the volunteers for the two TTS systems in terms of the Mean Opinion Score (MOS) is depicted in Fig. 6.11.

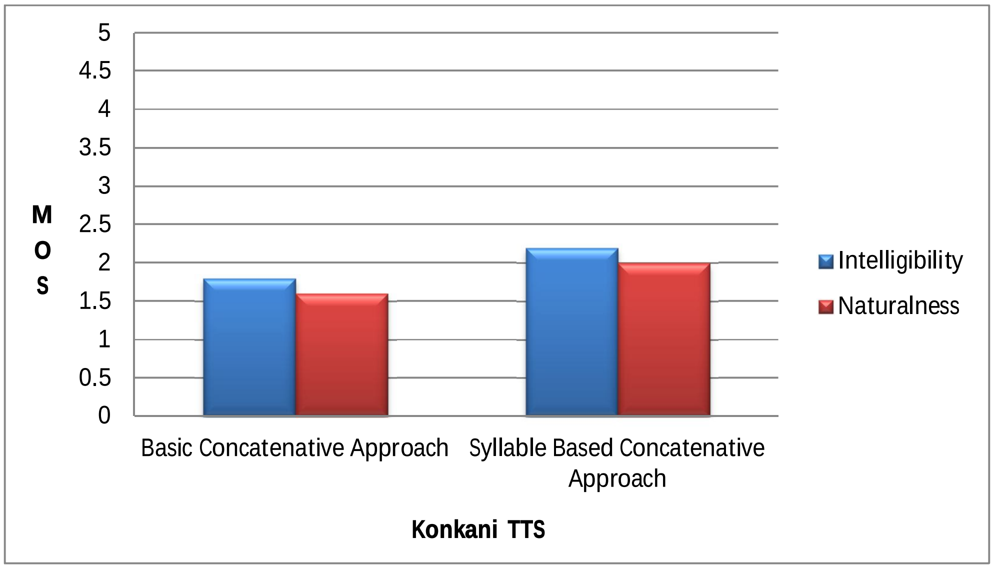


Fig. 6.11 : Feedback of the Volunteers for the TTS System Using Basic Concatenative Approach and Syllable based Concatenative Approach

Based on the outcome of the volunteers test, it is observed that the syllable based concatenative approach shows overall improvement over the basic concatenative approach. On analyzing the extensive tests carried out by exposing the system to more Konkani text considering Konkani phonology [11] [14] [75] and the listener’s feedback, following are the observations:

(i) Recorded Syllables are best suited for generation of speech as compared to the speech obtained from syllables formed by concatenation diphones / phonemes.

(ii) Syllables formed using diphones / phonemes do not differentiate the presence of inherent schwa in consonants. For example the word, ‘घर’ (ghar) after deletion of the inherent schwa should be read as ‘घर्' (ghar) which is actually spoken as ‘घर’(ghar**a**) due to the uttering of the consonants straight forward from the database without any processing.

(iii) Syllables formed using diphones/phonemes do not recognize or classify the words for the presence of nasal sounds represented by ‘ं’ over consonant or vowel. For example in the word ‘झुंबर’ (jhumber), the nasal sound is represented as ‘म्’ (m). e.g. झुंबर → झु+म्+बर.

(iv) Syllables formed do not differentiate the effect of vowel harmony for specific words. For example the phrase ‘ती तेंफळ’ uttered as ‘ती तेंफळ’ (tee teefal) represents the tree and the phrase ‘तें तेंफळ’ uttered as ‘तें तॅफळ’ (te tefal) represents the fruit of the tree. It is significant to note that, the word is written as ‘तेंफळ’ in both the phrases.

(v) Syllables formed using diphones/phonemes do not consider diphthong wherein sound units of type इव (Ev), इय (Ey), उव (Uv), उय (Uy), एव (Iv), एय (Iy), ओव (Ov) and ओय (Oy) are differently uttered. For example in the word ‘दिवज’ (divaj), there is stress on the first vowel and ‘व’ (V) remains silent, resulting in the utterance of sound unit ‘दिवज् (इव)’.

(vi) Not adequately considered for sentence and words level stress as prosody control in synthesis.

6.7. Summary

We developed and demonstrated the syllabification algorithms using the forward approach and the backward approach for Devanagari text. The manual tracing of the algorithms for Konkani words to generate syllables as well as the system generated syllables indicate that the backward approach for syllabification gives 100% accuracy as compared to the forward approach for syllabification. The backward syllabification algorithm can be of use to the linguists for research in their domain. The same can be also extended for all Devanagari languages by giving the appropriate syllable patterns for the language.

Using the syllabification approach we are able to generate speech for the given text based on the limited recorded speech corpus. The developed syllable based concatenative TTS system performs better than the basic concatenative TTS system. In terms of the Konkani phonology rules and the observations thereon, we infer that in order to get better naturalness, it is essential that the TTS system address the language specific issues. The improvement in the performance of the system is dealt by studying Konkani phonology and is detailed in the subsequent chapter.

Chapter 7

Syllabification Using Konkani Phonology Rules

7.1. Introduction

Different types of nasal vowels add special features to the Konkani language [15]. In the context of phonology for the TTS we have considered the phonology rules for Konkani [11] [14] [75] for the development of a Konkani TTS which are explained in this chapter.

7.2. Related work

We reviewed the following phonology based papers in the area of TTS:

(i) The work on schwa-deletion phenomenon in Hindi and how it is handled in the pronunciation component of a multilingual concatenative TTS system indicate that two main factors complicate the issue of schwa pronunciation in Hindi. First, not every schwa following a consonant is pronounced within the word. Second, in multimorphemic words, the presence of a morpheme boundary can block schwa deletion where it might otherwise occur. A model for schwa-deletion is proposed which combines a general purpose schwa-deletion rule proposed in the linguistics literature with additional morphological analysis necessitated by the high frequency of compounds in the database [76].

(ii) Using polysyllabic units initially, this work later builds a monosyllable cluster unit TTS to design and develop an Indian language TTS systems. It also addresses the issues of “halants”. Halant denotes consonants without the inherent vowel or half consonants. In Hindi, in the presence of halants, vowel deletion takes place and thus the pronunciation does not exactly match the written form. Since Indian languages are syllabic, a need for pronunciation definition is felt for all possible contexts. The one-to-one correspondence between the written and the spoken form does not always hold good [77].

7.3. Konkani Phonology Rules

The following phonology rules for Konkani were studied for implementation.

7.3.1. Schwa Deletion Rules

In Konkani some words express different meaning depending on their position in the sentence. For such words, without schwa deletion, the word not only sounds unnatural, but also makes it extremely difficult for the listener to infer the correct sound. Thus the pronunciation of schwa is context dependent in Konkani. The rules for schwa deletion are well known for Konkani language and are depicted in Table 7.1 with examples.

Table 7.1 : Schwa Deletion Rules with Examples.

Character Based Rules	**Utterance Rule**	**Sample Word**
Single letter words	Presence of '**अ**' (a) is always uttered	**व** (Va), **ह** (Ha), **क** (Ka)
Words terminating with '**-ना**'(Naa) or '**-नात**'(Naat)	Presence of '**अ**' (a) in the previous character is un-uttered	**वचना** → **वच्-ना** (Vach**a**na) →(Vachna) **पाव्-नात** (Pav**a**nat) →(Pavnat)
Words with **जोडगिरा** (jodakshars)	Presence of '**अ**' (a) in the jodakshars is uttered.	**घुस्पले** → **घुस्पले** (Ghus**a**p**a**le) →(Ghuspale)
Three letter words	Presence of '**अ**' (a) in the middle is uttered while at the end remains silent	**माजर** → **माजर्** (Majara) → (Majar)
Four letter words	Presence of '**अ**' (a) at the second position remains silent while '**अ**' (a) at the third position is uttered.	**मणकट** →**मण्-कट्** (Manakata) → (Mankat)
Five letter words	Presence of '**अ**' (a) at the second position remains silent**.** At the third position, '**अ**' (a) is uttered if present. At the last position, remains un-uttered and if '**अ**' (a) is present prior to this position then '**अ**' (a) is uttered.	**लकलकप** → **लक्-लक-प्** (Lakalakapa) → (Laklakap)

7.3.2. Jodhakshars Concept and its Syllabification

In Devanagari languages, two or more consonants can occur as a single unit. They are usually referred to as jodakshars or jodgir (जोडगीर अक्षरांत). Jodakshars and the syllabification rules for Konkani language are presented with syllabification examples in Table 7.2.

Table 7.2 : Jodakshars Rules with Examples.

Input Word	Pre-processing Steps	Input String	Syllables Encountered	Syllabified Word
अर्वळ (Arval)	अर् – वळ	VCCC	VC+CC	अर् – वळ
विठ्ठल (Vithal)	विठ् – ठल	CVCCC	CVC+CC	विठ् – ठल
माळ्ळो (Maaloo)	माळ् –ळो	CVCCV	CVC+CV	माळ् – ळो
पन्नासावी (Pannasavi)	पन् – नासावी	CCCVCVCV	CC+CV+CV+CV	पन् – ना - सा – वी

The sound classes (syllable types) of the input phonetic string for an example Jodhakshar for the Konkani word “अस्मिताय” is as shown in Table 7.3. The test results are demonstrated in Table 7.4.

Table 7.3 : Preprocessing Steps for the Word “अस्मिताय”

Input Word	Pre-Processing Steps
अस्मिताय	Apply Case 3: (Split jodakshar into two parts)
अस् – मिताय	**VCCVCVC**

Table 7.4 : Syllabification with Phonology for the Word “अस्मिताय”

Input String	Syllable Found	Description of Rules Applied
VCCVCV$\underline{C}$	CVC	The symbol traversed is ‘C’, its left most sound unit is ‘V’, so we further traverse. Symbol traversed left most is ‘C’ and a class of “CVC” will be formed. Pointer will be shifted 3 places left.
VCC$\underline{V}$CVC	CV	The symbol traversed is ‘V’ and its left most symbol is ‘C’, hence a class of “CV” will be formed. Pointer will be shifted by 2 places left.
V$\underline{C}$CVCVC	VC	The symbol traversed is ‘C’, its left most sound unit is ‘V’, and hence a class of “CVC” will be formed. Pointer will be shifted 3 places left.
अस्मिताय **VCCVCVC**	$\Longrightarrow$	**अस् -मि - ताय** **VC - CV – CVC**

7.3.3. Nasal Utterance and its Syllabification

Nasalization is utterance of sound through nose. Nasal sound in Devanagari is represented by ‘◌ं’ (अनुस्वार - Anuswar) over consonant or vowel. Nasal utterance is specifically of consonants of type न् (na), ण् (na), ङ् (ṅa) and म् (ma). Hence there is a need to carry out pre-processing of words involving nasal sounds (अनुस्वार) followed by the normal flow of algorithm. Table 7.5 depicts nasalization rules with examples for Konkani. Thus, it is noted that, अनुस्वार depends on succeeding letter and hence it may be tough to predict the output when अनुस्वार is on the last letter of the word. In such cases, pre-processing is suitable.

Table 7.5 : Nasalization Rules with Examples.

Nasal Utterance Character	**Preceding Character**	**Sample Word**
ङ्(ṅa)	क, ख, ग, घ	रंग → र+ङ्+ग (Rang)
न्(na)	च, छ, ज, झ, त, थ, द, ध, न	हांचो → हा+न्+चो (Hancho)
ण्(ṇa)	ट, ठ, ड, ढ, ण	पंटू → प+ण्+टू (Pantu)
म्(ma)	प, फ, ब, भ, म	झुंबर → झु+म्+बर (zhumber)

The sound classes (syllable types) of the input phonetic string for an example nasal (अनुस्वार) word for the Konkani word "चिंता" is as shown in Table 7.6. The test results are demonstrated in Table 7.7.

Table 7.6 : Preprocessing Steps of the Word "चिंता"

Input Word	**Pre-Processing Steps**
चिंता	Apply Case 2: (Open अनुस्वार)
चि- न्- ता	**CVCCV**

Table 7.7 : Syllabification with Phonology for the Word "चिंता"

Input String	**Syllable Found**	**Description of Rules Applied**
CVCC<u>V</u>	CV	The symbol traversed is 'V', its left most sound unit is 'C', hence a class of "CV" will be formed. Pointer will be shifted 2 places left.
CV<u>C</u>CV	CVC	The symbol traversed is 'C', its left most sound unit is 'V', so we further traverse. Symbol traversed left most is 'C' and a class of "CVC" will be formed. Pointer will be shifted 3 places left..
चिंता **CVCCV**	$\Longrightarrow$	चिन् – ता **CVC - CV**

In order to syllabify Jodakshars and nasal words, pre-processing is required to be carried out on words before applying simple syllabification. Consider the examples of table 7.2 and 7.8 to recognise the sound classes (syllable types) formed by jodakshar and nasal words. Support to jodakshar requires improvement by understanding more concrete rules, requiring all cases to be examined equally instead of focusing on a few. Also there are cases like jodakshar and nasal occurring in same unit of letter. The algorithm is subjective in enhancement with support of vowel alteration, diphthongs and additional prosody information.

Table 7.8 : Examples of Konkani Jodakshar and Nasal Words

Input Word	Pre-processing Steps	Input String	Syllables Encountered	Syllabified Word
स्पेन	स् – पेन	CCVC	C+CVC	स् – पेन
हांचो	हान् चो	CVCCV	CVC+CV	हान् – चो
झुंबर	झुम् बर	CVCCC	CVC+CC	झुम् - बर
थंडी	थण् डी	CCCV	CC+CV	थण् – डी
तांका	ताङ् का	CVCCV	CVC+CV	ताङ् - का
संबंधित	स म् बन् धित	CCCCCVC	CC+CC+CVC	सम् - बन् – धित

7.3.4. Diphthong Concept and its Syllabification

A diphthong (संदी-स्वर) is "two sounds" or "two tones" and refers to two adjacent vowel sounds occurring within the same syllable. A diphthong is a vowel wherein the tongue moves during the pronunciation of the vowel.

In Konkani, vowels ‘आ (Aa)’, ‘इ (i)’, ‘उ (u)’, ‘ए (e)’ and ‘ओ (Au)’ when adhered to semi-vowels ‘य (y)’ and ‘व (v)’, diphthongs (संदी-स्वर) are formed. Thus we have type इव (Ev), इय (Ey), उव (Uv), उय (Uy), एव (Iv), एय (Iy), ओव (Ov) and ओय (Oy). Table 7.9 and 7.10 depicts diphthong rules for Konkani with examples and Diphthongs in Konkani with C-V Patterns

Table 7.9 : Diphthongs Rules with Examples

Input Word	Input String	Syllables Encountered	Diphthongs	Syllabified Word
कायल (Kayal)	CVCC	CVC+C	**आय**	**काय – ल**
जेवन (Jevan**)**	CVCC	CVC+C	**एव** (**अेव**)	**जेव – न**
रोयण (Royan**)**	CVCC	CVC+C	**ओव**	**रोय – ण**
लुवंया (Luvaya**)**	CVCCV	CVC+CV	**उव** (**अुव**)	**लुवं – या**
घेवंक (Ghevak**)**	CVCC	CVC+C	**एव** (**अेव**)	**घेवं – क**
मेवणी (Mevni**)**	CVCCV	CVC+CV	**एव** (**अेव**)	**मेव – णी**

In order to syllabify diphthongs in words, modifications to Syllabification algorithm is required. Here we infer that, presence of diphthongs in words requires modifications to Syllabification algorithm.

In the proposed algorithm to syllabify words, we need them in terms of simply consonants (C) and vowels (V). In phase I, it scans each token from right to left. In phase II, on encounter of symbol ‘C’, it identifies patterns of type CVC, VC and C. As diphthongs in Konkani are of type ‘CVC’, it checks additionally if ‘CVC, is diphthong. If diphthong is encountered then it replaces it with appropriate unit. Similarly in phase III, on encounter of symbol ‘V’, it identifies patterns of type CV and V. There is no effect of diphthong detection in phase-III.

Table 7.10 : Diphthongs in Konkani with C-V Patterns

Diphthongs	C-V Representation	Diphthongs	C-V Representation
आव	CVC	आय	CVC
इव (अिव)	CVC	इय (अिय)	CVC
उव (अुव)	CVC	उय (अुय)	CVC
एव (अेव)	CVC	एय (अेय)	CVC
ओव	CVC	ओय	CVC

To depict the results of some traced examples of Diphthong words. We summarize our algorithm as follows: The backward syllabification algorithm works in three phases scanning patterns of Cs and Vs in backward. In phase I, it scans each token from right to left. In phase II, on encounter of symbol ‘C’, it identifies patterns of type CVC, VC and C. Similarly In phase III, on encounter of symbol ‘V’, it identifies patterns of type CV and V. A special case for diphthong detection is to check if ‘CVC’ pattern belongs to class of diphthong sounds.

The test results of syllabification for an example diphthong for the Konkani word “बापायक” is demonstrated in Table 7.11. Note that the example presented here are only to depict diphthong identification and should not be combined with identifying vowel harmony to avoid confusion.

