Linguistic challenges in Malayalam speech recognition: Analysis and solutions

Submitted in partial fulfilment of the

requirements of the degree of

DOCTOR OF PHILOSOPHY

Kavya Manohar

(D-TVE19JAN002)

Supervisor: Dr. A. R. Jayan

Co-Supervisor: Dr. Rajeev Rajan

Department of Electronics and Communication Engineering

College of Engineering Trivandrum

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Thiruvananthapuram

2023

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permissions have not been taken when needed.

Name: Kavya Manohar

Register No. D-TVE19JAN002

Date:30-10-2023

Place: Thiruvananthapuram

CERTIFICATE

It is certified that work contained in the thesis titled **"Linguistic challenges in Malayalam speech recognition: Analysis and solutions"** by "Kavya Manohar" has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Dr. A. R. Jayan
Professor
Dept. of Electronics and Communication Engineering
Govt. Engineering College, Thrissur

Dr. Rajeev Rajan
Associate Professor
Dept. of Electronics and Communication Engineering
Government Engineering College
Bartonhill, Thiruvananthapuram

Date: 30-10-2023

Place: Thiruvananthapuram

ACKNOWLEDGEMENTS

It is a great pleasure for me to express my respect and deep sense of gratitude to my supervisor **Dr. A. R. Jayan**, Professor, Department of Electronics & Communication Engineering, Government Engineering College, Thrissur, for his vision, guidance, enthusiastic involvement and persistent encouragement during the planning and development of this research work.

I make use of this opportunity to thank my co-supervisor **Dr. Rajeev Rajan**, Associate Professor, Department of Electronics & Communication Engineering, Government Engineering College, Bartonhill, Thiruvananthapuram for his expertise, wisdom, and involvement without which this work could not have been completed.

I would like to extend my heartfelt appreciation to the members of the Doctoral Committee for their valuable feedback and constructive suggestions, which have been helpful in shaping the direction of my research. I am forever thankful to my supervisors, for their trust and support, which provided me with the freedom to explore and experiment without constraints. Their guidance was instrumental in staying focused to my research goals.

My sincere gratitude goes to the members of the **Swathanthra Malayalam Computing**, whose efforts on creating accessible language technology tools has motivated me to ensure that my research and its outcomes are available to the public under permissive licenses.

I am immensely grateful to my fellow researchers, **Amlu**, **Lekshmi**, **Noumida**, and **Mala**, for their invaluable support and willingness to go the extra mile to help in times of need. I fondly acknowledge the encouragement, timely help and cooperation, I have received from the staff, faculty members and fellow researchers at CET, during various stages of this research.

I express my gratitude to **my parents and my sister**, whose enthusiastic interest in my work has been a source of motivation for me and I am honoured to have them as my family. I would like to express my heartfelt gratitude and love to **Santhosh**, whose presence in my life has been a spirited source of strength. I am beyond words to thank his dedication to our growth and well-being. I thank **Vani**, for joining us in the middle of this journey and filling our life with love, warmth, and joy.

Kavya Manohar

ABSTRACT

Speech is a simple and natural form of communication among humans. With the proliferation of devices and the widespread availability of the internet, it is essential to have machines that can interact with humans without any language barriers. However, devices and applications that can recognise, transcribe and respond to Malayalam speech is still far from a reality.

Successful development of natural language processing (NLP) solutions including automatic speech recognition (ASR) for morphologically complex low resource languages like Malayalam, requires many fundamental computational linguistic tools and techniques, in addition to vast amount of speech and text corpora. There is currently a severe lack of openly licensed speech corpora in Malayalam, with less than hundred hours of available data to train an ASR system. This is significantly less than the amount of data available for high resource languages with established ASR systems. However the large amount of web scraped Malayalam text corpora is easily available. This fact has motivated this research to stick on to the the classical speech recognition architecture of using separately trained acoustic and language models, rather that using modern End to End approaches that has orders of magnitude higher transcribed speech corpora requirement.

This research work addresses different linguistic domain challenges associated with converting Malayalam speech to textual form. Specifically, this research work explores the morphological complexity of Malayalam language using corpus linguistic parameters like type token ratio and moving average type token ratio, and concludes that Malayalam is more complex than other Indian and European languages analysed in the study. Based on this analysis, two potential directions for research in Malayalam automatic speech recognition (ASR) have been identified. The first involves developing a tool to create large vocabulary pronunciation lexicons, while the second involves implementing subword based ASR techniques. Both approaches have the potential to address the challenges posed by the complex morphology of Malayalam.

The classical architecture of ASR system rely on knowledge source like a pronunciation lexicon. A static pronunciation lexicon used to build an ASR decoder may not be sufficient to handle words the system may encounter in future. It will also be required to update the pronunciation lexicon, with new words from time to time. The morphological complexity of Malayalam leading to ever expanding word vocabulary pointed to the need for generating pronunciations of words automatically. The research in this direction, led to the development of a finite state transducer based software tool, Mlphon. Mlphon performs script grammar check, orthographic syllabification, phonetic feature analysis, grapheme to phoneme and phoneme to grapheme conversions. Mlphon is published as an

open source python library under MIT License. Using Mlphon, the largest open licensed pronunciation lexicon for Malayalam which contains 100,000 common words of different word categories is published.

Employing a large vocabulary pronunciation lexicon created using Mlphon, an ASR for Malayalam is developed. The acoustic modelling for this ASR is based on hybrid deep neural network and hidden Markov model (DNN-HMM) technique. The speech recognition system thus developed is evaluated on diverse test sets with low and medium out of vocabulary (OOV) words. The resulting ASR model is published under open license for enabling integration to various tasks.

To address the issue of OOV words in Malayalam ASR, this research work proposes the development of an open vocabulary speech recognition system using subword lexicons. The current research presents two linguistically motivated subword segmentation strategies, which are compared with the data driven strategies. The subword based approach has significantly reduced the ASR model size and improved the word error rate by recognising many out of vocabulary words that a word based ASR would typically miss. Overall, the proposed open vocabulary speech recognition system that utilises subword lexicons presents a promising solution to the problem of OOV words in Malayalam ASR, resulting in improved model performance and recognition of a wider range of words.

Contents

Declaration iii

Certificate v

Acknowledgement vii

Abstract ix

List of Figures xv

List of Tables xvii

List of Algorithms xix

List of Acronyms xxi

List of Symbols xxiii

1 Introduction 1

1.1 Background 1

1.2 Motivation, Challenges and Proposed Solutions 3

1.3 Research Objective 4

1.4 Contributions of this Thesis 4

1.5 Organisation of the Thesis 5

2 Review of Related Works 7

2.1 Introduction 7

2.2 ASR System Architectures 8

2.2.1 Pipeline Architecture 8

2.2.2 End to End (E2E) Architecture 17

2.3 Grapheme to Phoneme Conversion Systems 18

2.3.1 A Review of G2P Tools in Malayalam 19

2.3.2 Ready to Use Pronunciation Lexicon 21

2.4 Morphological Complexity Analysis 21

2.5 Subword Based Morphology Aware ASR Systems 22

2.6 Malayalam Speech Corpora 23

2.7 Malayalam Speech Recognition Systems . . . 25
2.8 Summary . . . 28

3 Quantitative Analysis of the Morphological Complexity of Malayalam 29
3.1 Introduction . . . 29
3.2 Morphological Complexity . . . 29
3.3 Dataset . . . 31
3.4 Experimental Details . . . 31
3.5 Result and Discussion . . . 32
3.6 Summary . . . 35