Table 7.11 : Syllabification with Phonology for the Word “बापायक”

Input String	Syllable Found	Description of Rules Applied
CVCVC<u>C</u>	C	The symbol traversed is ‘C’, its left most sound unit is ‘C’, so we stop traverse. We ignore current ‘C’ and hence class of “C” will be formed. Pointer will be shifted 1 place left.
CVCV<u>C</u>C	CVC	The symbol traversed is ‘C’, its left most sound unit is ‘V’, and hence a class of “CVC” will be formed. Since CVC class is found, we check for diphthong sound group. It belongs to ‘**आय**’ group. Pointer will be shifted 3 places left.
CV<u>C</u>VCC	CV	The symbol traversed is ‘V’ and its left most symbol is ‘C’, hence a class of “CV” will be formed. Pointer will be shifted by 2 places left.
बापायक **CVCVCC** $\Longrightarrow$		**बा - पाय – क (आय)** **CV - CVC – C**

7.3.5. Vowel Harmony and its Utterance

Vowel harmony is a type of long-distance assimilatory phonological process involving vowels that occur in some languages. In such languages, there are constraints on which vowels may be found near each other. In Konkani each of vowels ‘अ’, ‘ए’ and ‘ओ’ has two vowel variations. That is two ‘अ’, two ‘ए’ and two ‘ओ’ are important variation of vowels in its script.

Let us try to understand each of these cases with examples shown in Table 7.12.

Table 7.12 : Vowel Harmony Rules with Examples

Input Phrase	Vowel Harmony Character (‘ए’ and ‘ओ’)	Utterance	Meaning of the Context
तें आंतेर (Te Aanter)	**ए**	**तें आंतॅर** (Te Aanter)	Represents fruit of a tree
ती आंतेर (Tee Aanteer)	**ए**	**ती आंतेर** (Tee Aanteer)	Represents tree
तें बोर (Te Bor)	**ओ**	**तें बॉर** (Te Bor)	Represents fruit of a tree
ती बोर (Tee Boor)	**ओ**	**ती बोर** (Tee Boor)	Represents tree

If the word occurs in two different contexts then it is uttered differently. The way of writing the words may be reasonably same, but their utterance is not the same way. Aksharas can be explicitly noted by giving चंद्र (ॅ). But the use of such explicit चंद्र (ॅ) is not noted in Konkani. Considering the above phonological rules affecting syllabification, the sound inventory was appended with the following for improving the naturalness of the produced speech:

(i) Consonants (in Konkani): with schwa and without schwa.

(ii) Vowels (in Konkani): including for vowel harmony for ‘ए’ and ‘ओ’ type.

(iii) Nasal, Diphthong sounds (in Konkani).

7.4. The TTS Model

The architectural block diagram for the developed syllable based TTS system with Konkani phonology rules is depicted in Fig. 7.1.

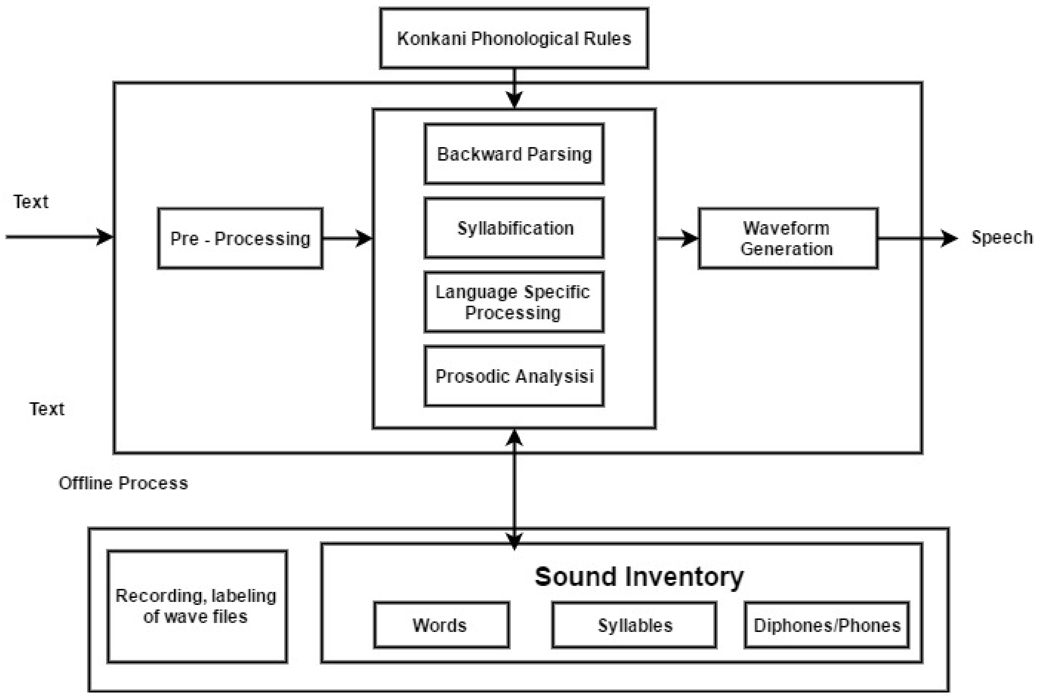


Fig. 7.1 : Syllable based TTS System with Konkani Phonology

The block diagram depicted above is explained as follows: The major components in TTS Syllabification using Phonological rules includes pre-processing, backward parsing, syllabification, language specific processing, prosodic analysis, sound inventory and waveform synthesis. The given input raw text is analyze to consider all types of punctuations for prosodic analysis, numerical expansions and pass the plain text for syllabification algorithm. The backward syllabification algorithm along with the phonology rules for Konkani language are applied further passing the text to consider for prosodic analysis. Finally the pre-recorded speech segments are used to concatenate and play the desired output as the sequences generated by the syllabification output. Note that the recording, cutting and labeling of wave files of words, syllables, diphones and phones are done offline to the TTS system developed.

In this approach we are using phonemes, diphones, syllables and words, as the basic units. The evaluation criterion for a speech unit used for the repository is defined in terms of two costs [78] namely:

(i) **Target cost**: indicates the closeness to the actual speech.

(ii) **Concatenation cost**: indicating the perfect matching of the join when using concatenation.

An ideal speech unit for the speech repository would be the unit that minimizes both costs.

7.5. Implementation

The following sections discuss the implementation of the Konkani phonology rules toward improving the naturalness in the developed TTS system.

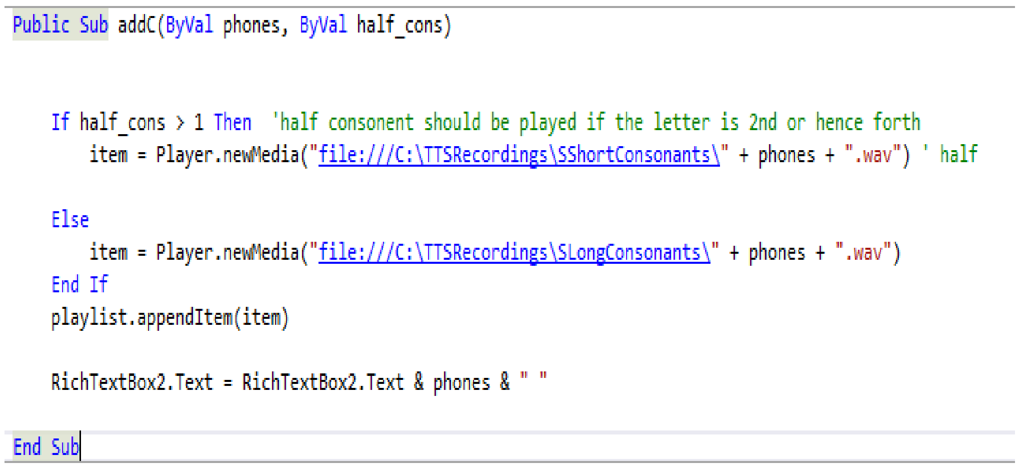
7.5.1. Schwa Deletion

As examined in the earlier sections, there are a set of schwa deletion rules for Konkani. There are specific rules in Konkani for schwa utterance present in a character.

To implement the schwa deletion rules, all the consonants of Konkani are stored as half utterance sound and as full utterance sound. These wave files are stored in two separate file folders. The consonants with the presence of schwa characters are recorded with longer duration while the consonants without schwa are recorded with shorter duration. Thus the system maintains two types of voices for all consonants.

The function used in the implementation examines the given string of CV patterns against the actual character under examination.

The most commonly encountered rule for schwa deletion is for more than one letter words, in the first letter if ‘अ’ is present, then it always gets uttered while in the last letter if ‘अ’ is present, then it always remain un-uttered. The function scans for consonants as per the if-else condition and extracts the respective short or long recorded consonant. Fig. 7.2 below shows the sub function to extract and concatenate the corresponding audio file. This process is repeated for the whole text.



```
Public Sub addC(ByVal phones, ByVal half_cons)


    If half_cons > 1 Then  'half consonent should be played if the letter is 2nd or hence forth
        item = Player.newMedia("file:///C:\TTSRecordings\SShortConsonants\" + phones + ".wav") ' half

    Else
        item = Player.newMedia("file:///C:\TTSRecordings\SLongConsonants\" + phones + ".wav")
    End If
    playlist.appendItem(item)

    RichTextBox2.Text = RichTextBox2.Text & phones & " "

End Sub
```

Fig. 7.2 : Sub Function for Schwa Deletion

7.5.2. Jodakshars Syllabification

Jodakshars are formed by combination of two or more consonants. Whenever a jodakshar is encountered, two or more consonants are examined as individual letters, however while uttering individual letters, the first letter is played from the short recorded sound and the second part of the letter is played from the long recorded sound.

7.5.3. Nasal Syllabification

Nasal sounds are specifically generated for consonants of type न् (na), ण् (ṇa), ङ् (ṅa) and म् (ma). A nasal sound in Devanagari is represented by ‘ं’ (अनुस्वार - Anuswar) over a consonant or a vowel. The algorithm to implement nasal utterance and its syllabification is outlined below:

```
--------------------------------------------------------------------------------------------------------------
Nasal_Syllabification_Algorithm (Word, Syllable)
--------------------------------------------------------------------------------------------------------------

        Input :- Word
        Output :- Syllable
        i=1;
        While ( i < len (Word)
        {
        if( Word[i+1] == "क" ||  Word[i+1]  == "ख" || Word[i+1] = = "ग" || Word[i+1]  == "घ")
        {
        i= i+1 ;
        Syllable=Syllable+"SNasal\ङ.wav";
        }
        if( Word[i+1] == "च" || Word[i+1]  == "छ" || Word[i+1] = = "ज" || Word[i+1] = = "झ" || Word[i+1] == "त" || Word[i+1] == "थ" || Word[i+1]  == "द" || Word[i+1]  == "ध" || Word[i+1] = = "न")
        {
        i= i+1;
        Syllable=Syllable+ "SNasal\न.wav";
        }
        if( Word[i+1] == "ट" || Word[i+1]  == "ठ" || Word[i+1] == "ड" || Word[i+1] == "ढ" || Word[i+1]  == "ण")
        {
        i= i+1 ;
        Syllable =Syllable +"SNasal\ ण.wav";
        }
        if( Word[i+1] == "प" || Word[i+1]  = "फ" || Word[i+1]  = "ब" || Word[i+1]  = "भ" || Word[i+1]  = "म")
        {
        i= i+1 ;
        Syllable =Syllable + "SNasal\म.wav";
        }

        }
        return Syllable;
--------------------------------------------------------------------------------------------------------------
```

A function is used to detect and utter the nasal sounds in our implementation of rule based synthesis technique. As the nasal sounds are not fully uttered, they are saved in the wave file as half sound for the purpose of implementation. As the nasal sound depends on the succeeding letter, the function checks for specific case based characters. To detect the length of each word present in a sentence, the text file is read. Assuming 'syllablech' is holding each read character of 'ं', the steps involved in detecting the word length is as shown below:

1. Case 1 (for ङ): If syllablech (i+1) = "क" Or syllablech (i+1) = "ख" Or syllablech (i+1) = "ग" Or syllablech (i+1) = "घ" Then increment the pointer by 1 and extract "SNasal\ङ.wav" audio file and concatenate.
2. Case 2 (for न): If syllablech (i+1) = "च" Or syllablech (i+1) = "छ" Or syllablech (i+1) = "ज" Or syllablech (i+1) = "झ" Or syllablech (i+1) = "त" Or syllablech (i+1) = "थ" Or syllablech (i+1) = "द" Or syllablech (i+1) = "ध" Or syllablech (i+1) = "न" Then increment the pointer by1 and extract "SNasal\न.wav" audio file and concatenate.
3. Case 3 (for ण): If syllablech (i+1) = "ट" Or syllablech (i+1) = "ठ" Or syllablech (i+1) = "ड" Or syllablech (i+1) = "ढ" Or syllablech (i+1) = "ण" Then increment the pointer by 1 and extract "SNasal\ ण.wav" audio file and concatenate.
4. Case 4 (for म): If syllablech (i+1) = "प" Or syllablech (i+1) = "फ" Or syllablech (i+1) = "ब" Or syllablech (i+1) = "भ" Or syllablech (i+1) = "म" Then increment the pointer by 1 and extract "SNasal\म.wav" audio file and concatenate.

The above steps are repeated for the whole text.

7.5.4. Diphthong Syllabification

The specific types of syllables obtained using diphthong are of the type इव (Ev), इय (Ey), उव (Uv), उय (Uy), एव (Iv), एय (Iy), ओव (Ov) and ओय (Oy). In our implementation, we have considered diphthongs itself as syllables and thus we store each of these syllables as a single unit. The presence of diphthongs in words requires modifications to the syllabification algorithm as shown in Fig. 7.3.

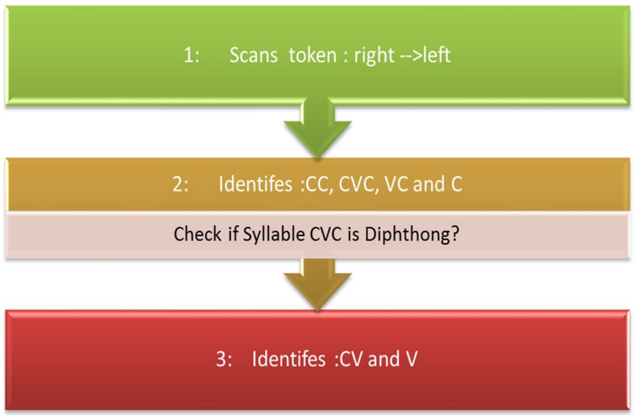

1: Scans token : right -->left
2: Identifes :CC, CVC, VC and C
Check if Syllable CVC is Diphthong?
3: Identifes :CV and V

Fig. 7.3 : Modifications to Existing Syllabification Algorithm

In our implementation of the syllabification algorithm, the function scans for patterns of the type 'CVC' where it belongs to the diphthong category. If it satisfies the condition then it extracts the corresponding diphthong audio file and concatenate. The process is repeated for the whole text.

7.5.5. Vowel Harmony

There are no generalized rules to implement vowel harmony thus making it difficult to write a procedure for its implementation. For example, the utterance of vowels ‘ए’ and ‘ओ’ have two variations. We have not come across any rules in the available literature to represent the 2 variations.

We consider two approaches for implementing vowel harmony: first, at the letter level and second, at the phase level.

1. Letter level: For implementation of the Konkani TTS system, we notice that the ‘◌̆’ behavior of the vowel is more frequently used then the normal behavior of the vowel. Thus we consider a function to examine the specific type of vowels ‘ए’ and ‘ओ’. If there exist such letter, then by default it will be considered as ‘◌̆’.
2. Phase level: We implement this in order to represent vowel harmony for a specific set of words. For each occurrence of such word, scan the preceding letter. If the letter is ‘तें’, the pronunciation of the word will be with vowel harmony. If the letter is ‘ती’ the pronunciation of the word will be without vowel harmony.

   The above steps are repeated for the whole text.

7.5.6. Normalization in Syllabification

Text normalization converts non-standard text such as number, year, date, time, acronym and abbreviation to its standard form. Konkani abbreviations (mostly used) are stored in a separate file in the TTS Corpus. When any of these abbreviations appears in the text, it will be searched in the database. If it is present, the system will replace the text and play the voice from the database. If it is not present, the original text will be left as it is and read as per the syllabification and concatenation process. It is difficult to add all the abbreviations in the database and thus most commonly used abbreviations are stored and used.

The second step in text normalization is normalizing the non-standard words. Non-standard words are tokens like numbers or abbreviations, which are required to be expanded into sequences of Konkani words before they are pronounced and these numbers, are also added to the speech corpus. It is assumed that digits more than number 10 (cardinal numbers) will be pronounced as digits or its direct words representation. Numbers larger then 99, will be pronounced with concatenation of all the preceding digits recordings.

For example, चि. (Chi) would sound like चिरंजीव (Chiranjiv means Boy). Moreover, certain numbers are to be pronounced as individual digits. For example, a phone number such as ९८५०४०९५७९ (9850409579) has to be pronounced as णव आठ पाच शून्य चार शून्य णव पाच सात णव (Naav Aath Paach Shunya Char Shunya Naav Paach Saat Naav), but it will be pronounced as nine hundred and eighty five crores four lakhs nine thousand five hundred and seventy nine if it is referred to as some measurement. We have handled non-standard words using expression for specific type of patterns.

In order to implement the date type of field in the text, the text is examined for expression of the type <two digit no>/<two digit no>/<four digit no> or expression of the type <two digit no>.<two digit no>.<four digit no>, then correct expansion of the expression is done as <day>/<month>/<year> in Konkani. For example if the input text encounters an expression of type ०२/०६/२००७, then it will be expanded as दोन जून दोन हजार सात (Don June Don Hazzar Sat) since it represents date.