4 Finite State Transducer based Grapheme to Phoneme Conversion 37
4.1 Introduction . . . 37
4.2 Malayalam Grapheme to Phoneme Correspondence . . . 38
4.3 FSTs for Pronunciation Modelling . . . 38
4.4 Architectural Description . . . 40
4.4.1 Normalisation . . . 42
4.4.2 Word Boundary Tagging . . . 43
4.4.3 Syllable Boundary Tagging . . . 43
4.4.4 Preliminary Phonemisation . . . 43
4.4.5 Inherent Vowel Addition . . . 45
4.4.6 Alveolar Conjuncts Remapping . . . 45
4.4.7 Reph Sign Correction . . . 45
4.4.8 Schwa Addition (*Samvruthokaram*) . . . 46
4.4.9 Dental Nasal Disambiguation . . . 46
4.4.10 Labial Plosive Disambiguation . . . 47
4.4.11 Feature Tag Removal . . . 48
4.5 Syllabifier FST . . . 48
4.6 Phoneme Analyser FST . . . 48
4.7 G-P Converter FST . . . 49
4.8 The Python Library: Mlphon . . . 51
4.9 Intrinsic Evaluation of Mlphon . . . 52
4.9.1 Design of Gold Standard Lexicon . . . 52
4.9.2 Syllabification . . . 54
4.9.3 Grapheme to Phoneme Conversion . . . 56
4.10 Applications of Mlphon . . . 60
4.10.1 Phonemic Diversity Analysis of Speech Corpora . . . 60
4.10.2 Assisted Pronunciation Learning . . . 61
4.10.3 Text Sanity Check and Correction . . . 61
4.10.4 Creation of Large Vocabulary Pronunciation Lexicon . . . 63

4.11 Summary 63

5 Large Vocabulary Pronunciation Lexicons for Malayalam ASR 65
5.1 Introduction 65
5.2 Large Vocabulary Pronunciation Lexicons 65
5.3 Experimental Setup for Malayalam LVCSR 69
5.3.1 Dataset 69
5.3.2 Language Model 70
5.3.3 Pronunciation Lexicon 70
5.3.4 Acoustic Modelling 71
5.3.5 Decoding Lattice 74
5.4 Result Analysis 74
5.5 Publication of Malayalam LVCSR Model 75
5.6 Summary 76

6 Subword based Open Vocabulary Continuous Speech Recognition for Malayalam 79
6.1 Introduction 79
6.2 Open Vocabulary ASR 79
6.3 Graphemic and Phonemic Pronunciation Lexicon 81
6.4 Subword Segmentation Strategies - A Review 82
6.4.1 Word Segmentation 83
6.4.2 Morpheme Segmentation - Using Morfessor 83
6.4.3 BPE Segmentation 84
6.4.4 Unigram Segmentation 84
6.5 Proposed Subword Segmentation Algorithms for ASR 84
6.5.1 Syllable Segmentation 84
6.5.2 S-BPE Algorithm 85
6.6 Experimental Setup 87
6.6.1 Datasets 87
6.6.2 Hybrid ASR components 87
6.6.3 Creating Segmented Text Corpora 88
6.6.4 Language modelling 89
6.6.5 Creating Segmented Lexicons 91
6.6.6 Acoustic modelling 92
6.6.7 Summary of Experimental Investigations 92
6.7 Results 93
6.7.1 Corpus Linguistic Analysis 93
6.7.2 Language modelling Complexity 95
6.7.3 Evaluation of ASR performance 97

6.7.4 Comparison with Other Reported works . . . 100
6.8 Findings and Discussions . . . 100
6.8.1 Language modelling Complexity . . . 100
6.8.2 ASR results . . . 100
6.9 Summary . . . 101

7 Conclusion **103**
7.1 Major Contributions . . . 103
7.2 Limitations . . . 104
7.3 Future Scope . . . 104

Appendix I Characteristics of Malayalam Orthography **107**
I.1 Grapheme Inventory of Malayalam . . . 107
I.1.1 Vowels . . . 107
I.1.2 Consonants . . . 107
I.1.3 Signs . . . 109
I.1.4 Complex Graphemes in Malayalam . . . 109
I.2 Phoneme Inventory of Malayalam . . . 110
I.3 Grapheme to Phoneme Correspondence . . . 111
I.3.1 Vowels . . . 112
I.3.2 Base Consonants . . . 112
I.3.3 Consonant Clusters . . . 112
I.3.4 Multiple Functions of *virama* . . . 113
I.3.5 Syllable Final Consonants (Chillu, Anusvara, Visarga) . . . 113
I.4 The Syllable Structure of Malayalam . . . 114

Bibliography **117**

List of Publications **133**

List of Figures

1.1 Architecture of an ASR system 2

2.1 Architecture of a pipeline ASR decoder 8
2.2 Extracting Speech Features 9
2.3 GMM-HMM model of a phoneme, P 10
2.4 DNN-HMM model of a phoneme, P 12
2.5 TDNN based acoustic modelling with sub-sampling (red) and without sub-sampling (blue+red). 13
2.6 Finite state acceptor graph formed by the sequence of M feature vectors extracted from speech to be decoded 16

3.1 TTGR and TTR plot of Malayalam for SMC Corpus of Wikipedia text . . 33
3.2 Comparison of Malayalam TTR with that of European Union Constitution Corpus and DoE-CIIL Corpus 33
3.3 TTR plotted at different segments of the SMC corpus for 1000 window positions 34
3.4 Comparison of MATTR values computed for Malayalam on SMC Corpus with that of European Union Constitution Corpus 34

4.1 An FST representing a simple pronunciation mapping that accepts two words അവൻ and അവൾ. 39
4.2 The system architecture of Mlphon 41
4.3 Two alternate representations of ൻ at input is being mapped to the normalised form at the output. 42
4.4 Normalisation of the word അവൻ, indicating the two possible input sequences generating the normalised output. 43
4.5 Insertion of syllable tags, <BoS> <EoS> are indicated by transitions in green colour. All other symbols are mapped to themselves. Word boundary tags inserted in previous FST is also shown. 43
4.6 Syllabifier performing syllabification on the word അവൾ. Boundary tags for words and syllables are demonstrated. 48
4.7 Phoneme analyser performing analysis on the word അവൾ. It returns the sequence of phonemes in its pronunciation along with articulatory feature tags in angle brackets. 49

4.8 Phoneme analyser FST, showcasing grapheme to phoneme conversion on the word അവൾ. 49
4.9 G-P Converter FST, performing phoneme analysis on the word അവൾ. . 50
4.10 G2P converter FST performing (i) grapheme to phoneme conversion on the words അവൾ and അവൻ in analysis mode and (ii) phoneme to grapheme conversion on avaɭ and avan in generate mode. 50
4.11 The lexical entries of gold standard lexicon is chosen from the most frequent thousand words from the IndicNLP corpus [124] such that these words cover 26% of the 167.4 million tokens present in this corpus. . . . 53
4.12 Distribution of word types in gold standard lexicon 53
4.13 Phoneme diversity analysis in gold standard lexicon 54
4.14 Distribution of syllabification errors in different word types in gold standard lexicon 56
4.15 Confusion matrix comparing Mlphon transcription with gold standard transcription. The values are normalised and represented as percentage. . . . 58
4.16 In an evaluation space of 100k tokens, we computed the accuracy of transcribing ഫ. The two possible pronunciations /f/ and /pʰ/ were accurately identified in more than 99% of the cases as shown in this confusion matrix. 59
4.17 Distribution of G2P errors in different word types in gold standard lexicon. 60
4.18 Phonemic diversity analysis of various speech corpora used in ASR Experiments. It indicates relative frequency of each phoneme. 61
4.19 Web interface for Mlphon. Features of syllabification, phonetic analysis and IPA transcription are shown. 62

5.1 The Website demonstrating the ASR developed as per the description in this work 76

6.1 Block schematic representation of hybrid ASR system, with subword based language model and pronunciation lexicon 79
6.2 Distribution of the number of subword segments per word 94
6.3 Segment Length distribution in the datasets 95
6.4 ASR performance evaluation in terms of the WER (lower is better) 97
6.5 Trade-off between ASR performance and Model Memory Requirement (n-gram order is shown as labels) 98

I.1 Structure of a syllable. C-consonant, V-vowel, ?- indicates optionality . . 114

List of Tables

2.1 A phonemic and graphemic pronunciation lexicon 14
2.2 Comparing the functionalities and features of G2P conversion tools in Malayalam 19
2.3 Details of Speech data sets available under open license so that the derived models can have unrestricted usage. 24

3.1 Complex morphological word formation in Malayalam 30

4.1 Parameters of FST illustrated in Fig. 4.1 is defined in this table. 40
4.2 Comparing the syllabification provided by Mlphon with gold standard reference. 55
4.3 Comparing the G2P transcription provided by Mlphon with gold standard reference. 57
4.4 Precision, Recall and F1 Scores of phoneme transcription by Mlphon. For all other phonemes, these metrics are evaluated to be 100%. 57
4.5 Number of word tokens flagged as invalid by Mlphon on different transcribed speech corpus and corresponding error rates. 62

5.1 Pronunciation lexicons of different word categories. 66
5.2 An excerpt from pronunciation lexicons. 66
5.3 A qualitative linguistic comparison between the lexicons produced by Mlphon and freely accessible automated tools 68
5.4 Comparison of the word processing speed of the proposed tool Mlphon with other openly available tools. 69
5.5 Details of Speech data sets used in our experiments. 70
5.6 Comparison of pronunciation lexicons created by different automated tools 70
5.7 Comparing WER (%) obtained in Malayalam ASR experiments with lexicons created using the proposed tool, Mlphon, and other openly available tools. 75
5.8 Comparison of WER from previously reported works on Malayalam ASR. The ASR we built using the lexicon created with Mlphon performs at par with the previously reported works. 76

6.1 Segmentation illustrating the usage of continuity marker symbol '+'. . . . 81
6.2 Phonemic Lexicon 81

6.3 Graphemic Lexicon 81
6.4 Special consonants and Virama sign in Malayalam 85
6.5 Details of Speech data sets used in our experiments. 87
6.6 Examples for different segmentation strategies. Space is used as delimiter between segments. 89
6.7 Lexicon sizes of different segmentation 92
6.8 Sentence length statistics in terms of the number of segments per sentence. 94
6.9 Language modelling Complexity in terms of Perplexity 96
6.10 Language Modelling Complexity in terms of SPS 96
6.11 WER on OOV words 99

I.1 Short and long vowels in Malayalam and their IPA representations 108
I.2 Consonants in Malayalam and their IPA representations 108
I.3 *Chillus* in Malayalam and their IPA representations along with the base consonants from which *chillus* were derived. 109
I.4 Signs in Malayalam 109
I.5 Examples of consonant clusters in Malayalam and their constituents . . . 110
I.6 Vowel phonemes of Malayalam 110
I.7 Consonant Phonemes of Malayalam 111

List of Algorithms

1 Computation of MATTR 32
2 Normalisation, Word Boundary Tagging, Syllable Boundary Tagging . . . 44
3 Preliminary Phonemisation 45
4 Inherent Vowel Addition, Alveolar Conjuncts Remapping and Reph Sign Correction 46
5 Schwa Addition, Dental Nasal Disambiguation, Labial Plosive Disambiguation 47
6 FST based Syllabification Algorithm 85
7 S-BPE Training Algorithm 86
8 Subword Lexicon from Word Lexicon 91

List of Acronyms

AED	attention based encoder decoder
AM	acoustic model
ANN	artificial neural network
API	application program interface
ASR	automatic speech recognition
BPE	byte pair encoding
CD	context dependent
CD GMM-HMM	context dependent GMM-HMM
CI GMM-HMM	context independent GMM-HMM
CMVN	cepstral mean and variance normalisation
CNN	convolutional neural network
CTC	connectionist temporal classification
DNN	deep neural network
E2E	End to End
EM	expectation maximisation
fMLLR	feature space maximum likelihood linear regression
FST	finite state transducer
G2P	grapheme to phoneme
GMM	Gaussian mixture model
HCI	human computer interaction
HMM	hidden Markov model
ICFOSS	International Centre for Free and Open Source Software
IITM	Indian Institute of Technology Madras
IMaSC	ICFOSS Malayalam speech corpus
LDA	linear discriminant analysis
LF-MMI	lattice free maximum mutual information
LFBE	log filter-bank energy
LM	language model
LPC	linear prediction coefficient
LVCSR	large vocabulary continuous speech recognition
MATTR	moving average type token ratio
MFCC	mel frequency cepstral coefficient
MLLT	maximum likelihood linear transformation

MLP	multi layer perceptron
MMI	maximum mutual information
MSC	Malayalam speech corpus
MSL	mean segment length
NLP	natural language processing
OOV	out of vocabulary
Open SLR	Open speech and language resources
P2G	phoneme to grapheme
PASM	pronunciation assisted subword modeling
PER	phoneme error rate
PL	pronunciation lexicon
PLP	perceptual linear prediction
RNA	recurrent neural alignment
RNN	recurrent neural network
S-BPE	Syllable-BPE
SAT	speaker adaptive training
SFST	Stuttgart finite state transducer
SPS	surprisal per sentence
SVM	support vector machine
TDNN	time delay neural network
TTGR	type token growth rate
TTR	type token ratio
TTS	text to speech
WER	word error rate
WFST	weighted finite state transducer
WPS	word processing speed

List of Symbols

$Pr(.)$	Probability function
X	Speech feature vector sequence
X_k	k^{th} speech feature vector
P	An independent phoneme
W	Words in a sentence
$\widehat{W}$	Words in a sentence predicted by the ASR decoder
Q	Finite set of states in an HMM
Q_i	i^{th} state in an HMM
a_{ij}	Transition probability in an HMM
$b_i(X_k)$	The probability of an HMM state Q_i generating an observation X_k
$\widehat{Q}$	HMM state predicted by the acoustic model
w_i	i^{th} word or subword in a sentence
P_{CD}	A context dependent phoneme
V	Type count in a corpus
N	Token count in a corpus or a sentence
L	The length of text window
q	Finite set of states in an FST
qi	i^{th} state in an FST
I	Finite set of initial states in an FST
F	Finite set of final states in an FST
Σ	Finite set of output symbols of an FST
Γ	Finite set of input symbols of an FST
δ	The state transition function in an FST
σ	The output function in an FST
<BoW>	A tag to indicate beginning of word
<EoW>	A tag to indicate end of word
<BoS>	A tag to indicate beginning of syllable
<EoS>	A tag to indicate end of syllable
NLL	Negative Log Likelihood
PPL	Perplexity
$PPL_c(W)$	Perplexity of a sentence W, normalised to number of characters
$PPL_w(W)$	Perplexity of a sentence W, normalised to number of words
OOV-WER	word error rate of out of vocabulary words

Chapter 1

Introduction

Speech is a natural form of communication among humans. Speech based human computer interaction (HCI) is often preferred in personal digital assistants and home automation tools. ASR is the process of converting spoken utterances into textual form as a sequence of words. ASR systems have applications in generating captions for videos, online meetings, transcribing lectures, speech based text input systems, interactive voice response systems etc.