7.5.7. Sound Inventory

The test speech corpus created and discussed at section 6.6.3 was appended with half consonants (36), vowels with harmony (36), diphthongs (19), and some more syllables and words involving phonology and non-standard words involving text normalization for Konkani as detailed in the preceding sections. The developed speech corpus is for the testing purpose only and can be expanded on need basis to provide functionality for a full-fledged Konkani TTS system.

We could also obtain speech data of approximately 5 hours from a Konkani news channel of Goa. A sample test data of 2000 words and 200 syllables were cut from this speech corpus to also test our TTS implementation.

Challenges faced during Speech Recording

Speech recording is a major task involved in corpus based speech synthesis. Proper creation of the speech corpus and thus speech recording ensures good performance of a TTS system.

In the preceding sections, we have highlighted some of the major issues in phonology. A good TTS requires speech handling, controlling and recording by a recording artist. For instance, in order to handle the issues of schwa deletion, the consonants are sometimes spoken long while in some cases the consonants are spoken short. The recording artist requires having a control in reading out the consonants as full consonants and as half consonants. Although we have carried out the recording in a studio environment, such features are not easily recordable even by a professional in the field.

In case of synthesis for words, syllables, diphones and phones; for convenience we initially recorded continuous speech for paragraphs which was then cut into sentences, words and syllables. Words and syllable were extracted from the recorded sentences, which have helped us retain the prosodic features in the recorded speech. Diphones and phonemes were recorded individually.

7.5.8. Data sets used

The data set at section 6.6.4 of chapter 6 was used for testing the syllable based concatenative Konkani TTS system with phonology. The system was tested with a sample text of Konkani sentences inclusive of words containing jodakshars, nasals, diphthongs, vowel harmony and schwa deletions. Different test cases were presented and evaluated as discussed earlier.

7.5.9. Experimental Results

The feedback of the volunteers for the TTS systems to compare the syllable based concatenative approach with the syllable based concatenative approach with phonology in terms of the Mean Opinion Score (MOS) is depicted in Fig. 7.4 and 7.5 when tested for the created Konkani speech corpus and the Konkani news speech corpus respectively. Based on the outcome of the MOS test, it is observed that the output speech performs better with the syllable based concatenative approach with phonology. i.e. when features of Konkani language like jodakshars, nasals, diphthongs, vowel harmony and schwa deletions are considered for syllabification.

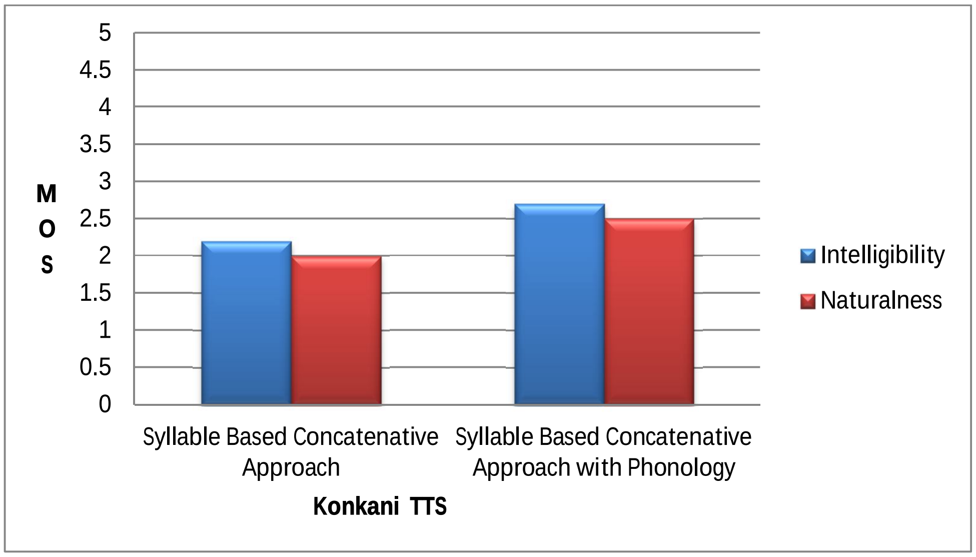


Fig. 7.4 : Feedback of the Volunteers for the TTS System using Syllable based Concatenative Approach and Syllable based Concatenative Approach with Phonology for the created Konkani Speech Corpus

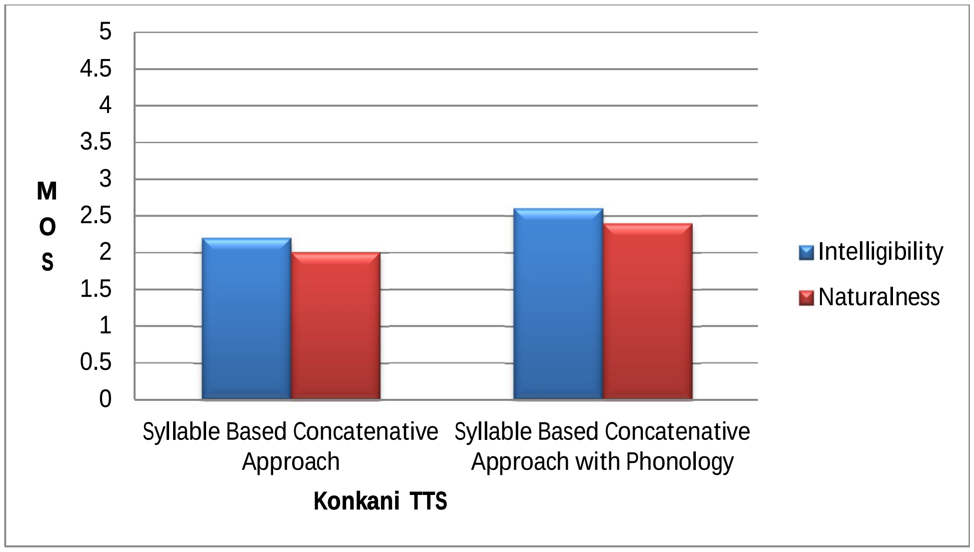


Fig. 7.5 : Feedback of the Volunteers for the TTS System using Syllable based Concatenative approach and Syllable based Concatenative approach with Phonology for the Konkani News Speech Corpus

7.6. Summary

In order to produce quality output, the synthesizer additionally requires considering text processing rather than simply using syllabification. We have addressed some phonological rules for Konkani language like schwa deletion, jodaksharas, nasal words, diphthongs and vowel harmony along with its implementation. The same can be extended for other language specific rules. The speech synthesis output from the implemented system needs to be checked and compared with other available systems for performance analysis.

Chapter 8

TTS System Using Festival Framework

8.1. Introduction

Festival Framework is widely used for the implementation of TTS for many languages [79] [80] [60]. The general framework of festival in addition to building TTS systems can also be used to include examples of various modules and thus adding to its advantage [42][81].

8.2. Festival System

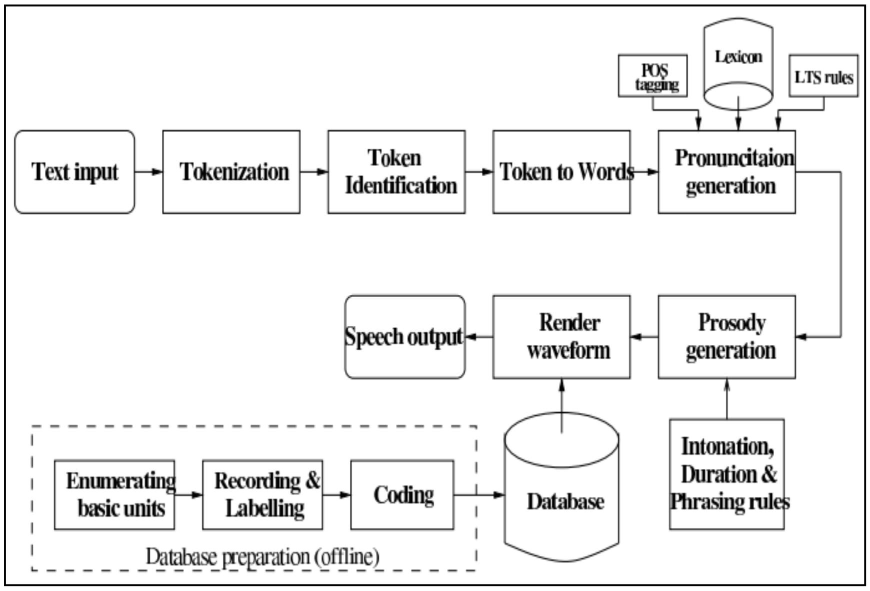


Fig. 8.1 : Festival Architecture

Festival is an open source speech synthesis system framework written in C++, supporting all types of voices and different platforms and has a scheme based command interpreter for control. It is developed in CSTR (Center for Speech Technology Research), University of Edinburgh. The Edinburgh Speech Tools Library is used for low level architecture [42].

Festival speech synthesis system consists of different modules and they together produce the synthetic speech. The modules are: Tokenization and pronunciation generation, prosody generation and Waveform generation and are depicted in Fig. 8.1.

In festival, utterance plays an important role in the generation of synthetic speech. This framework takes the utterance and each of the modules present, manipulates it in some way or the other and passes it to the next module. Utterance consists of a set of items which are related through a set of relations. Each relation consists of a list or tree of items. Items are used to represent objects like words or segments. Relations are used to link items together in a useful way. An item may have one or more relations.
e.g. Normalization of Non-Standard Words.

Table 8.1 : NSW and its Pronunciation

NSW Category	Written format	Pronunciation	Translation
Cardinal Number	९८५०४०९५७९	णव आठ पाच शून्य चार शून्य णव पाच सात णव	Naav Aath Paach Shunya Char Shunya Naav Paach Saat Naav
Ordinal Number	७३४	सातशे चौतीस	Saatshe Chautis
Decimal	२७.५	सत्तावीस पूंज पाच	Sattavis Pungh Paach
Date	०२/०६/२००७	दोन जून दोन हजार सात	Don June Don Hazzar Saat
Time	९.२५	णव वरा पंचवीस मिनटां	Naav Vara Panchvis Minta
Special Character	रू.	रूपया	Rupaya
Abbreviation	चि.	चिरंजीव	Chiranjiv

In the given input text, all the words which are available in the dictionary are called standard words. Numbers, symbols, abbreviations, etc. which are not available in the dictionary for their pronunciations are called as non-standard words (NSW). NSW can be identified as a separate token by the token identifier rule. Table 8.1 shows examples of Konkani NSW and its pronunciations. To identify the token we can use scheme regular expression in festival.

8.3. Implementation

In order to implement the TTS for Konkani language, festival has to be trained with the Konkani text and voice to understand the issues and challenges and then accordingly train the system. Festival lacks few language specific issues and are discussed further in this section.

8.3.1. Installation Details

This section explains the process of installation and configuration of festival Framework for the implementation of a TTS.

8.3.1.1. Linux Platform

- **Linux Platform:** For our system implementation, we have used BackTrack 5 R2, which is an open source Linux distribution. BackTrack provided users with easy access to a comprehensive and large collection of security-related tools. It supports live CD and live USB functionality to allow users to boot BackTrack directly from portable media without requiring installation, though permanent installation to hard disk is possible [82].

8.3.1.2. Installation of Tools

- **Installation of Tools:** Edinburgh Speech Tools provides a set of executables, which offer access to speech tools functionality in the form of a standalone program. As far as possible, the programs follow a common format in terms of assumptions, defaults and processing paradigms. Some of the common features of these tools are arguments to functions which can be specified in any order in the command line.

Installation of festival speech synthesis system tool requires the source of festival framework and speech tools. Festvox project is another framework to generate the new voice models from the recorded speech database developed at CMU (Carnegie Mellon University) [7][83].

To generate voice models and use these models to synthesize the input text for Indic speech database, current versions of festival, speech tools and festvox frame work are used. The latest version of festival and speech tools is available at http://www.cstr.ed.ac.u and the latest version of festvox is available at http://www.festvox.org. We have used the following tools for the implementation of the TTS:

festival-2.4-release.tar.gz, speech _tools 2.4.tar.gz, festvox-2.7.0-release.tar.gz

Next step is to install a simple test suite with festival which requires the three basic voices and their respective lexicons pre-installed for it to work. These are available at:

festvox_kallpc16k.tar.gz, festlex_POSLEX.tar.gz, festlex_CMU.tar.gz

Next, extract festival, speech_tools, and festvox to your home directory and set up the environment variables.

8.3.1.3. Recording and Labeling

We carried out the speech recording using two methods; voice recorded without prosody and voice recorded with prosody. The voice recorded without prosody is not accepted since the festival framework tries to capture phonesets which is possible only with prosody based sounds. Thus we continued the speech recording process with the acceptable approach for the creation of Konkani speech corpus.

Since Devanagari script is also used by other Indian languages, readily available Marathi voice data (around 2000 sentences) was useful to implement and compare the Konkani TTS system using the Festival Framework.

For our implementation, we have used the voice data of the same 500 Konkani sentences mentioned at section 6.6.3. The total recording time for 500 sentences is about 1.5 hrs. The next phase is to label the sentences of the recorded sound file. This is one of the most important and time consuming task and needs to be done very carefully. For this purpose we have used sound editing software Wavesurfer 1.8.8 and the sentences have been labeled manually one by one after carefully listening and analyzing the word sounds [45].

We have followed the same procedure to cut and store the Konkani news voice data of about 5 hours (around 2000 sentences) for the creation and testing of speech corpus for Konkani Language.

Assuming that festival works perfectly, we then build the voices using festvox. The format to pass data is shown in Table 8.2.

The format is "(" followed by a filename, root followed by the text for that sentence, followed by a ")" each on separate line. This text when converted to a phone sequence by festival should match (as closely as possible) the phone sequence of the speech.

Table 8.2 : Sample Training Data

Sr. No.	**Sentence in Konkani**
1	(data_00001 "*रमाबाय पणजे शारांत वाडिल्ली*.") (data_00001 "Ramabai Panaje Sharat Vadilee.")
2	(data_00002 "*शारांचें वातावरण तिजेर परिणाम करूंक पावलें नाशिल्लें*.") (data_00002 "Sharache Vatavaran Tijer Parinam Karunk Pavle Nashille.")
3	(data_00003 "*ताचीं कारणांय उणी नाशिल्ली*") (data_00003 "Tachi Karnai Unni Nashille.")
4	(data_00004 "*राज्य गोऱ्यांचें*.") (data_00004 "Rajya Goryanche.")
5	(data_00005 "*तातूंत आनी राजधानयेचें शार*.") (data_00005 "Tatut Ani Rajdhanyeche Shar.")

8.3.2. Data sets used

The data sets at section 6.5.5 of chapter 6 were used for testing. The TTS system was tested using the training with speech corpus 1 (Marathi), Training with speech corpus 2 (Konkani News), Training with speech corpus 3 (Konkani created).

8.3.3. Experimental Results

The Konkani TTS system using festival was tested for 5 different Konkani sentences trained using the available Marathi voice data.

It is observed that the generated Konkani speech using Marathi voice training is more towards the Marathi accent. And thus we used Konkani speech training data to implement the TTS system using the festival framework.

The evaluation of voice quality testing was performed using Mean Opinion Score (MOS) using the same scores and same volunteers. The MOS depicting the output Konkani speech obtained using the above data sets are shown in Fig. 8.2.

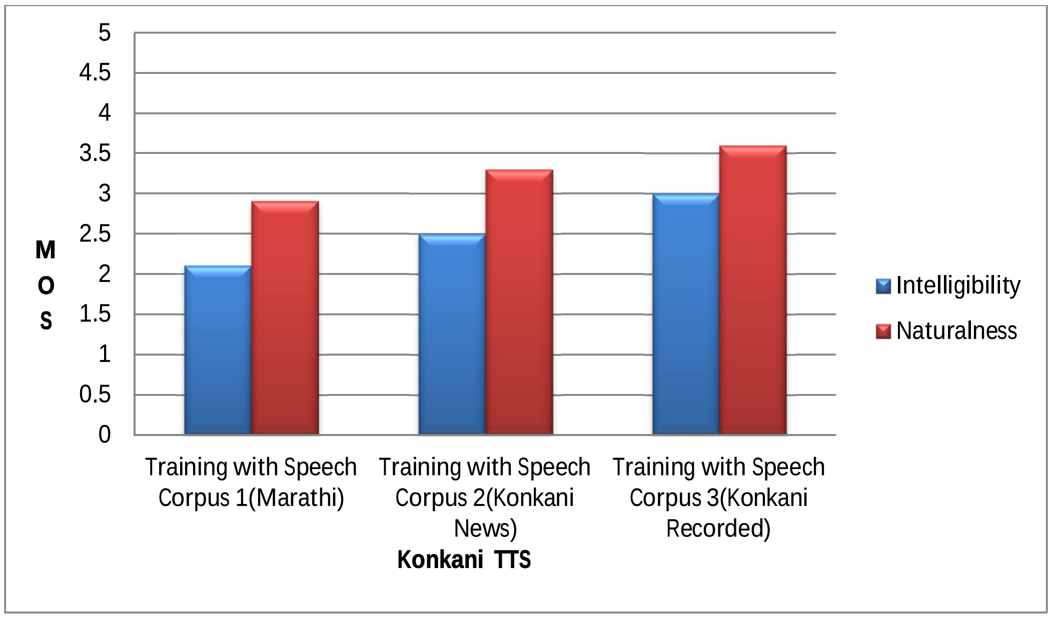


Fig. 8.2 : Feedback of the Volunteers for the TTS System using Festival Framework

Based on the outcome of the listeners test, it is observed that TTS system using festival for Konkani generates an acceptable output for synthesis. On analyzing the test cases and listener's feedback, in terms of some Konkani Phonology rules when applied to certain types of sentences [11][14][75], following are the some of the important observations:

(i) Recorded Syllables are best suited for generation of speech as compared to the speech obtained from syllables formed by concatenation diphones/ phonemes.