Speech recognition is a challenging problem that demands for addressing both the acoustic and linguistic aspects of spoken language. This includes understanding the physical properties of speech signals, as well as the linguistic structure and meaning of the words and phrases being spoken. However the solutions to linguistic challenges that are effective for one language cannot be easily transferred to other languages. This is because natural human languages are so diverse in terms of their typology, phonetic inventory, word vocabulary and in digital resources. This thesis aims to address specific linguistic domain challenges in the development of large vocabulary continuous speech recognition (LVCSR) for Malayalam language. This research work presents a comprehensive analysis of the challenges, in terms of morphology and phonology, faced in building an ASR system for Malayalam and proposes various tools and techniques to overcome them. Furthermore, these proposed solutions have applications beyond ASR and can address some of the generic problems in Malayalam natural language processing (NLP).

1.1 Background

Digital devices and hardware are getting cheaper and easily available. Utilisation of these devices to its full potential is possible only if every user is able to communicate with these devices without barriers. A functional ASR in one's native language is essential for speech based typing, accessing speech enabled digital assistants and using interactive voice response systems. This can be extended to applications like transcribing speech for the hearing challenged and futuristic applications like speech to speech translation.

The experiments used in this research work utilises the classical architecture of speech recognition described in Fig. 1.1. Over the last three decades, the field of ASR has undergone significant advancements, with researchers focusing on optimising each component

of the ASR architecture. Toolkits such as HTK [1], CMU Sphinx [2], and Kaldi [3] have been developed based on this architecture, enabling researchers to integrate their work on improving individual components and evaluate their impact on the overall performance of the ASR system.

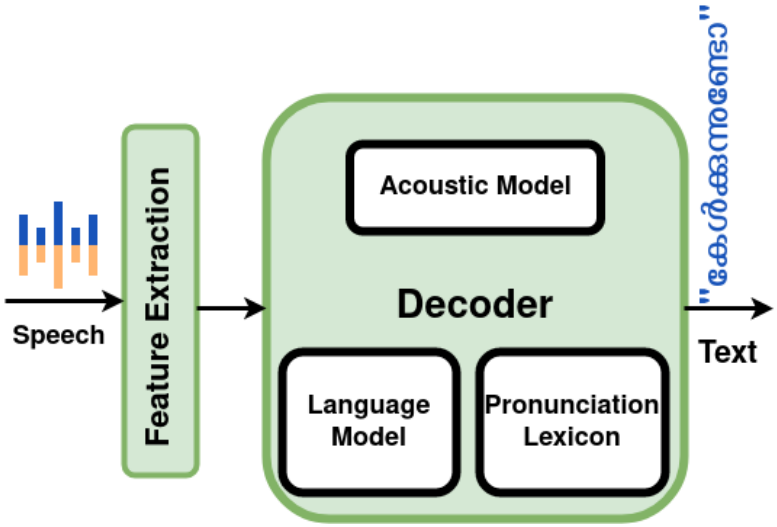


Figure 1.1: Architecture of an ASR system

In the context of this research on the morphologically complex Malayalam language, a significant challenge was encountered, stemming from the limited availability of annotated speech corpora under open licenses, thus positioning it as a low-resource language. Classical or pipeline architecture is best suited than the alternate End to End (E2E) architecture, in scenarios where text data outweighs audio data, a characteristic of the Malayalam language [4]. The research outcomes presented in [5] and [6] indicate that attempting to train an E2E ASR system from scratch, using a dataset comprising of less than 100 hours of annotated speech data, would lead to unsatisfactory accuracy. Additionally, the classical ASR models offer the advantage of easy integration into small hardware devices, enabling fast on-device speech recognition [4]. Thus, the research makes the deliberate choice to employ classical architecture over the E2E approach, driven by the unique challenges and characteristics associated with the Malayalam language.

In the pipeline ASR system, the acoustic model (AM) is the learnt representation of relationship between the acoustic features of speech and the phonemes of the language. The AM uses the Gaussian mixture model (GMM) to model the acoustic features of speech and the hidden Markov model (HMM) to model the temporal structure of speech. The GMM-HMM AM is trained on annotated speech data, and its parameters are learned from the data using statistical techniques. The deep neural network (DNN) based AM is an up-gradation over the conventional GMM-HMM approach of acoustic modelling and is generally called DNN-HMM approach. The language model (LM) is a model of the probability distribution of words or subwords in the language. The LM is typically implemented

as a statistical n-gram, which predicts the likelihood of a sequence of words or subwords given the previous words in the sentence. The pronunciation lexicon (PL) is a list of words or subwords in the language, along with the corresponding phoneme level transcriptions. A detailed description of ASR architectures is provided in Chapter 2.

1.2 Motivation, Challenges and Proposed Solutions

Malayalam is a Dravidian language spoken in the Indian state of Kerala and the union territories of Lakshadweep and Puducherry. It is one of the twenty two scheduled languages of India spoken by nearly 2.88% of Indians, with more than thirty five million speakers across the globe. Malayalam has the official language status in the state of Kerala and in the union territories of Lakshadweep and Puducherry (Mahé). Malayalam has been designated as a classical language in India since 2013[1] due to its rich literary heritage and cultural significance.

However, despite a huge number of speakers, based on our investigations and survey of reported literature, it is understood that Malayalam is a low-resourced language for the following reasons [7]:

1. Limited availability of linguistic resources, like
   (a) Lexical resources. eg: Machine-readable dictionaries.
   (b) Linguistic corpora. eg: Transcribed speech corpus [8] and parallel text corpus.
   (c) Linguistically annotated corpora. eg: part-of-speech tagged text corpus.
2. Lack of tools for creating linguistic resources manually or semi-automatically [2].
3. Lack of benchmark datasets for comparison of results.

The field of speech recognition for the Malayalam language has not yet reached maturity and is not ready for real-world applications due to a variety of reasons, including a shortage of annotated speech data and a lack of computational linguistic resources that can address the unique morphological and phonological features of the Malayalam language.

The morphological complexity in the word formation rules in Malayalam practically makes its vocabulary unlimited [9]. Each root word can give rise to hundreds of derived words by agglutination, inflection and compounding [10]. Additionally, loan words from other languages and proper nouns get added to the vocabulary very fast in modern days. The grapheme to phoneme correspondence in Malayalam is not truly one-to-one (See Appendix I for details). A static PL used to build an ASR decoder may not be sufficient

[1] https://en.wikipedia.org/wiki/Malayalam
[2] https://en.wikipedia.org/wiki/Language_resource

to handle words the system may encounter in the future. It will also be required to update the PL, with new words from time to time. All of this leads to the necessity for an automated grapheme to phoneme (G2P) conversion tool and this research work proposes and implements its development.

The challenges of the morphological complexity of the language can be addressed by open vocabulary speech recognition. This refers to the use of subword based language models and lexicons instead of word based ones [11]. This method effectively handles each word by decomposing it to subwords and then reconstructing them back to words after decoding. This research work proposes and implements linguistically informed subword modelling algorithms and compare their effectiveness in Malayalam ASR task.

1.3 Research Objective

The focus of this research is on the the investigation of the linguistic challenges associated with ASR in Malayalam language and the development of open source computational linguistic tools and techniques essential to solve them. Specifically the objectives can be described as:

1. To analyse the morphological complexity of Malayalam and the challenges it imposes on building a continuous speech recognition system.
2. To automate the process of creating a large vocabulary pronunciation lexicon taking care of precise grapheme to phoneme conversion rules of Malayalam.
3. To build a LVCSR model for Malayalam using the pipeline architecture, combining a large vocabulary pronunciation lexicon, a hybrid DNN-HMM acoustic model and a statistical language model.
4. To build an open vocabulary ASR system for Malayalam using subword based PL and LM to reduce the impact of morphological complexity.

1.4 Contributions of this Thesis

Considering the challenges involved in an ASR system for Malayalam, this research has contributed towards the development of an open and functional ASR model for Malayalam that could be integrated to various tasks. The major contributions are:

1. Quantitative analysis of the morphological complexity of Malayalam language.
2. Documentation of the graphemic and phonemic inventory of Malayalam and the correspondence between the two.

3. Development of an algorithmic description of the grapheme-phoneme correspondence in Malayalam and its implementation into a modular toolkit which may find applications in script grammar check, orthographic syllabification, phonetic feature analysis, grapheme to phoneme and phoneme to grapheme conversions.
4. Publication of large vocabulary pronunciation lexicon of more than hundred thousand words belonging to different word categories.
5. Development of a LVCSR model for Malayalam and publication of an open licensed Malayalam ASR model that could be integrated to various applications.
6. Exploration of subword segmentation strategies suited for Malayalam considering its morphological complexity.
7. Development of subword based open vocabulary ASR model to reduce word error rate (WER) and model memory requirement.

Furthermore, open dissemination of the source codes and the developed models is very much essential to ensure reproducibility, reusability, and most importantly for validation and further improvements, leading to research continuity. This aspect has been given utmost priority at every stage of this research work.

1.5 Organisation of the Thesis

The chapter 1 of this thesis highlights the motivation, specific challenges, research objectives and the contributions of this work. Chapter 2 reviews the related works on general ASR system architectures, G2P conversion systems, morphology aware ASR systems, Malayalam speech corpora and Malayalam ASR systems.

Chapter 3 describes the quantitative analysis on the morphological complexity of Malayalam language. This analysis suggested two potential directions for research. The first involves developing a tool to create large vocabulary pronunciation lexicons, while the second involves implementing subword based ASR techniques.

Chapter 4 explains the design and development of bidirectional G2P toolkit Mlphon. The chapter also presents the evaluation of the toolkit intrinsically on a gold standard lexicon and the NLP applications of this toolkit. Chapter 5 presents the development of an LVCSR system for Malayalam with a large vocabulary pronunciation lexicon created using Mlphon, acoustic and language models trained using various openly licensed speech and text datasets. It also presents a comparison of the ASR results with other lexicon creation tools.

Chapter 6 explores different subword segmentation strategies for ASR in Malayalam language. Chapter 7 concludes the thesis highlighting the contributions, limitations and list-

ing the scope for future research work.

A detailed documentation of the graphemic and phonemic inventory of Malayalam is provided in Appendix I. It is followed by the list of references and the list of publications based on the findings in this thesis.

Chapter 2

Review of Related Works

2.1 Introduction

This chapter describes various aspects in the development of an ASR system for Malayalam language. It starts with a comprehensive review of different speech recognition architectures in Section 2.2, which helps to make an informed choice of the right architecture for building the Malayalam ASR system.

One of the critical components of building an ASR system is the G2P conversion system. In Section 2.3, we discuss the need for G2P conversion systems and the various approaches we can take to build them. G2P conversion systems are crucial in transforming the words into sequence of fundamental speech units (phonemes), which is essential in developing an ASR system. Additionally, we also explore the creation of ready to use pronunciation lexicon (PL) for different languages in Section 2.3.2. A PL is a dictionary that contains a mapping of words to their corresponding phonemes. These pre-built lexicons can aid in building an ASR system more efficiently and accurately. We then move on to discuss how the morphological complexity of languages is quantitatively analysed in Section 2.4. As we explore the Malayalam language, we find that it has a high degree of morphological complexity, which presents some unique challenges for developing an ASR system. We explain how we can overcome these challenges and the approaches we can take to tackle them.

As we move on to discuss the development of ASR systems in languages with high morphological complexity in Section 2.5, we explore some of the challenges faced in developing these systems. We look at some of the approaches and techniques that researchers have used to address these challenges and to build effective ASR systems for such languages. Then in section 2.6, we document various openly available speech and text corpora available in Malayalam for developing speech and language technology applications. Finally, we review previous reported works on the development of ASR systems for the Malayalam language in Section 2.7. We examine the different approaches taken by researchers and the outcomes of their efforts in building ASR systems for Malayalam. This review will help to understand the progress made in this field and the areas that require further attention in developing an effective ASR system for Malayalam.

2.2 ASR System Architectures

The development of ASR architectures has undergone significant changes since its inception in the mid-twentieth century. One of the early attempt of ASR was the basic word template matching method, which was used for isolated digit recognition in English in 1952 [12]. The Harpy system [13] introduced in 1972, used phoneme level template matching and graph based search for recognising spoken sentences with a limited vocabulary. However, a major breakthrough in ASR research occurred in the 1990s with the release of open source HMM toolkits such as HTK [1] CMU Sphinx [2] and Kaldi [3].

These toolkits effectively combined the concepts of HMM, GMM, and statistical language models, which had been introduced in the 1970s and 1980s [14]. This led to the development of more accurate and robust ASR systems capable of recognising larger vocabularies and handling variations in speech due to factors such as accent and speaking rate. Recent advancements in deep learning techniques such as Transformer models have further improved the performance of ASR systems, leading to their widespread use in various applications such as virtual assistants, transcription services, and language translation.

In the following sections we describe in detail the pipeline architecture of ASR and an overview of E2E ASR architecture.

2.2.1 Pipeline Architecture

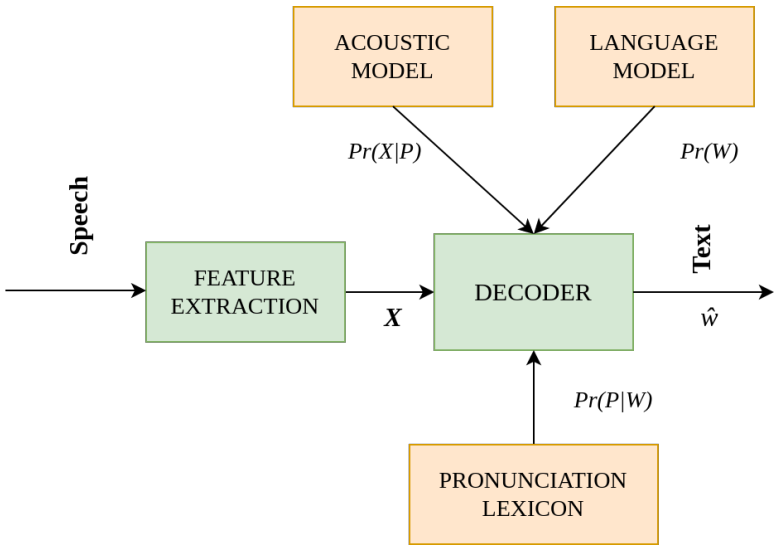


Figure 2.1: Architecture of a pipeline ASR decoder

We present the most widely used ASR architecture here. A pipeline ASR is built on different components, each one developed independently during the training stage. The components are: an AM typically implemented using a statistical approach or using a neural network, a static PL, and a LM usually represented as a probabilistic component. The

decoding process is carried out using a weighted finite state transducer (WFST) which is created by combining these elements through composition operations [15]. This pipeline structure shown in Fig. 2.1 [4] is now considered the classical architecture with the advent of modern E2E ASR architectures. In this figure, $Pr(.)$ indicates the probability function, X indicates the speech feature vector, P indicates the phonemes, W indicates the words in a sentence and $\widehat{W}$ indicates the words predicted by the ASR decoder. The steps in pipeline architecture of ASR are described in the following subsections.