(ii) Syllables formed using diphones/phonemes do not address inherent schwa for consonants, presence of nasal sounds, vowel harmony for specific words and diphthong.

(iii) Not adequately considered for sentence and words level stress as prosody control in synthesis.

Thus in context of the above observations, we infer that in order to obtain better naturalness, the synthesizer requires addressing the language specific issues like phonology for implementation in festival.

8.4. Summary

A TTS system for Konkani was implemented and tested successfully on Festival Framework. The output synthesized speech using festival exhibits quality and naturalness based on the subjective quality test results. The system can be implemented and tested for more Devanagari languages to draw comparative results. Further, the voice quality can be improved by implementing language specific phonological rules.

Chapter 9

HMM-Based Speech Synthesis

9.1. Introduction

Currently the most popularly used speech synthesis technique is unit selection, wherein appropriate sub-word units are selected from a large speech databases. Although it is very hard to surpass the quality of the best examples of unit selection, it does have a limitation that the synthesized speech will strongly resemble the style of the speech recorded in the database. However, recording such a large database is very difficult and costly [85].

Over the last few years, a statistical parametric speech synthesis system based on Hidden Markov Models (HMMs) has grown in popularity. A HMM is a finite state machine which generates a sequence of discrete time observations. At each time unit, the HMM changes the states at Markov process in accordance with a state transition probability and generates observational data with an output probability distribution of the current state [86].

9.2. Hidden Markov Model (HMM)

The hidden Markov model is one of the most important machine learning models in speech synthesis. HMM is a mathematical model, which trains the system by observing previously generated outputs. It follows a supervised learning process to train the system and then recognize the pattern. HMM model is formulated for a given problem by defining its different parameters.

Figure 9.1 gives an overview of this system. The basic procedures of implementing HMM-based speech synthesizers to produce synthetic speech can be grouped into two parts: a training part and a synthesis part. The training part is similar to that used in speech recognition systems. The main difference is that both the spectrum and excitation parameters are extracted from a speech database and modeled by context dependent HMMs (phonetic, linguistic, and prosodic contexts are taken into account) [86][87].

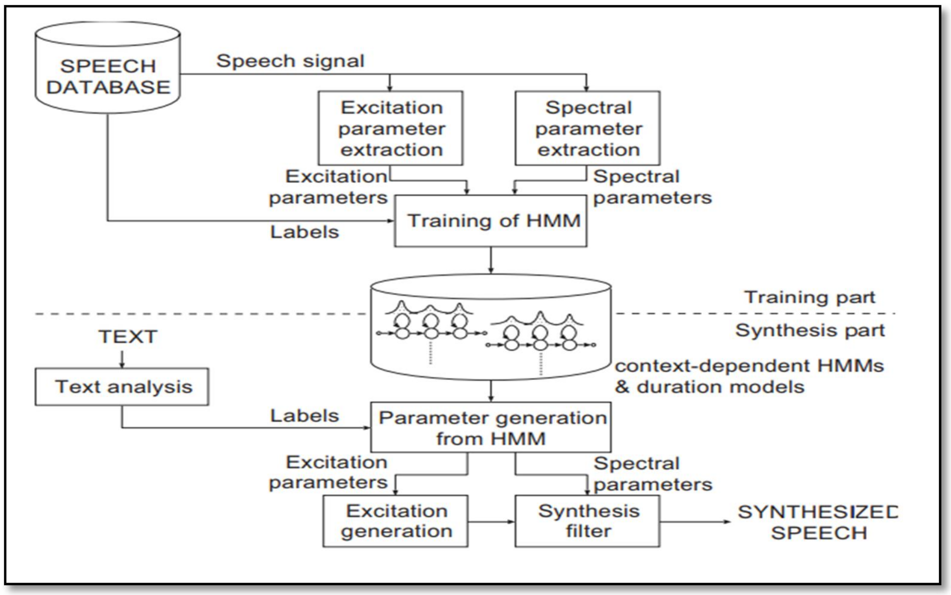


Fig. 9.1 : Architecture of HMM based Speech Synthesis System

The statistical parametric speech synthesis based on HMM has the following features:

- Original speaker's voice characteristics can be easily reproduced because all speech features including spectral, excitation, and duration parameters are modeled in a unified framework of HMM, and then generated from the trained HMMs themselves.
- Using a very small amount of adaptation of speech data, voice characteristics can easily be modified by transforming HMM parameters by a speaker adaptation technique used in the speech recognition systems. From these features, the HMM-based speech synthesis approach is expected to be useful for constructing speech synthesizers which can give us the flexibility we have in human voices.

There are two main advantages of using HMMs to generate speech synthesizers. One is that the produced synthesized speech can be smoothed and made to sound natural. The other is that, since the synthetic speech is created from HMM models with parameters, the characteristics of the voice can be modified easily with adequate parameter transformations [88]. Although the operation and advantages of statistical parameter speech synthesis is impressive, a few disadvantages are associated with it. First, the parameters must be automatically derivable from databases of natural speech. Second the parameters must give rise to high quality synthesis. Finally, the parameters must be predictable from text; the synthesis quality is intelligible but not close to natural speech [85].

A comparison of the HMM based speech synthesis and Unit Selection is presented in Table 9.1:

Table 9.1 : Comparison of the HMM based and Unit Selection Speech Synthesis

HMM based	**Unit selection**
Statistics based	Multi template based
Clustering(Use of HMM)	Clustering(possible use of HMM)
Multiple tree	Single tree
Advantage: Smooth , stable	Advantage: High quality at waveform level
Disadvantage: Vocoded speech (buzzy)	Disadvantage: Discontinuity or miss
Small run time data	Large run time data
Various voices	Fixed voices

9.3. HMM-based Speech Synthesis System (HTS)

HMM based Text to Speech Synthesis system (HTS) developed at the Nagoya Institute of Technology (Nitech-HTS) for a competition of text-to-speech synthesis systems using the same speech databases [88]. STRAIGHT-based voice coding, hidden semi-Markov Model (HSMM) based acoustic modeling and parameter generation considering global variance is illustrated in the paper [89]. The HTS is developed by the HTS working group as an extension of the HMM toolkit (HTK) [85]. The latest version of HTS is 2.3.2 released in December 2017.

The HMM is modeled in such a way that optimal parameters are extracted, which are used in synthesis in the later stage. The following four steps are carried out for modeling of HMM in HTS: a) Definition of Hidden Markov model (HMM), b) Speech Parameter Generation from HMM with dynamic features, c) Determination of State Durations and d) Solution for The Problem

HTS is developed successfully and it is being continuously enhanced further. The following are recent improvements which are incorporated in HTS: a) STRAIGHT, b) Statistical mixed excitation, c) HSMM, d) MGE training, e) GV-based parameter generation algorithm, and f) Blizzard Challenge.

9.4. Configuration and Installation of HTS

HTS can be installed on windows or Linux platform. However, it is recommended that Linux platform is better choice. In order to start installing HTS on Ubuntu 16.04 Linux platform, we need following basic packages such as csh gcc g++ g++-multilib make libncurses5-dev libx11-dev clang byacc flex bc. We have installed the basic packages required for HTS on Ubuntu 16.04 Linux platform.

Once these basic packages are installed on the Ubuntu 16.04 platform, we have to install all the main packages of HTS. The following are the main HTS packages we have installed on Ubuntu 16.04 Linux platform.

a) Speech_tools

b) Festival

c) Festvox

d) HTS_patch

e) HTS_engine

f) SPTK tool

g) ActiveTCL

After successful installation of the basic and main packages, as a final step of HTS configuration and installation we have installed the HTS package using configure and make command on Ubuntu 16.04 platform. The next step is to train the system with voice data to generate speech for the language under consideration. We plan to implement this TTS for Konkani with the availability of large speech corpus from the Linguistics Data Consortium for Indian Languages (LDC-IL), Central Institute of Indian Languages (CIIL), Mysuru.

Chapter 10

Conclusion and Future Work

After a brief study of some of the available tools for building a text synthesiser, a TTS prototype for Konkani was implemented using eSpeak. We defined the Phonetic Transcription for Konkani numbers to produce speech for Konkani numbers (0 - 100). The produced output was robotic sounding speech. To overcome the shortcomings, we developed a TTS using the basic concatenative synthesis with recorded voices. The subjective tests indicate that the word units perform better than diphone or phoneme units in concatenative synthesis.

Indian languages are syllabic in nature and based on our study and the comparison carried out for syllable based TTS systems we proceeded to develop syllabification algorithms. The forward and backward approach for syllabification was successfully developed and implemented to generate syllables for Devanagari languages. Experimental results indicate that syllable formation using the backward approach is best suited for syllabification of Devanagari words.

Our study of phonology rules for Konkani, suggest that the synthesizer should also deal with text processing rather than simply using syllabification to generate syllables. Hence the backward syllabification algorithm was enhanced to addresses the Konkani phonology rules like schwa deletion, jodaksharas, nasal words, diphthongs and vowel harmony. The output speech performs better with the syllable based concatenative approach with phonology when tested with the developed speech corpus as well as the Konkani news speech corpus

The speech corpus can be updated with a professional voice with better linguistic pronunciation and larger coverage of voice data. The system needs to be checked and compared with the other available systems for performance analysis.

There is scope to study and implement more language specific requirements, word and sentence stress, sentiments, emotions etc. to further improve the naturalness and intelligibility of the developed TTS system.

We also implemented a TTS using festival framework. The output synthesized speech obtained using festival framework exhibits good quality based on the subjective quality test results.

The system can be implemented and tested for more Devanagari languages to draw comparative results. Further, the voice quality can be improved by implementing language specific phonological rules using the festival framework.

The speech data can be increased from time to time with the availability of more news files. We also expect to get annotated, quality Konkani speech corpus from the Linguistics Data Consortium for Indian Languages (LDC-IL), Central Institute of Indian Languages (CIIL), Mysuru, Ministry of Human Resource Development (MHRD), Government of India in the near future to work towards building a robust TTS for Konkani Language. The performance of the system can be improved by enhancing the size and quality of the speech database.

Appendix A

User Manuals for TTS Systems for Konkani

1. **TTS System using eSpeak**
2. **TTS System using Concatenative Synthesis**
3. **Syllable based TTS system with Phonology**
4. **Konkani TTS System using Festival**

1. TTS System using eSpeak

A.1.1. Introduction

eSpeak is an example of speech synthesis tools used for English and other languages for Linux and Windows environment. eSpeak is a free, open-source software and is available at: espeak.sf.net/test/latest.html. For our system implementation we have used compiled eSpeak and espeakedit for Windows (espeakedit-1.48.03.exe - 5.1 MBytes).

Although a high speed, small storage and clear speech is generated by eSpeak, it is low on naturalness compared to the speech corpus based synthesizers. The eSpeak speech synthesizer uses a "formant synthesis" method and supports several languages including some of the Indian languages.

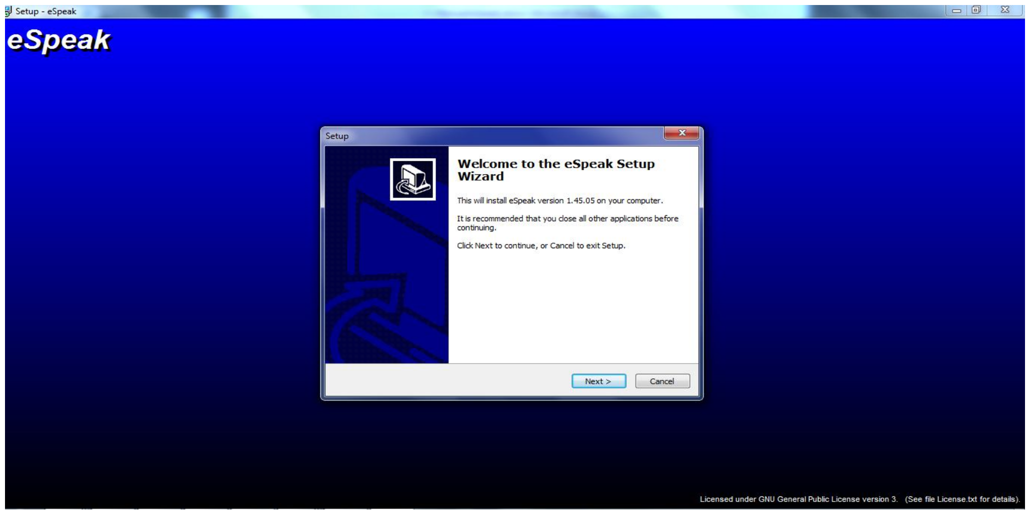
One can generate phoneme data and tune it as per the requirement using eSpeakedit and the available tools. It is now available for download with limited resources.

A.1.2. Pre-requisite

With an objective to produce Konkani speech with eSpeak, we worked on different files like ph_hindi, hi_rules and hi_list. ph_hindi is a phoneme definition file and contains phoneme definitions for the vowels and consonants for Devanagari language. hi_rules contains the spelling-to-phoneme translation rules. hi_list contains pronunciations for numbers, letter and symbol names, and words with exceptional pronunciations. These files are modified using espeakedit and are made available along with the manual.

A.1.3. Installation

In order to start eSpeak application, double click on the icon of the setup file (Fig. A1.1).



Welcome to the eSpeak Setup Wizard

This will install eSpeak version 1.45.05 on your computer.

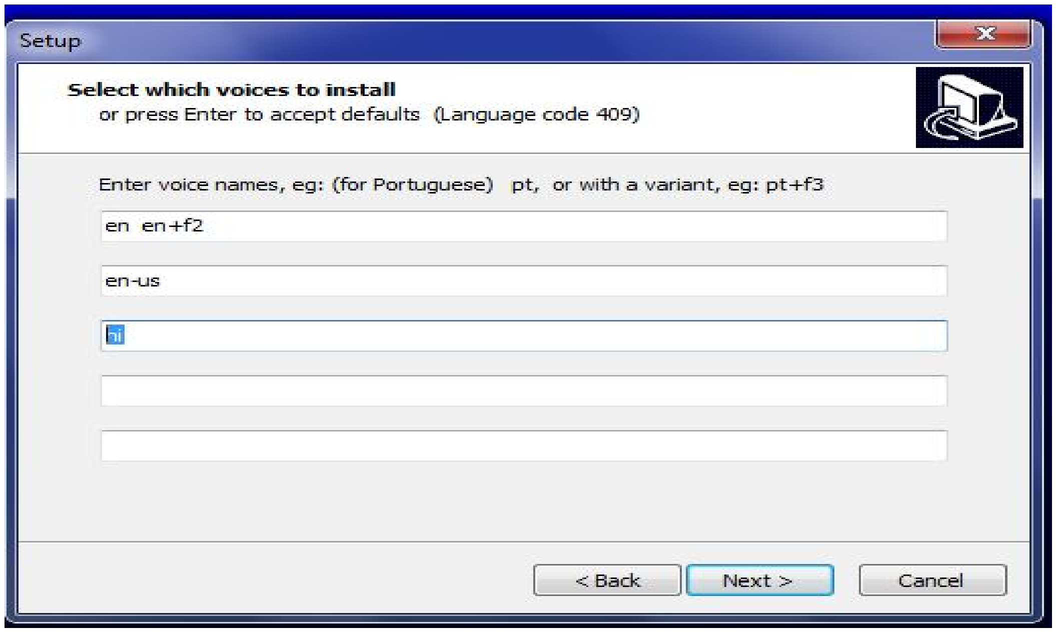
It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Licensed under GNU General Public License version 3. (See file License.txt for details).

Fig. A.1.1: eSpeak: Installation (1)

Next select the language code as two letter. Here we give 'hi' code for hindi as we modify for konkani (Fig. A.1.2).



Select which voices to install
or press Enter to accept defaults (Language code 409)

Enter voice names, eg: (for Portuguese) pt, or with a variant, eg: pt+f3

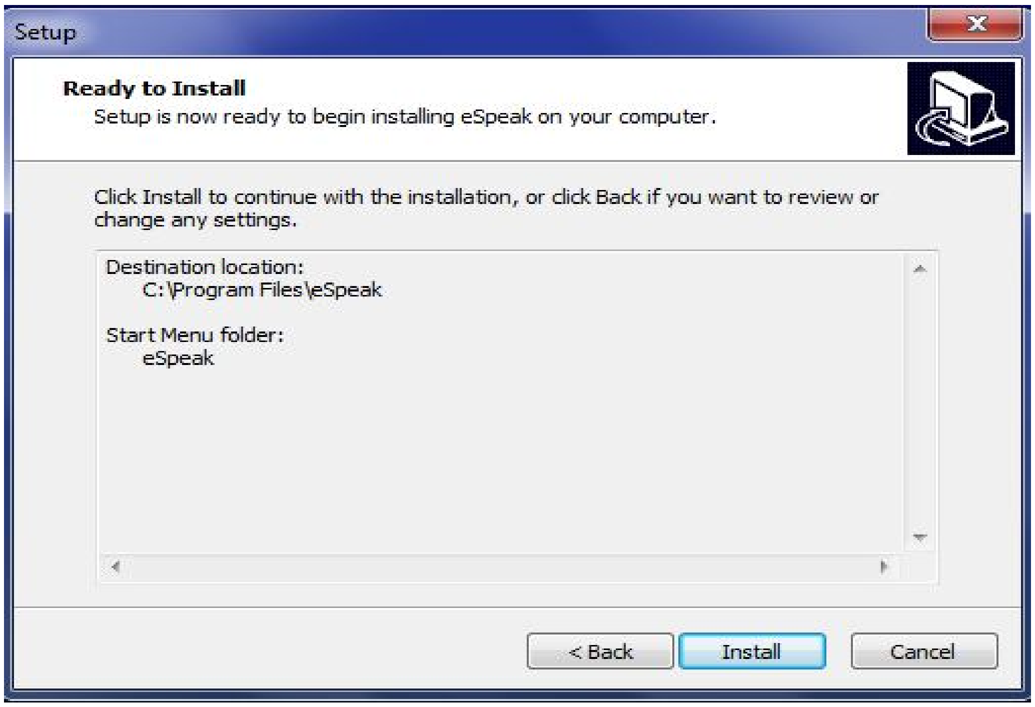
en en+f2

en-us

hi

Fig. A1.2: eSpeak: Installation - Langauge Code

Setup destination files are set in the path C:\Program Files\eSpeak.



Setup is now ready to begin installing eSpeak on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\eSpeak

Start Menu folder:
eSpeak

Fig. A.1.3: eSpeak: Installation (2)

Once the installtion is complete, eSpeak will be displayed as shown in Fig. A.1.4. Similarly install espeakedit.

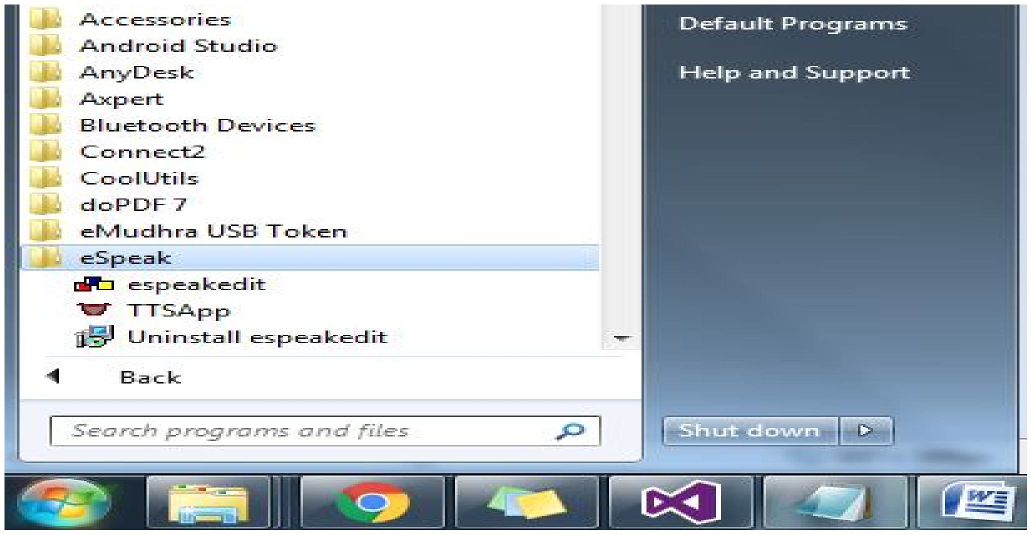
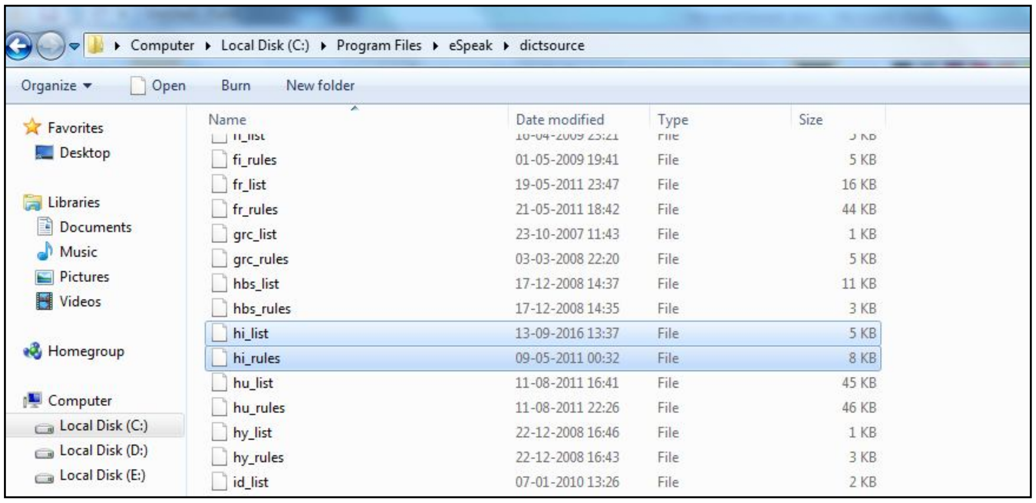


Fig. A.1.4: eSpeak: Open

Next, place the files ‘hi_list’ and ‘hi_rules’ updated in the path C:\Program Files\eSpeak\dictsource\ (Fig. A.1.5).



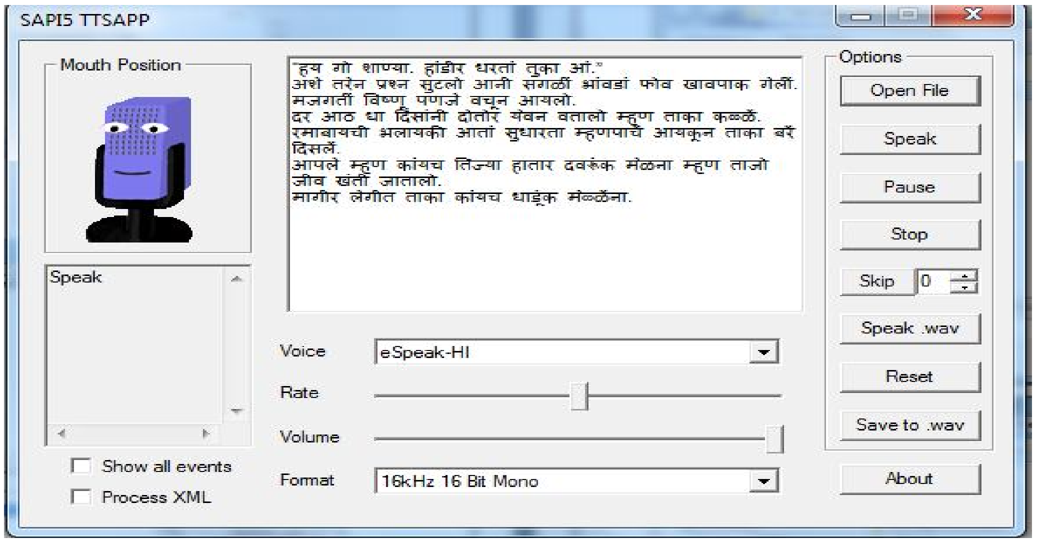
Computer ▸ Local Disk (C:) ▸ Program Files ▸ eSpeak ▸ dictsource

Name	Date modified	Type	Size
fi_list	10-04-2009 23:21	File	[illegible]
fi_rules	01-05-2009 19:41	File	5 KB
fr_list	19-05-2011 23:47	File	16 KB
fr_rules	21-05-2011 18:42	File	44 KB
grc_list	23-10-2007 11:43	File	1 KB
grc_rules	03-03-2008 22:20	File	5 KB
hbs_list	17-12-2008 14:37	File	11 KB
hbs_rules	17-12-2008 14:35	File	3 KB
hi_list	13-09-2016 13:37	File	5 KB
hi_rules	09-05-2011 00:32	File	8 KB
hu_list	11-08-2011 16:41	File	45 KB
hu_rules	11-08-2011 22:26	File	46 KB
hy_list	22-12-2008 16:46	File	1 KB
hy_rules	22-12-2008 16:43	File	3 KB
id_list	07-01-2010 13:26	File	2 KB

Fig. A.1.5: eSpeak: ‘hi_list’ and ‘hi_rules’

A.1.4. Run theTTS Application

In order to run the application, select the voice as ‘eSpeak-HI’ and enter the Devanagari text script to speak (Fig. A.1.6).



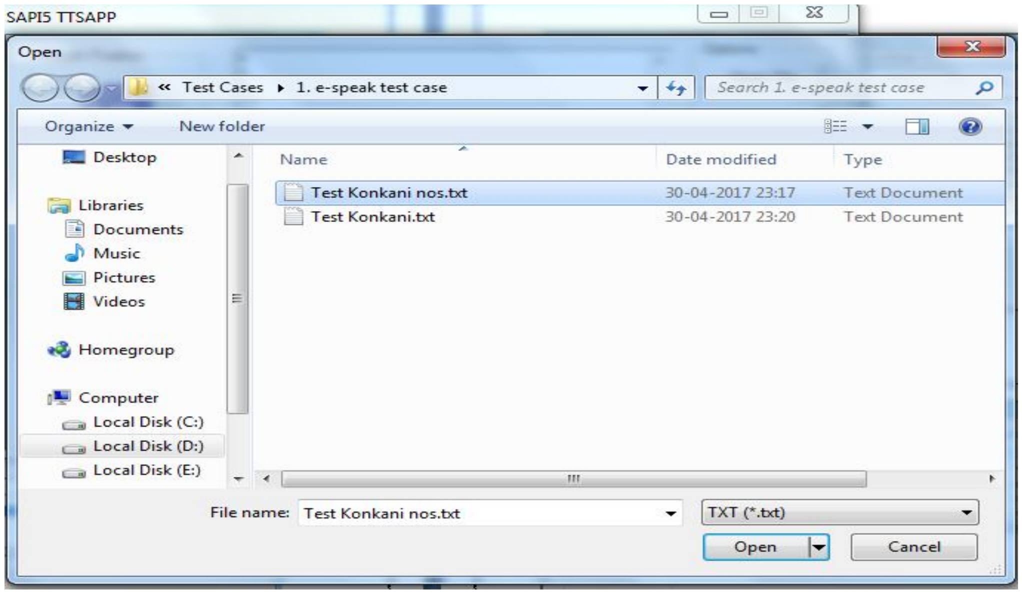
SAPI5 TTSAPP

"हय गो शाण्या. हांडीर धरतां तुका आं."
अशे तरेन प्रश्न सुटलो आनी सगळीं भोंवडां फोव खावपाक गेलीं.
मजगतीं विष्णु पणजे वचून आयलो.
दर आठ धा दिसांनी दोतोर येवन वतालो म्हूण ताका कळ्ळें.
रमाबायची भलायकी आतां सुधारता म्हणपाचें आयकून ताका बरें दिसलें.
आपले म्हूण कांयच तिज्या हातार दवरूंक मेळना म्हूण ताजो जीव खंतीं जातालो.
मागीर लेगीत ताका कांयच धाडूंक मेळ्ळेंना.

Voice: eSpeak-HI
Format: 16kHz 16 Bit Mono
Skip: 0

Fig. A.1.6: eSpeak: Interface

The User can even open a text file to be played by eSpeak. Click the open file button and choose the desired text file of Devanagari script to be played and click the play button as shown in Fig. A.1.7. eSpeak will speak the text in the file.



Name	Date modified	Type
Test Konkani nos.txt	30-04-2017 23:17	Text Document
Test Konkani.txt	30-04-2017 23:20	Text Document

File name: Test Konkani nos.txt
TXT (*.txt)

Fig. A.1.7: eSpeak: File Browse

Note: Sample Konkani text is placed in the local folder of the setup for testing.

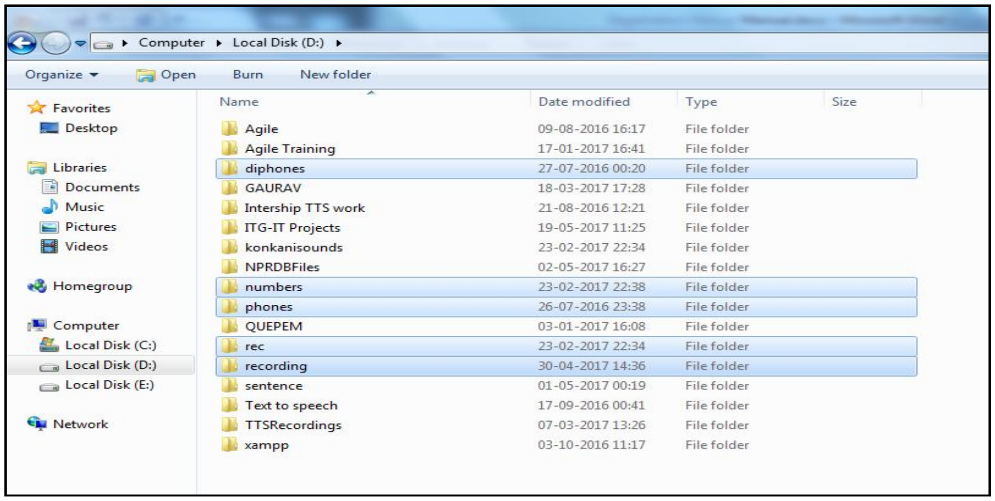
A.1.5. Test Data Sets

Two sample files containing Konkani numbers and text are placed in the local folder of the setup for testing.

2. TTS System using Concatenative Synthesis

A.2.1. Placement/Setup of TTS Corpus:

In order to run the Text To Speech(TTS) application, The recorded wave files are required. Place the wave files folder in the D:\ drive of my computer. The folder includes diphones, numbers, phones, rec and recording as shown in Fig. A2.1.

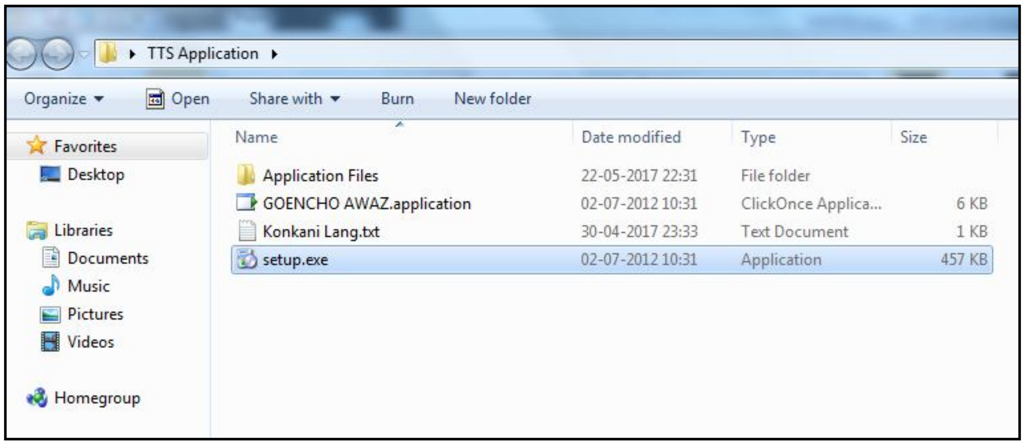


Name	Date modified	Type	Size
Agile	09-08-2016 16:17	File folder	
Agile Training	17-01-2017 16:41	File folder	
diphones	27-07-2016 00:20	File folder	
GAURAV	18-03-2017 17:28	File folder	
Intership TTS work	21-08-2016 12:21	File folder	
ITG-IT Projects	19-05-2017 11:25	File folder	
konkanisounds	23-02-2017 22:34	File folder	
NPRDBFiles	02-05-2017 16:27	File folder	
numbers	23-02-2017 22:38	File folder	
phones	26-07-2016 23:38	File folder	
QUEPEM	03-01-2017 16:08	File folder	
rec	23-02-2017 22:34	File folder	
recording	30-04-2017 14:36	File folder	
sentence	01-05-2017 00:19	File folder	
Text to speech	17-09-2016 00:41	File folder	
TTSRecordings	07-03-2017 13:26	File folder	
xampp	03-10-2016 11:17	File folder	

Fig. A.2.1: Concatenative TTS: Wave File Folder

A.2.2. Installation of TTS Application

In order to start the TTS application, double click on the icon of the setup file (Fig. A.2.2)



Name	Date modified	Type	Size
Application Files	22-05-2017 22:31	File folder	
GOENCHO AWAZ.application	02-07-2012 10:31	ClickOnce Applica...	6 KB
Konkani Lang.txt	30-04-2017 23:33	Text Document	1 KB
setup.exe	02-07-2012 10:31	Application	457 KB

Fig. A.2.2: Concatenative TTS: Setup

The following application screen will appear which shows text window, ‘PLAY’ and ‘Open File’ button (Fig. A.2.3).