Feature Extraction

The waveform of raw speech signal in a speech corpus is simply a representation of air pressure variation over time. In order to extract information that is relevant for acoustic modelling, the raw speech signal is transformed into alternate representations. This process involves splitting the audio into frames of fixed size using overlapping windows. The window shapes, like Hamming or Hanning, are designed to smoothen frame border discontinuities in order to avoid the occurrence of frequency artifacts [16]. Typically the window size is 25ms with 10ms overlap with previous frame. The series of frequency domain transformations on each frame returns a low dimensional feature vector [14].

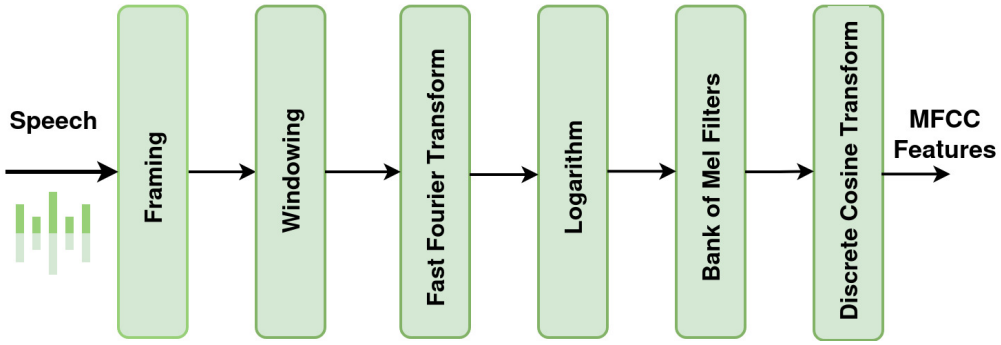


Figure 2.2: Extracting Speech Features

Mel frequency cepstral coefficient (MFCC) is one of the most popular acoustic feature used in speech recognition systems. The extraction of this feature is described in Fig. 2.2 [4]. Alternately, there are features like log filter-bank energy (LFBE), or perceptual linear prediction (PLP) cepstral coefficients that provide good acoustic representations of speech units (or phones) while suppressing variations in the signal due to factors such as the speaker, channel, speaking style, and recording environment [17]. In all our experiments, we have used MFCC features unless stated otherwise. In addition to the MFCCs, first-order (delta) and second-order (delta–delta) regression coefficients are often appended in a heuristic attempt to compensate for the conditional independence assumption made by the HMM-based acoustic models [18]. For DNN based acoustic modelling we have additionally used i-vectors as feature vectors [19].

Acoustic Model (GMM-HMM)

A phoneme is the fundamental unit of speech in a language. To create an AM, each phoneme in a language is given an identifiable label. The AM represents the relation between the phoneme labels and the acoustic features [17]. The acoustic manifestation of a phoneme is dependent on the speaker, rate of speech delivery, environmental factors and additionally on the phoneme context in which it occurs. To model the temporal variations of a phoneme we use HMM with multiple states. It allows for fine grained modelling of a single phoneme into its component parts.

When a single independent phoneme is modelled using an HMM, we call that the monophone acoustic modelling scheme. However the acoustic realisation of the phoneme depends largely on the context of its occurrence. A vowel occurring in between two plosives behave differently to that of a vowel that occurs at the beginning of an utterance. When a phoneme in different contexts is modelled using different HMMs the modelling effectiveness would be better. This type of an acoustic model which takes into account the left and right context of a phoneme is called the triphone acoustic model. In triphone acoustic model, the speech units are labelled as triphones. The HMM states of triphones having similar acoustic identities are given same labels and are called tied states. The similar sounding triphone states that are tied together are determined by phonetic decision trees [20].

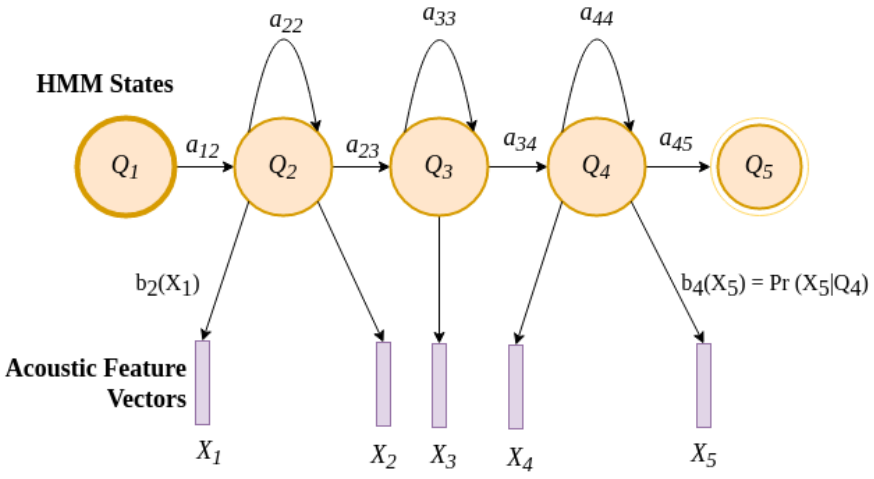


Figure 2.3: GMM-HMM model of a phoneme, P

In general an HMM is defined by:

1. a set of states, $Q = \{Q_0, Q_1, ...Q_N\}$
2. a set of transitions between states
3. transition probability, a_{ij}: the probability of traversing from state Q_i to Q_j

4. a set of possible observations which are acoustic (speech) feature vectors, X_k
5. observation probability, $b_i(X_k)$: probability of state Q_i generating possible observation X_k.

While modelling a phoneme using HMM, the acoustic features, X_k are the observations. The set of hidden states are referred to as senones. Each phoneme is composed of multiple senone states indicated by Q_i. Each state in an HMM is defined by an observation probability density function specified by multivariate Gaussians or GMMs indicated by $\mathcal{N}(\mu, \sigma^2)$, where μ and σ^2 indicate the mean and variance of the observations [15]. These Gaussians represent the probability of audio feature vectors, X_k given the phoneme state label Q_i, ie., $Pr(X_k|Q_i)$. The transition between states are strictly left to right with the addition of self loops. This ensures a phoneme can always be represented by the same GMM-HMM model, whether it is uttered fast or slow. A slowly uttered phoneme will pass through more self loops. This structure of GMM-HMM phoneme model is described in Fig. 2.3 [14,18]. The purpose of GMM-HMM acoustic model training is to estimate the parameters of this model, corresponding to every phoneme. These parameters are learnt from a huge collection of speech, spoken by multiple speakers.

Acoustic Model (DNN-HMM)

Instead of using GMMs to model the observation probability density function of the HMM states, DNNs [21] can as well be used. This configuration as shown in Fig. 2.4 is referred to as the hybrid DNN-HMM acoustic model [14, 22]. The DNNs have the audio frame features at the input and HMM state labels at the output. The DNN model training procedure relies on the GMM-HMM training to produce labelled data. The advantage of using DNNs instead of GMMs is that they are better equipped to learn complex non-linear functions [17].

Even though the labels come directly from the GMM model, for the DNN, each training data instance (audio feature vector) will contain additional contextual information about the left and right frames [14,22]. Unlike GMM-HMM models which predict the likelihood of audio feature vector given a context dependent phoneme state, hybrid DNN-HMM models predict the posterior probability of a context dependent tied phoneme state given an audio, $Pr(Q|X)$. However both defines a relation between audio and phonemes.

The GMM-HMM training is called generative training while the DNN-HMM is called discriminative training. In discriminative training, models are trained to to maximise the separation between the correct and incorrect labels, or to discriminate between correct and incorrect labels rather than simply assign high weights to the correct sequence of labels[1]. The discriminative training objective is usually based on reducing the frame level cross-

[1] `https://m-wiesner.github.io/LF-MMI/`

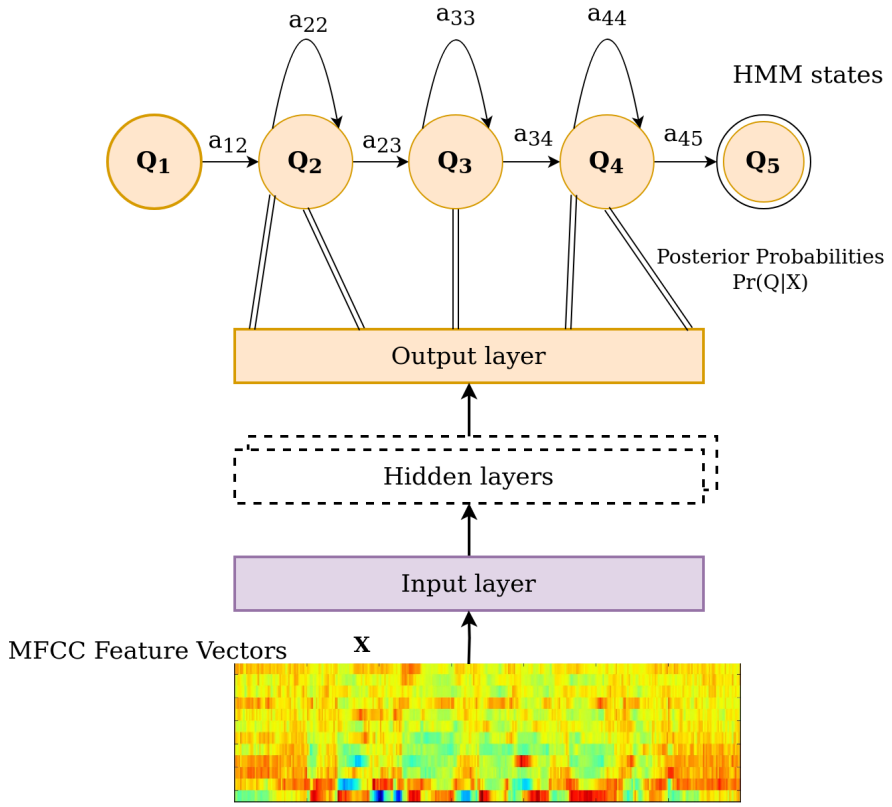


Figure 2.4: DNN-HMM model of a phoneme, P

entropy loss as in equation 2.2.1, where Q and $\widehat{Q}$ represents the true and predicted HMM state labels.

$$L_{CE}(Q, \widehat{Q}) = \sum_{i=1} Q_i \log(\frac{1}{\widehat{Q}_i}) \tag{2.2.1}$$

An improvement over the basic feed forward DNN to use more temporal context would be to use time delay neural network (TDNN) as demonstrated in Fig. 2.5 [23]. Each layer in TDNN acts at different temporal resolution, where higher layer are designed to have a wider receptive field. Since there would be overlap between input contexts at adjacent time steps, the connections in the network are sub-sampled as shown by solid lines in Fig. 2.5. It reduces redundancy and improves computational efficiency during training.

A factored form of TDNNs (TDNN-F) which is structurally the same as a TDNN whose layers have been compressed via singular value decomposition, but is trained from a random start with one of the two factors of each matrix constrained to be semi-orthogonal has proven to give substantial improvements over TDNNs [24].

Hybrid DNN-HMM architectures using frame level cross entropy loss as a discriminative training criteria can be aided by maximum mutual information (MMI) objective. The MMI training objective tries to predict the correct sequence of words corresponding to

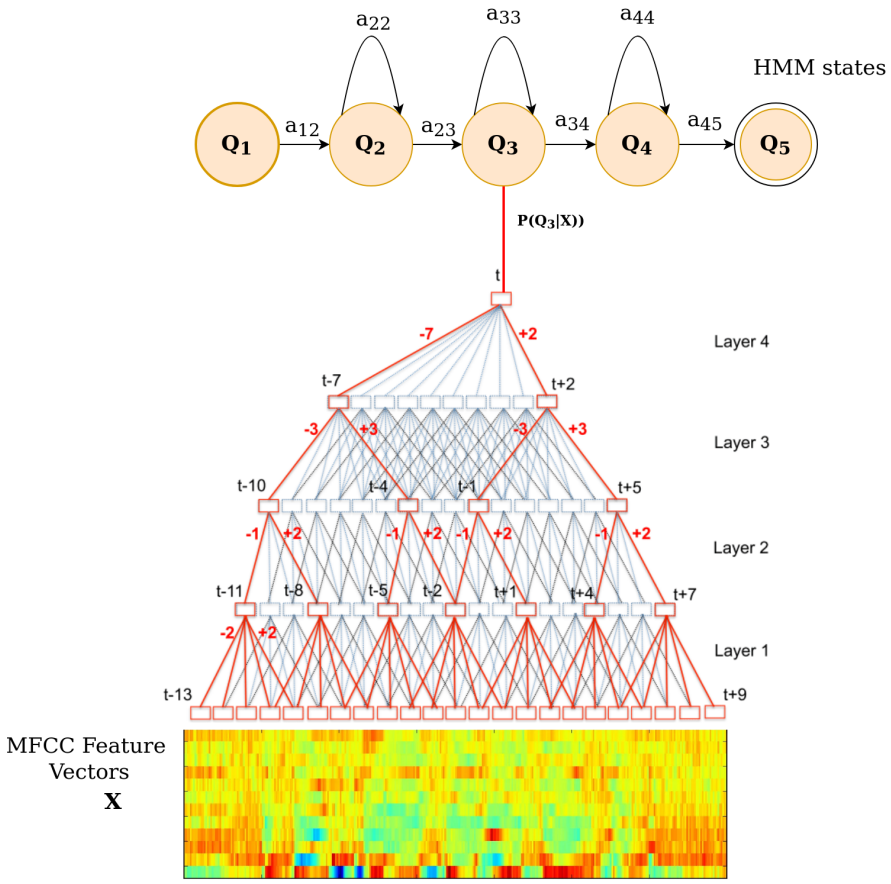


Figure 2.5: TDNN based acoustic modelling with sub-sampling (red) and without sub-sampling (blue+red).

an utterance as a whole [25]. The lattice free maximum mutual information (LF-MMI) algorithm proposed in [26], has enabled the usage of GPUs to implement the MMI criteria and has reported improved ASR results. The Kaldi implementation of TDNN-F has been the choice of DNN-HMM acoustic model in the current research, due to its proven acoustic modelling efficiency in the context of limited training data.

Pronunciation Lexicon

A pronunciation lexicon (PL) is an important component in speech recognition and synthesis systems. It serves as a mapping between written words and their corresponding pronunciations, usually represented as a sequence of phonemes. In a classical ASR system, the acoustic model undergoes training using annotated speech corpora, where the annotations consist of word-level transcripts of the spoken utterances. However, to align the utterances with phoneme-level transcriptions, the training process relies on the PL. This lexicon additionally holds a crucial role within the ASR decoder, working in tandem with the acoustic model and the language model to ensure precise transcription of the spoken content [18].