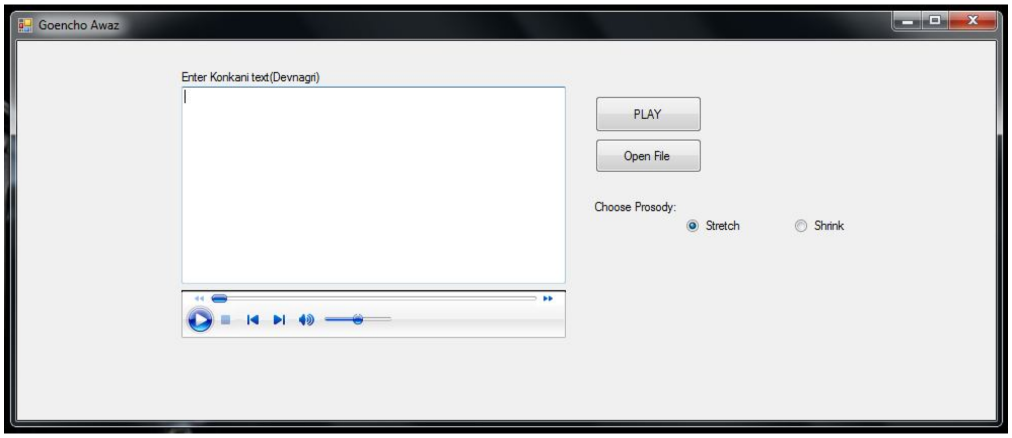
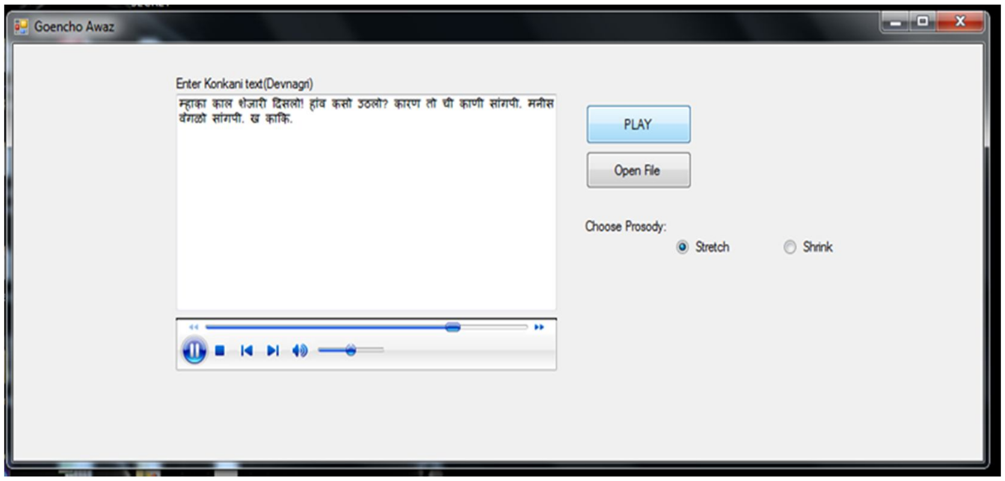


Fig. A.2.3: Concatenative TTS: Front End

A.2.3. Run theTTS Application

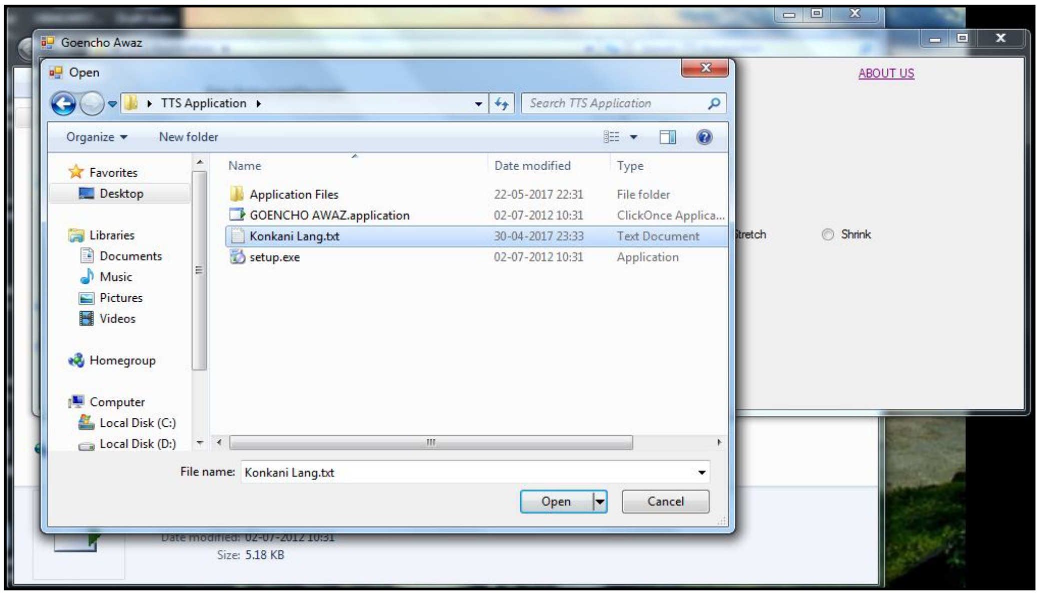
In order to run the application, type text the devanagri text in the textbox shown below and click the ‘PLAY’ button. The windows media player will play the concatenated wave files stored in the sound database. The player contains various controls such as pause / play, volume control, next and previous (Fig. A.2.4).



म्हाका काल शेजारी दिसलो! हांव कसो उठलो? कारण तो घी काणी सांगपी. मनीस वेगळो सांगपी. ख काफि.

Fig. A.2.4: Concatenative TTS: Text Entered

The User can even open a text file to be played by the TTS application. Click the open file button and choose the desired text file of Devanagari script to be played and click the play button as shown in Fig. A.2.5. The TTS application will speak the text in the file.



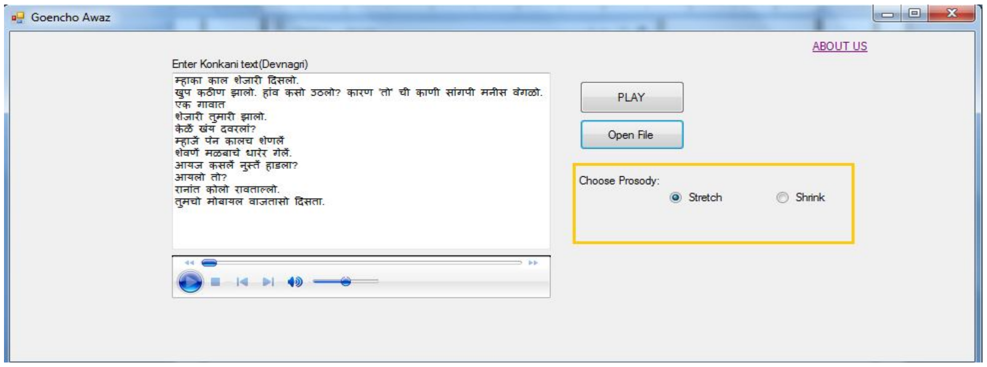
Name	Date modified	Type
Application Files	22-05-2017 22:31	File folder
GOENCHO AWAZ.application	02-07-2012 10:31	ClickOnce Applica...
Konkani Lang.txt	30-04-2017 23:33	Text Document
setup.exe	02-07-2012 10:31	Application

File name: Konkani Lang.txt

Date modified: 02-07-2012 10:31
Size: 5.18 KB

Fig. A.2.5: Concatenative TTS: File Browse

For prosody, if there is an exclamation mark or a question mark in the sentence, then prosody can be chosen appropriately for all the text in the textbox by selecting either stretch or shrink radio buttons (available for words stored in the database only) as shown in the Fig. A.2.6.



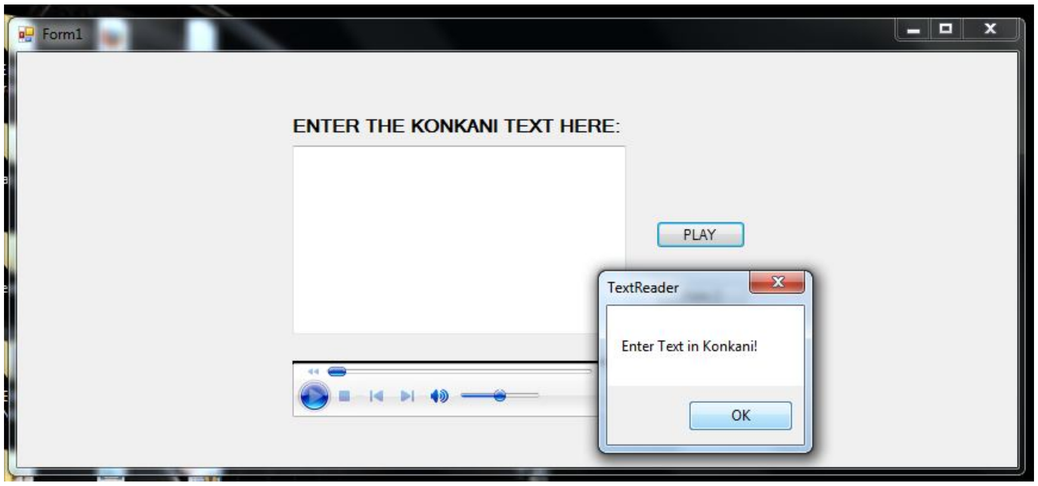
Enter Konkani text(Devnagri)

म्हाका काल शेजारी दिसलो.
खुप कठीण झालो. हांव कसो उठलो? कारण 'तो' ची काणी सांगपी मनीस वेगळो.
एक गावात
शेजारी तुमारी झालो.
केळें खंय दवरलां?
म्हाजें पेन कालच शेणलें
शेवणें मळबाचे धारेर गेलें.
आयज कसलें नुस्तें हाडला?
आयलो तो?
रानांत कोलो रावताल्लो.
तुमचो मोबायल वाजतासो दिसता.

Choose Prosody: [✓ Stretch] Shrink

Fig. A.2.6: Concatenative TTS: Prosody

If the user does not enter text and clicks the play button, the system will report an error and will prompt to enter text as shown in Fig. A.2.7.



Enter Text in Konkani!

Fig. A.2.7: Concatenative TTS: Validation

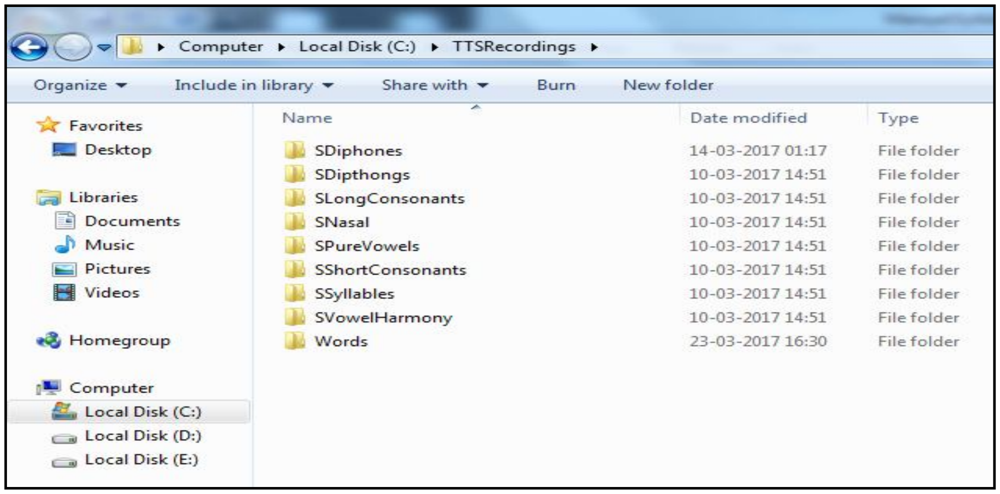
A.2.4. Test Data Set

Two Sample files containing Konkani numbers and text are placed in the local folder of the setup for testing.

3. Syllable Based Concatenative TTS system with Phonology

A.3.1. Placement/Setup of TTS Corpus

In order to run the Text To Speech(TTS) application, the wave files are required. Place the wave files folder “TTSRecordings”in the C:\ drive of my computer as shown in Fig. A.3.1.



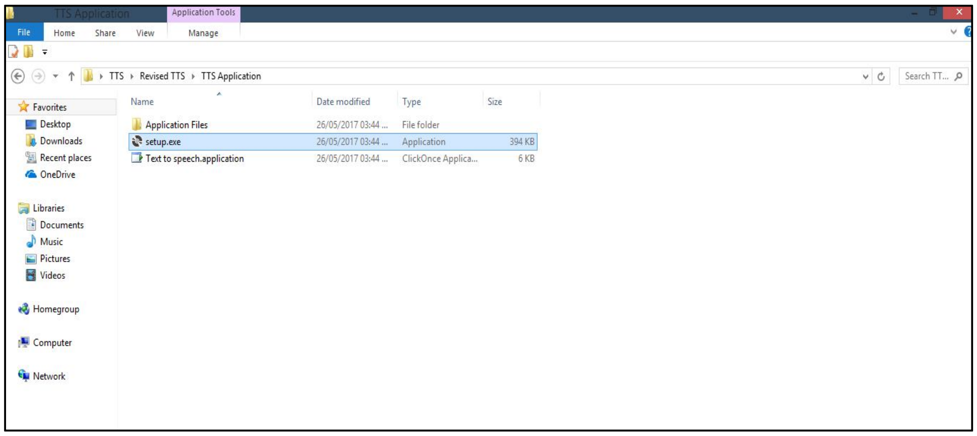
Computer ▸ Local Disk (C:) ▸ TTSRecordings ▸

Name	Date modified	Type
SDiphones	14-03-2017 01:17	File folder
SDipthongs	10-03-2017 14:51	File folder
SLongConsonants	10-03-2017 14:51	File folder
SNasal	10-03-2017 14:51	File folder
SPureVowels	10-03-2017 14:51	File folder
SShortConsonants	10-03-2017 14:51	File folder
SSyllables	10-03-2017 14:51	File folder
SVowelHarmony	10-03-2017 14:51	File folder
Words	23-03-2017 16:30	File folder

Fig. A.3.1: Syllable based TTS: Wave File Folder

A.3.2. Installation of TTS Application

In order to start the TTS application, double click the icon of the setup file (Fig. A.3.2).



TTS ▸ Revised TTS ▸ TTS Application

Name	Date modified	Type	Size
Application Files	26/05/2017 03:44 ...	File folder	
setup.exe	26/05/2017 03:44 ...	Application	394 KB
Text to speech.application	26/05/2017 03:44 ...	ClickOnce Applica...	6 KB

Fig. A.3.2: Syllable based TTS: Setup

The following application screen will appear which shows the text window, ‘PLAY’ and ‘Upload File’ button (Fig. A.3.3).

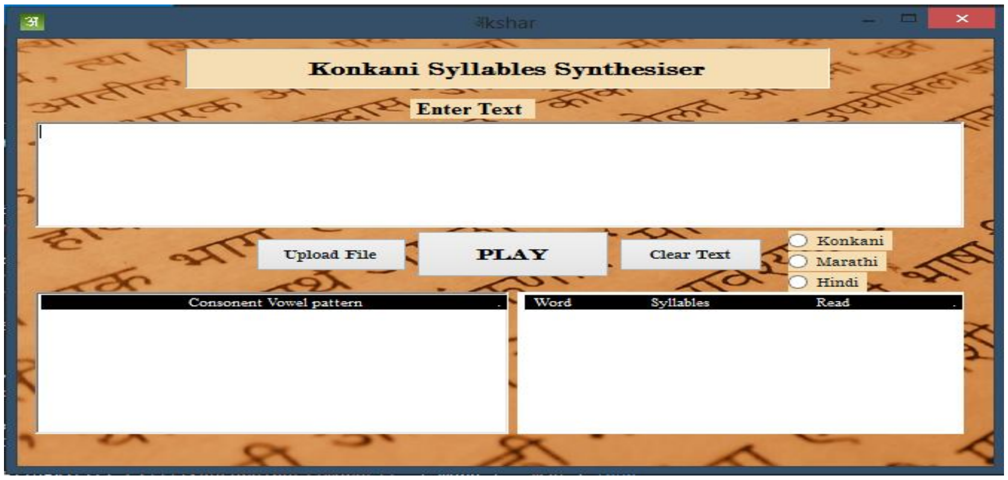
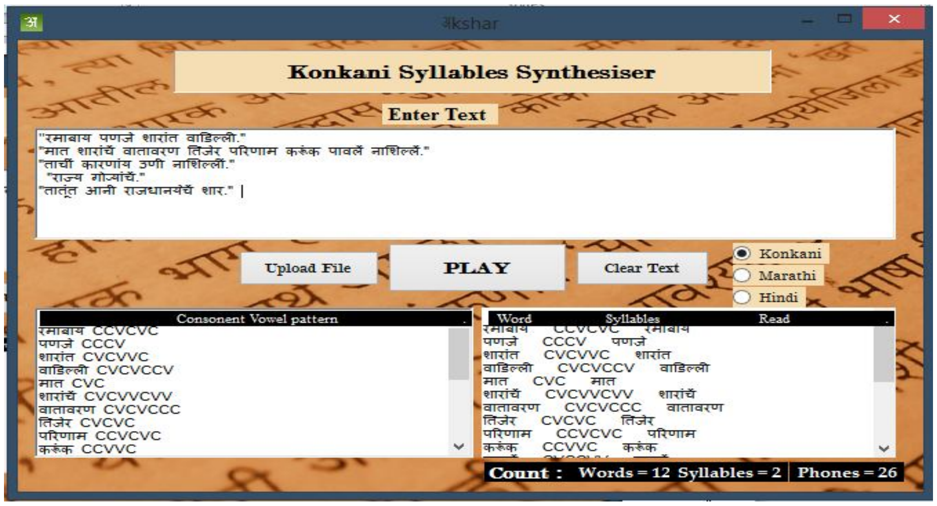


Fig. A.3.3: Syllable based TTS: Front End

A.3.3. Run theTTS Application

In order to run the application, type the Devanagri text in the textbox as shown below and click the ‘PLAY’ button. The language option (Konkani, Marathi and Hindi) is to choose the appropriate language. The system will play the concatenated wave files stored in the sound database (refer Fig. A.3.4).



[✓ Konkani] Marathi Hindi

"रमाबाय पणजे शारांत वाडिल्ली."
"मात शारांचें वातावरण तिजेर परिणाम करूंक पावलें नाशिल्लें."
"ताचीं कारणांय उणी नाशिल्लीं."
"राज्य गोंयांचें."
"तातूंत आनी राजधानयेचें शार."

Consonent Vowel pattern

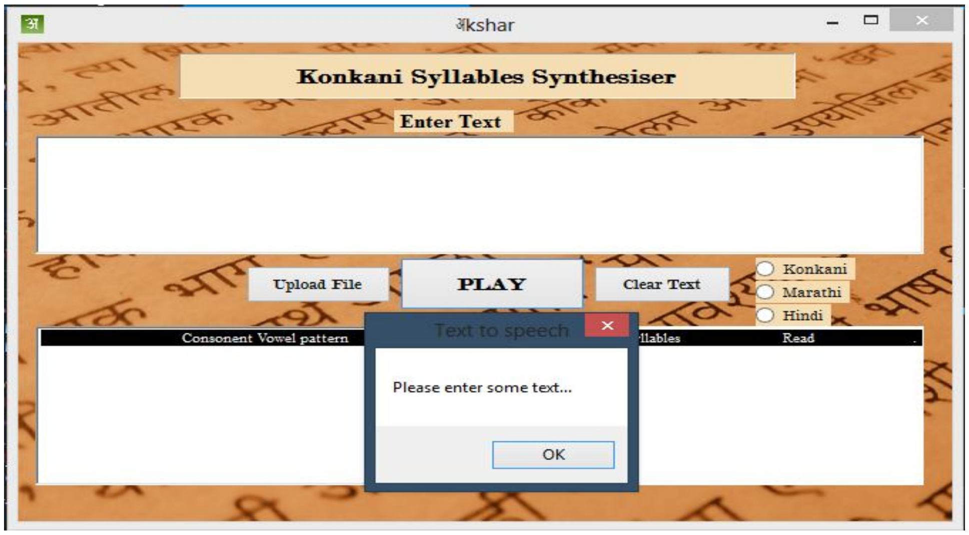
```
रमाबाय CCVCVC
पणजे CCCV
शारांत CVCVVC
वाडिल्ली CVCVCCV
मात CVC
शारांचें CVCVVCVV
वातावरण CVCVCCC
तिजेर CVCVC
परिणाम CCVCVC
करूंक CCVVC
```

Word	Syllables	Read
रमाबाय	CCVCVC	रमाबाय
पणजे	CCCV	पणजे
शारांत	CVCVVC	शारांत
वाडिल्ली	CVCVCCV	वाडिल्ली
मात	CVC	मात
शारांचें	CVCVVCVV	शारांचें
वातावरण	CVCVCCC	वातावरण
तिजेर	CVCVC	तिजेर
परिणाम	CCVCVC	परिणाम
करूंक	CCVVC	करूंक

Count : Words = 12 Syllables = 2 | Phones = 26

Fig. A.3.4. Syllable based TTS: Text Entered and Language

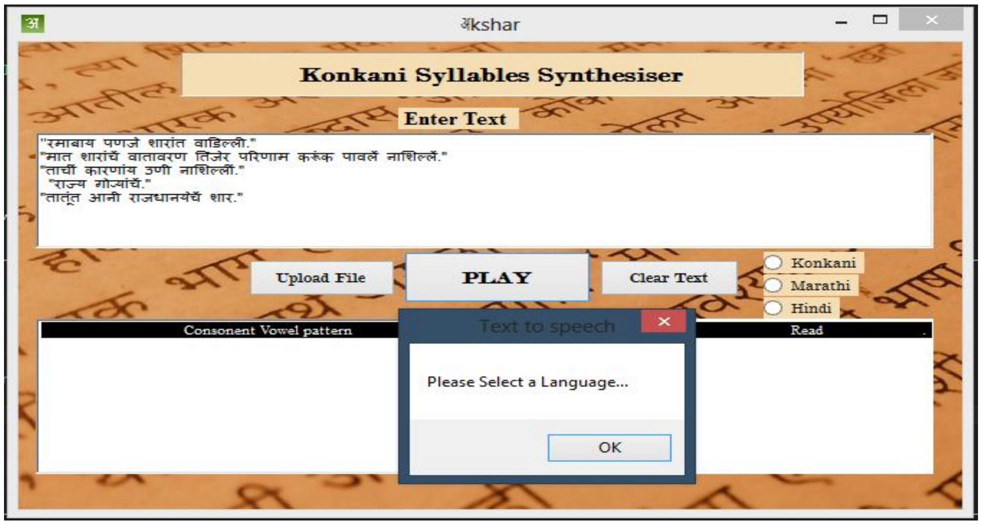
If the user does not enter text and the play button is clicked, the system will prompt to enter text as shown in Fig. A.3.5..



Please enter some text...

Fig. A.3.5: Syllable based TTS: Validation (1)

If the user does not select any language and the play button is clicked, the system will prompt to enter text as shown in Fig. A.3.6..

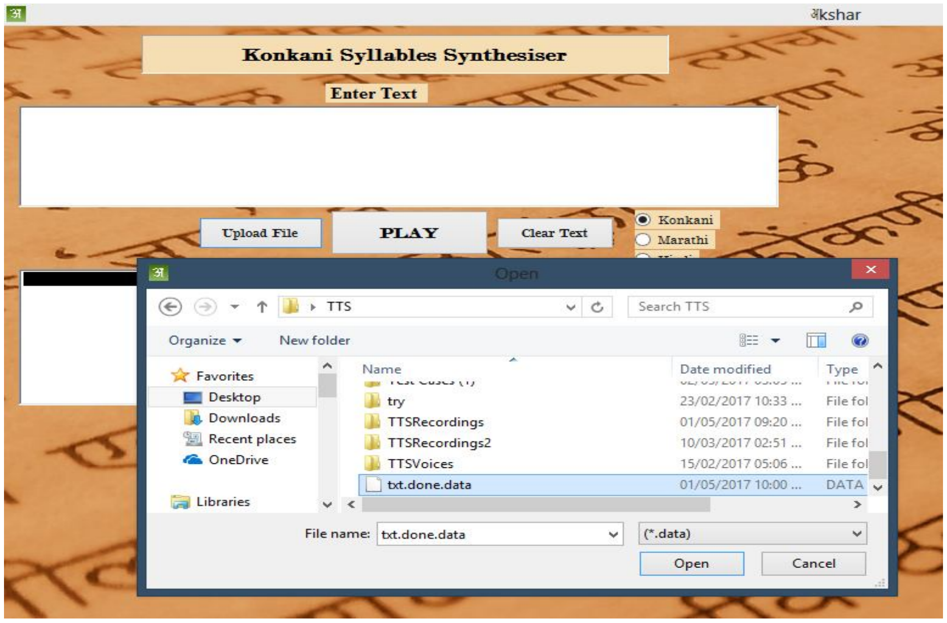


"रमाबाय पणजे शारांत वाडिल्ली."
"मात शारांचें वातावरण तिजेर परिणाम करूंक पावलें नाशिल्लें."
"ताचीं कारणांय उणी नाशिल्लीं."
"राज्य गोंयांचें."
"तातूंत आनी राजधानयेचें शार."

Please Select a Language...

Fig. A.3.6: Syllable based TTS: Validation (2)

The User can even open a text / pdf / word file to be played by the TTS, click the open file button and choose the desired Devanagari text file to be played and click the play button as shown in Fig. A.3.7. The TTS will read the text in the file.



Name	Date modified	Type
try	23/02/2017 10:33 ...	File fol
TTSRecordings	01/05/2017 09:20 ...	File fol
TTSRecordings2	10/03/2017 02:51 ...	File fol
TTSVoices	15/02/2017 05:06 ...	File fol
txt.done.data	01/05/2017 10:00 ...	DATA

File name: txt.done.data (*.data)

Fig. A.3.7: Syllable based TTS: File Browse

A.3.4. Test Data Sets

Sample Konkani text file is placed in the local folder of the setup for testing.

4. TTS System using Festival

A.4.1. Pre-requisite

Festival framework works on Linux platform. For our system implementation, we have chosen BackTrack 5, which is Linux distribution and an open source software. BackTrack provided users with easy access to a comprehensive and large collection of security-related tools.

A.4.2. Installation of Festival Framework

(i) How to Install:

Edinburgh Speech Tools provides a set of executables, which offer access to speech tools functionality in the form of a standalone program. Some of the common features of these tools are Arguments to functions can be specified in any order on the command line.

Installation of festival speech synthesis system tool requires the sources of festival framework and speech tools. Festvox project is another framework to generate the new voice models from the recorded speech data base, developed in CMU (Carnegie Mellon University).

To generate voice models and use those models to synthesize the input text for Indic speech database, current versions of festival, speech tools and festvox frame work are used. Download festival and speech tools from the website: (http://www.cstr.ed.ac.uk/downloads/festival/). We can download the festvox from the link: (http://www.festvox.org/).

- festival-2.4-release.tar.gz
- speech _tools 2.4.tar.gz
- festvox-2.7.0-release.tar.gz

We should install simple test suite with festival but it requires the three basic voices and their respective lexicons installed before it will work, that is available in the above link

- festvox_kallpc16k.tar.gz
- festlex_POSLEX.tar.gz

- festlex_CMU.tar.gz

Further, Extract festival, speech_tools, and festvox to your home directory and set up the environment variables.

Extract all by using the command: $ tar –zxvf [file name]

$tar -zxvf speech_tools-2.4.tar.gz

Now we can follow the steps:

$cd speech_tools

$./configure

$make info

$export ESTDIR=home/speech_tools

echo $ESTDIR

$make

If everything works properly then we have to configure festival.

$tar -zxvf festival-2.4-release.tar.gz

Now follow the steps:

$cd festival

$./configure

$make info

$make

$export FESTDIR=home/festival

echo $ESTDIR

Then extract the following files, with commands:

$tar –xvzf festlex_CMU.tar.gz

$tar –xvzf festlex_POSLEX.tar.gz

$tar –xvzf festlex_festvox_ kallpc16k.tar.gz

$sudo ln-s /home/festival/src/main/festival/usr/bin/festival

(ii) How to Build

Here we assume festival works perfectly, and then extract festvox by command:

$tar –zxvf festvox-2.7.0-release.tar

$cd festvox

$./configure

$make info

$make

$export FESTVOXDIR=home/festvox

$mkdir demo

$cd demo

$FESTVOXDIR/src/unitsel/setup_clunits demo

$FESTVOXDIR/src/prosody/setup_prosody

After executing the above command we will get the whole branch of directory. Make txt.done.data file with the following format and keep it in demo/etc/ directory:

```
( data_00001 "रमाबाय पणजे शारांत वाडिल्ली." )
( data_00002 "शारांचें वातावरण तिजेर परिणाम करूंक पावलें नाशिल्लें." )
( data_00003 "ताचीं कारणांय उणी नाशिल्लीं." )
( data_00004 "राज्य गोऱ्यांचें." )
( data_00005 "तातूंत आनी राजधानयेचें शार." )
```

The format is "(" followed by a filename, root followed by the text for that sentence, followed by ")" each on separate lines. This text when converted to a phone sequence by festival should match (as closely as possible) the phone sequence of the speech. With this in mind we should probably ensure all words are in your lexicon (if we are using one) and it is probably best to write numbers and dates out in full as they were spoken.

$festival -b festvox/build_clunits.scm '(build_prompts "etc/txt.done.data")'

The next command (prompt_them) is used for voice recording. We can record our voice in linux system by the following command. Advantage is that we do not have to label of our voice. But we have to careful about our voice recording. Recording voice will be automatically saved in /home/demo/wav directory. After recording of our voice, we have to test that, our recording voice and txt.done.data sentences matches perfectly.

$bin/prompt_them etc/txt.done.data

Execute the following command sequentially.

$FESTVOXDIR/src/ehmm/bin/do_ehmm setup
$FESTVOXDIR/src/ehmm/bin/do_ehmm phseq
$FESTVOXDIR/src/ehmm/bin/do_ehmm feats
$FESTVOXDIR/src/ehmm/bin/do_ehmm bw
$FESTVOXDIR/src/ehmm/bin/do_ehmm align
$bin/make_pm_wave wav/*.wav
$bin/make_pm_fix pm/*.pm
$bin/make_pmlab_pm pm/*.pm
$bin/make_pm_pmlab pm_lab/*.pm
$bin/make_mcep wav/*.wav
$festival -b festvox/build_clunits.scm '(build_utts "etc/txt.done.data")'
$festival -b festvox/build_clunits.scm '(build_clunits "etc/txt.done.data")'

Once the above set of commands is successfully executed, rename the voice folder as demo_clunits. Then copy the voice folder to festival/lib/voices.

Note: The recorded text/voice files for Marathi and Konkani trained data requires to be configured separately.

A.4.3. Run the Application

In order to run festival, open terminal and execute the following commands sequentially:

$festival festvox/demo_clunits.scm
festival festvox/demo
festival>(voice_demo_clunits)
festival>(tts “Konkani.txt” nil)

The above command will read out the input text file”Konkani.txt”. Place the text file in the local folder demo.

A.4.4. Test Data Sets:

Sample Konkani text file is placed in the local folder of the setup for testing.

Appendix B

Form for Performance Evaluation of TTS Systems

Performance Evaluation of the Konkani TTS Systems

Name		**Designation**	
Address		**Age (in years)**	
Date		**Signature**	

Please refer to (A) and (B) and give your Evaluation Scores on the scale of 1 (Bad) to 5 (Excellent) for 5 different sentences: tick (√) the appropriate

(1) eSpeak Konkani TTS System	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					

(1) Festival Konkani TTS System: Training with Marathi Speech Corpus	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					

Testing the TTS Systems with Konkani Speech Corpus 1 (Recorded)

(1) Basis Concatenative Konkani TTS System	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					
(2) Syllable Based Concatenative Konkani TTS System	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					
(3) Syllable Based Concatenative Konkani TTS System with Phonology	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					

(4) Festival TTS system: Training with Konkani Speech Corpus 1	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					

Testing the TTS Systems with Konkani Speech Corpus 2 (Konkani News)

(1) Basis Concatenative Konkani TTS System	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					
(2) Syllable Based Concatenative Konkani TTS System	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					
(3) Syllable Based Concatenative Konkani TTS System with Phonology	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					
(5) Festival TTS system: Training with Konkani Speech Corpus 2	**Sent. No**	**1**	**2**	**3**	**4**	**5**
Naturalness (How closely the output sounds like human speech)	1					
	2					
	3					
	4					
	5					
Intelligibility (the ease with which the output is understood)	1					
	2					
	3					
	4					
	5					

(A) Scores for Naturalness

Rating	**Quality**	**Description**
5	Excellent	Imperceptible
4	Good	Just perceptible but not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying but not objectionable
1	Bad	Very annoying and objectionable

(B) Scores for Intelligibility

Rating	**Quality**	**Description**
5	Excellent	Complete relaxation possible; no effort required
4	Good	Attention necessary; no appreciable effort required
3	Fair	Moderate effort required
2	Poor	Considerable effort required
1	Bad	No meaning understood with any feasible effort

Appendix C

Publications

1. Nilesh B. Fal Dessai, Gaurav A. Naik, Jyoti D. Pawar, “Review of Syllable based Text to Speech Systems: Strategies for enhancing naturalness for Devanagari languages”, International Journal of Computer Science and Applications (IJCSA) 2017, Technomathematics Research Foundation.
2. Nilesh B. Fal Dessai, Gaurav A. Naik, Jyoti D. Pawar, “Implementation of a TTS System for Devanagari Konkani Language using Festival”, International Journal of Advanced Research in Computer Science (IJARCS), Volume 8, Issue 5, May - June 2017, ISSN 0976 – 5697, PP. 386-391.
3. Nilesh B. Fal Dessai, Gaurav A. Naik, Jyoti D. Pawar, “Syllabification: An Effective Approach for a TTS System for Konkani” Proceedings of IEEE International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), December 2016, Mysuru, PP. 36.
4. Nilesh B. Fal Dessai, Gaurav A. Naik, Jyoti D. Pawar, “Development of Konkani TTS System using Concatenative Synthesis” Proceedings of the IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 16-17 December 2016, Kanyakumari, PP. 425- 429.

Bibliography

[1]. Masaaki Honda, “Human Speech Production Mechanism”, Vol 1, Mo2, NNT Technical Review, May 2003.

[2]. Mustafa Zeki, Othman O. Khalifa, A. W. Naji, “Development of an Arabic Text-to-Speech System”, International Conference in Computer and Communication Engineering (ICCCE), Kuala Lumpur, May 2010.

[3]. Thierrry Dutoit, “High Quality Text to Speech Synthesis: an Overview”, Journal of Electrical & Electronics Engineering, 1997.

[4]. M. Waseem, C. N. Sujatha, “Speech Synthesis System for Indian Accent using Festvox”, International Journal of Scientific Engineering and Technology Research, Issue 34, pp. 6903-6911, November 2014.

[5]. N. S. Krishna, H. A. Murthy, T. A. Gonsalves, “Text-to-Speech in Indian Languages”, Proceedings of International Conference on Natural Language Processing, ICON 2002, pp. 317-326, Mumbai, 2002.

[6]. D. H. Klatt, “The Klattalk text-to-speech conversion system in Acoustics, Speech, and Signal Processing”, IEEE International Conference on ICASSP '82, pp. 1589 – 1592, 1982.

[7]. M. Nageshwara Rao, Samuel Thomas, T. Nagarajan, Hema A. Murthy, “Text-to-Speech Synthesis using Syllable-like Units”, Proceedings of National Conference on Communication (NCC), IIT Kharagpur, pp. 227-280, 2005.

[8]. S. Saraswathi, R. Vishalakshy, “Design of Multilingual Speech Synthesis System in Information Management”, Vol. 2, pp. 58-64, 2010.

[9]. Suhas R. Mache, Manasi R. Baheti, C. Namrata Mahender, “Review on Text-To-Speech Synthesizer”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 8, pp. 54-58, August 2015.

[10]. Languages of India, February 2012,
[Online]: http://www.webindia123.com/india/people/language.htm

[11]. Priyadarshani S. Tadkodkar and Asha A. Mangutkar, "Text Processing for Text to Speech Synthesis in Indian languages by Konkani Parichay" book published in 2009.

[12]. Acharya Multilingual Computing for Literacy and Education, SDL, IIT Madras,
[Online]: http://www.acharya.gen.in:8080/sanskrit/script_dev.php

[13]. Sangam P. Borkar, "Text to Speech System for Konkani (Goan) Language".

[14]. Shanai Goibab, "Konkani Nadshahtra", Vol. 1, pp 392-439, 1940.

[15]. Konkani Language [Online]: https://en.wikipedia.org/wiki/Konkani_language

[16]. Subaryani D. H. Soedirdjo, Harshallah Zakaria, Richard Mengko, "Indonesian Text-to-Speech using Syllable Concatenation for PC-based low vision Aid", International Conference on Electrical Engineering and Informatics, Indonesia, July 2011.

[17]. Fatima Chouireb M. G., "Towards a High Quality Arabic Speech Synthesis System Based on Neural Networks and Residual Excited Vocal Tract Model", Signal, Image and Video Processing, Vol2, Issue 1, pp. 73-87, Springer, 2008.

[18]. S. Lemmetty, "Review of Speech Synthesis Technology", Master Thesis, Helsinki University of Technology, 1999.

[19]. Sandesh Aryal, Ricardo Gutierrez-Osuna, "Articulatory inversion and synthesis: Towards articulatory-based modification of speech", IEEE International Conference on Acoustics, Speech and Signal Processing, 2013

[20]. Alan Bundy, Lincoln Wallen, "Formant Synthesis", Catalogue of Artificial Intelligence Tools Part of the series Symbolic Computation, pp 37-37. Chapter in symbolic computation book series, Springer, 1984.

[21]. Diemo Schwarz, "Corpus-Based Concatenative Synthesis", IEEE Signal Processing Magazine, 2009

[22]. Md. Rafiqul Islam, Ram Shanker Saha, Ashif Rubayat Hossain, “Automatic Reading from Bangla PDF Document Using Rule Based Concatenative Synthesis”, International Conference on Signal Processing Systems, 2009.

[23]. Andrew J. Hunt, Alan W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database”, IEEE international Conference on Accoustics, Speech and Signal Processing, Atlanta, 1996.

[24]. Kevin A. Lenzo, Alan W Black, “Diphone Collection and Synthesis”, Proceedings of Sixth International Conference on Spoken Language Processing, ICSLP 2000, Beijing, China, Oct 2000.

[25]. Itunuoluwa Isewon, Jelili Oyelade, Olufunke Oladipupo, “Design and Implementation of Text To Speech Conversion for Visually Impaired People”, International Journal of Applied Information Systems, Foundation of Computer Science FCS, New York, USA Volume 7– No. 2, April 2014.

[26]. Sangramsing Kayte1, Monica Mundada1, Charansing Kayte, “Di-phone-Based Concatenative Speech Synthesis Systems for Marathi Language”, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), Sep - Oct. 2015.

[27]. P. Chaudhury, M. Rao, K. V. Kumar, “Symbol Based Concatenation Approach for Text to Speech System for Hindi using Vowel Classification Technique”, World Congress on Nature and Biologically Inspired Computing, pp.1082 – 1087, 2009.

[28]. Chauhan, V. Chauhan, S. P. Singh, A. K. Tomar, H. Chauhan, “A Text to Speech System for Hindi using English Language”, International Journal of Computer Science and Technology, Vol. 2, Issue 3, pp. 322-326, 2011.

[29]. Ashwin Bellur, K. Badri Narayan, Raghava Krishnan K., Hema A. Murthy, “Prosody Modelling for Syllable-Based Concatenative Speech Synthesis of Hindi and Tamil”, Proceedings of National Conference on Communications, 2011.

[30]. Gurpreet Kaur, Parminder Singh, “A Technique to Detect Syllable Boundary in a Wave File”, International Journal of Computer Science and Communication Engineering, IJCSCE Special issue on “Recent Advances in Engineering & Technology” NCRAET 2013.

[31]. Stas Tiomkin, David Malah, Slava Shechtman, Zvi Kons, “A Hybrid Text-to-Speech System That Combines Concatenative and Statistical Synthesis Units”, IEEE Transactions on Audio, Speech, and Language Processing, Volume 19, Issue 5, pp. 1278-1288, July 2011.

[32]. Mustafa Cem Orhan, Cenk Demirogl, “HMM-based text to speech system with speaker interpolation”, 19th Conference on Signal Processing and Communications Applications (SIU), IEEE, 2011

[33]. Hungyan Gu, Yenzuo Zhou, “Mandarin syllable signal synthesis using an HNM based scheme”, International Conference on Audio, Language and Image Processing, 2008.

[34]. K. Pavan Kumar, “Speech Synthesis Based on Sinusoidal Modelling”, M.Tech. Credit Seminar Report, Electronic Systems Group, EE Dept, IIT Bombay, November 2004.

[35]. Hermansky, H. Fujisaki and Y. Sato, “Analysis and Synthesis of Speech Based on Spectral Transform Linear Predictive Method,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Boston, MA, 1983.

[36]. eSpeak: Speech Synthesizers”, [Online]: http://espeak.sourceforge.net/

[37]. S. Thottingal, “Dhvani Indian Language Text to Speech System”, [Online]: http://foss.in/2007/register/slides/Dhvani_Indian

[38]. R. Hariharan, “Text to Speech System Demo, Adding Language support, Usage”. [Online]: http://dhvani.sourceforge.net

[39]. “Vani: An Indian Language text to speech synthesizer”, [Online]: http://www.cse.iitb.ac.in/vani/

[40]. H. Jain, V. Kanade, G. Sivakumar, "Design of a Text to Speech Synthesizer to Generate Arbitrary Speech", International Conference on Speech and Language Technology, Noida, 2004.

[41]. Maria Astrinaki, Alexis Moinet, Junichi Yamagishi, Korin Richmond, Zhen-Hua Ling, Simon King, Thierry Dutoit, “Mage - Reactive articulatory feature control of HMM-based parametric speech synthesis”, 8th ISCA Speech Synthesis Workshop, Barcelona, Spain 2013.

[42]. Festival Speech Synthesis, [Online]:http://www.cstr.ed.ac.uk/projects/festival/

[43]. Festvox, [online]:http://festvox.org/bsv/x262.html

[44]. Audacity Tool, [Online]: http://www.audacityteam.org

[45]. Wavesurfer Tool, [Online]: https://en.wikipedia.org/wiki/WaveSurfer

[46]. Wavesurfer [online]: http://www.wavesurfer.org/

[47]. "Text to Speech Testing Strategy", Technology Development for Indian Languages (TDIL) Programme, Department of Electronics and Information Technology (DeitY), Government of India, Version 2.1, 07 July, 2014.

[48]. P. C. Loizou, "Speech Quality Assessment", University of Texas-Dallas, Department of Electrical Engineering, Richardson, TX, USA, SCI 346, pp. 623-654.

[49]. J. O. Onaolapo, F. E. Idachaba, J. Badejo, T. Odu, and O. I. Adu, "A Simplified Overview of Text-To-Speech Synthesis", Proceedings of the World Congress on Engineering (WCE), Vol I, London, U.K, July 2014.

[50]. Mentor Hamiti, Ramiz Kastrati, "Adapting eSpeak for converting text into speech in Albanian", International Journal of Computer Science Issues (IJCSI), Vol. 11, Issue 4, No 2, pp. 21-26, July 2014.

[51]. Ramanpreet Kaur, Dharamveer Sharma, "An Improved System for Converting Text into Speech for Punjabi Language using eSpeak", International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue 04, pp. 500-504, April 2016.

[52]. Mahwash Ahmed, Shibli Nisar, "Text-to-Speech Synthesis using Phoneme Concatenation", International Journal of Scientific Engineering and Technology, Volume No.3 Issue No.2, pp. 193 – 197, February 2014 .

[53]. Tapas Kumar Patra, Biplab Patra, Puspanjali Mohapatra, "Text to Speech Conversion with Phonematic Concatenation", International Journal of Electronics Communication and Computer Technology (IJECCT), Volume 2, Issue 5, September 2012.

[54]. Sangramsing N. Kayte, Bharti Gawali, “The Marathi Text-To-Speech Synthesizer Based On Artificial Neural Networks”, International Research Journal of Engineering and Technology (IRJET), Volume 02, Issue 08, pp. 948-953, November 2015.

[55]. John Colaco, Sangam Borkar, “Design and Implementation of Konkani Text to Speech Generation System using OCR Technique”, Imperial Journal of Interdisciplinary Research (IJIR), Vol 2, Issue 6, pp. 1162-1167, 2016.

[56]. O‘Malley M. H., “Text to Speech conversion Technology”, in Computer, Vol 23, pp. 17-23, 1990.

[57]. S. D. Shirbahadurkar, D. S. Bormane, “Marathi Language Speech Synthesis Using Concatenative Synthesis Strategy”, Second International Conference on Machine Vision, ICMV '09, pp. 181 – 185, 2009.

[58]. M. B. Chandak, R. V. Dharaskar, V. M. Thakre, “Text to Speech Synthesis with Prosody feature: Implementation of Emotion in Speech Output using Forward Parsing”, International journal of computer science and security, vol 4, issue 3,pp. 352-360, 2010.

[59]. Nishihara Tetsuo, Van De Weijer Jeroen, “On Syllable - Timed Rhythm and Stress-Timed Rhythm in World Englishes : Revisited”, International Conference on Phonology and Phonetics held in Shanghai International Studies University, May 2010.

[60]. Abu Naser, Devojyoti Aich, Md. Ruhul Amin, “Implementation of Subachan: Bengali Text To Speech Synthesis Software”, 6th International Conference on Electrical and Computer Engineering, ICECE 2010, Dhaka, Bangladesh, December 2010.

[61]. E.Veera Raghavendra, B. Yegnanarayana, Kishore Prahallad, “Speech Synthesis Using Approximate Matching of Syllables”, IEEE Workshop on Spoken Language Technologies, 2008.

[62]. Kalika Bali, “Interpreting Text for Indian Language TTS”,
[Online]: https://www.researchgate.net

[63]. J. Sangeetha, S. Jothilakshmi, “Robust Automatic Continuous Speech Segmentation for Indian Languages to Improve Speech to Speech Translation”,

International Journal of Computer Applications (0975 – 8887), Volume 53, No. 15, pp. 13-16, September 2012.

[64]. K. C. Rajeswari, P. Uma Maheswari, “Prosody Modeling Techniques for Text-to-Speech Synthesis Systems - A Survey”, International Journal of Computer Applications, Volume 39, No. 16, pp.8-11, February 2012.

[65]. Ruvan Weerasingle, Asanka Wasala, Kumuda Gamage, “A Rule Based Syllabification Algorithm for Sinhala”, IJCNLP, Springer, Verlag, Berlimpp. 438-449, 2005.

[66]. Parminder Singh, Gurpreet Singh Lehal, “Text To Speech Synthesis System for Punjabi Language”, In Proceedings of International Conference on Multidisciplinary Information Sciences and Technologies, Merida, Spain.

[67]. J. Sangeetha, S. Jothilakshmi, S. Sindhuja, V.Ramalingam, “Text To Speech Synthesis System For Tamil”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Special Issue, January 2013.

[68]. M. Siva Kumar, E. Prakash Prabhu, M. V. Subba Reddy, M. S. Praveen Kumar, “Text To Speech System for Telugu Language”, International Journal of Engineering Research and Applications, Vol. 4, Issue 3, Version 1, pp. 913-917, March 2014.

[69]. Madhavi R.Repe, “Natural Prosody Generation in TTS for Marathi Speech Signal”, IEEE International Conference on Signal Acquisition and Processing, 2010.

[70]. S. D. Shirbahadurkar, D. S. Bormane, R. L. Kazi, “Subjective and Spectrogram Analysis of Speech Synthesizer for Marathi TTS Using Concatenative Synthesis”, International Conference on Recent Trends in Information, Telecommunication and Computing, 2010.

[71]. Manjare Chandraprabha Anil, S. D. Shirbahadurkar, “Expressive Speech Synthesis using ProsodicModification for Marathi Language”, 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 2015.

[72]. Manojit Chaudhury, “Rule Based Grapheme to Phoneme Mapping for Hindi”, [Online]: http://citeseerx.ist.psu.edu.

[73]. Linguistics Data Consortium for Indian Languages, [Online]: http://www.ldcil.org/default.aspx /

[74]. Farig Yousuf Sadeque, Samin Yasar, Md. Monirul Islam, “Bangla Text to Speech Conversion: A Syllabic Unit Selection Approach”, International Conference on Informatics, Electronics and Vision (ICIEV), 2013.

[75]. V. M. Dhume, “Konkani ShudhlekanacheNhem - Goa Konkani Academy”, pp 6-11.

[76]. Bhuvana Narasimhan, Richard Sproat, George Kiraz, “Schwa-Deletion in Hindi Text-to-Speech Synthesis”, International Journal of Speech Technology, Kluwer Academic Publishers. Manufactured, Netherlands, pp. 319-333, 2004.

[77]. M. V Vinodh, Ashwin Bellur, K. Badri Narayan, Deepali M. Thakare, Anila Susan, N.M. Suthakar, Hema A. Murthy “Using Polysyllabic units for Text to Speech Synthesis in Indian languages” National Conference on Communications (NCC), 2010.

[78]. M. L. Dhore, S. K. Dixit, Ruchi M.Dhore, “Issues in Hindi to English and Marathi to English Machine Transliteration of Named Entities”, International Journal of Computer Applications, Vol 15, No 14, pp. 37-44, 2012.

[79]. S. Kayte, B. Gawali, “A Text-To-Speech Synthesis for Marathi Language using Festival and Festvox”, International Journal of Computer Applications (0975 – 8887), Volume 132, No.3, pp. 35-41, December 2015.

[80]. F. Y. Sadeque, S. Yasar, M. Islam, “Bangla Text to Speech Conversion: A Syllabic Unit Selection Approach”, International conference on Informatics, electronics and Vision (ICIEV), IEEE, Dhaka Bangladesh, 978-1-4799-0400-6, 2013.

[81]. S. N. Kayte, M. Mundada, C. Kayte, “Speech Synthesis System for Marathi Accent using FESTVOX”, International Journal of Computer Applications (0975 – 8887), Volume 130 – No.6, pp. 38-42, November 2015.

[82]. Back Track Linux, [Online]: http://www.backtrack-linux.org/downloads/

[83]. F. Alam, “Text To Speech Synthesis for Bangla language”, Thesis Submitted to the Department of Computer Science of BRAC University.

[84]. Devanagari Script, [Online]: https://en.wikipedia.org/wiki/Devanagari

[85]. Heiga Zen, Takashi Nose, Junichi Yamagish, Shinji Sako, “The HMM-based Speech SynthesisSystem (HTS) Version 2.0”, ISCA Workshop on Speech Synthesis, Bonn, Germany, August 22-24, pp. 294-299, 2007.

[86]. Junichi Yamagishi, “An Introduction to HMM-Based Speech Synthesis”, textbook, 2006.

[87]. Sangramsing Kayte, Monica Mundada, Jayesh Gujrathi, “Hidden Markov Model based Speech Synthesis: A Review”, International Journal of Computer Applications (0975 – 8887), Volume 130 – No.3, pp. 35-39, November 2015.

[88]. Heiga Zen, Tomoki Toda, “An Overview of Nitech HMM-based Speech Synthesis System for Blizzard Challenge 2005”, INTERSPEECH, Lisben, Portugal, pp.93-96, 2005.

[89]. Sankar Mukherjee, Shyamal Kumar Das Mandal, “A Bengali HMM Based Speech Synthesis System” Oriental COCOSDA., pp.225 259, 2012.

[90]. HMM/DNN-based Speech Synthesis System (HTS), [Online]:http://hts.sp.nitech.ac.jp/

[91]. Yu-Yun Chang, “Evaluation of TTS Systems in Intelligibility and Comprehension Tasks” ROCKLING’11 Proceedings of the 23rd Conference on Computational Linguistics and speech Processing, Taipei, China, pp. 64-78, September, 2011.

[92]. Salomon, Richard, “On the Origin of the Early Indian Scripts: A review article, Journal of the American Oriental Society (JSTOR), 115.2, pp 271-279, 1995