In the case of Malayalam, a PL is used to map every word or subword to its pronunciation using a one-to-one mapping, with some exceptions that are handled using a bidirectional G2P toolkit, Mlphon, developed as part of this research work and explained in Chapter 4. The G2P toolkit is an automated system that can automatically generate pronunciations. In some cases, a word may have multiple valid pronunciations, and these alternate pronunciations are also included in the PL. This can happen for a variety of reasons, such as regional variations in pronunciation or different meanings associated with different pronunciations.

Table 2.1: A phonemic and graphemic pronunciation lexicon

Word	Phonemic Lexicon	Graphemic Lexicon
ഈ	iː	ഇ ◌ൗ
ഉന്നത	u n n a t̪ a	ഉ ന ◌്ന ത
എന്ന	e n̪ n̪ a	എ ന ◌്ന
ഒരു	o ɾ u	ഒ ര ◌ു
കഫേ	k a f eː	ക ഫ ◌േ
ഫോട്ടോ	f oː ʈ ʈ oː	ഫ ◌ോ ട ◌്ട ◌ോ

Finally, when generating the PL for subwords (i.e., smaller units of language that are not complete words), sometimes a graphemic PL is used. This means that the pronunciation is based solely on the spelling of the subword and not on any specific established pronunciation. This approach can be useful for handling subwords that have ambiguous pronunciations. Example entries from a phonemic and graphemic PL is shown in Table 2.1. The probability of a phoneme sequence given a word segment can be estimated from this lexicon and this probability $Pr(P|W)$ is referred to as the lexical model where P indicates the phoneme label.

Language Model

A language model (LM), predicts the most probable word (or subword) sequence, given the list of previously occurred words or subwords. For a sentence W formed by sequence of N word (or subword) segments $W = w_1, w_2\ w_i\ ...w_N$, the probability $Pr(W)$ of the sentence is given by the following formula applying the chain rule of probability.

$$Pr(W) = Pr(w_1, w_2, ...w_N) \tag{2.2.2}$$

$$= Pr(w_1)Pr(w_2|w_1)...Pr(w_N|w_{N-1}, w_{N-2}...w_1) \tag{2.2.3}$$

Based on the Markovian assumption of n-gram language modelling, probability of each segment depends only on the previous $n-1$ segments. This makes the sentence probability

to be computed as:

$$Pr(W) = \prod_{i=1}^{N} Pr(w_i|w_{i-1}, w_{i-2}..w_{i-(n-1)}) \tag{2.2.4}$$

The probability of words (or subwords) following each other can be calculated based on a large amount of text data. The most common n-gram models are bigram and trigram, where the history of one or two words or subwords is taken into account, respectively. Since $Pr(W)$ encodes information about grammatically valid sequence of segments, it is also referred to as the grammar model. Back-off probabilities are estimated using Kneser-Ney algorithm to avoid the zero-probable word sequence problem [27].

Decoding Graph

The function of an ASR decoder is to find the sequence of segments which are usually words $W = w_1, ..., w_K$ that is most likely to have generated the observed feature vectors $X = X_1, ..., X_T$.

The decoder tries to find

$$\widehat{W} = \arg\max_{W} \ Pr(W|X) \tag{2.2.5}$$

By the Bayes' rule, this equation could be transformed into an equivalent form:

$$\widehat{W} = \arg\max_{W} \ \frac{Pr(X|W)Pr(W)}{P(X)} \tag{2.2.6}$$

$$= \arg\max_{W} \ Pr(X|W)Pr(W) \tag{2.2.7}$$

The feature vector probability $Pr(X)$ is independent of the word probabilities and hence can be ignored to simplify the equation as in 2.2.7 [18]. The likelihood $Pr(X|W)$ is determined by the acoustic and phonetic models and the prior $Pr(W)$ is determined by the language model.

The probability $Pr(X|W)$ can be further decomposed into

$$Pr(X|W) = Pr(X|P)Pr(P|W) \tag{2.2.8}$$

$$= Pr(X|Q)Pr(Q|P_{CD})Pr(P_{CD}|P)Pr(P|W) \tag{2.2.9}$$

where Q, P_{CD} and P represents the HMM states, context dependent phonemes and context independent phonemes respectively. The link between the word segments W and the context independent phonemes P is provided by the lexical model $Pr(P|W)$, derived from a dictionary associating each word from the language model to a sequence of phonemes as shown in Table 2.1.

The building blocks of ASR are implemented in toolkits like Kaldi [3], in the form of WFSTs [28]. The weights in WFST can be interpreted similar to a probability. The language models are implemented as a grammar WFST, referred to as `G.fst` which encodes the information in $Pr(W)$. It accepts valid sequences of words, and returns both a weight and that same sequence of words. To encode the lexical information about word pronunciation from the PL, Kaldi uses a WFST called `L.fst`. This WFST accepts a sequence of phonemes and returns a word and a weight and it is derived from the probabilities $Pr(P|W)$. The next WFST, called `C.fst`, encodes information about the relationship between context independent phonemes in the lexicon and the context dependent phonemes P_{CD}, whose states form the HMM. The final WFST, called `H.fst` maps the HMM state labels to P_{CD}, ie., $Pr(Q|P_{CD})$ [14]. Once the H, C, L and G WFSTs are created, they are composed from right to left to form a single WFST, after which HMM self-loop arcs are added to form the final `HCLG.fst` which maps state labels to word sequences [17, 29]. While composing, determination and minimisation operations may also be performed at intermediate stages to reduce the graph size and complexity [17].

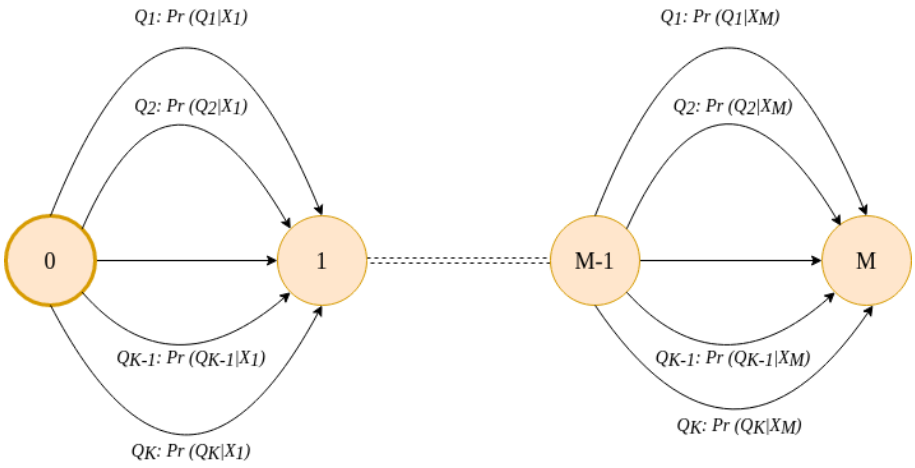


Figure 2.6: Finite state acceptor graph formed by the sequence of M feature vectors extracted from speech to be decoded

During the decoding phase, the acoustic feature vectors corresponding to the speech to be decoded is extracted. If there are M feature vectors, then a finite state acceptor graph with state labels 0 to M is constructed as shown in Fig. 2.6 [29]. If there are K HMM states defined in our acoustic model, then there will be K possible transition paths between every state. These transition paths are given the same labels as that of the HMM states,

$Q_i; i = 1, 2, ..K$ and the corresponding observation probability $Pr(Q|X)$ as weights. These probability values are obtained from GMMs or DNNs based on the type of acoustic model used.

This graph is composed with the ASR model graph `HCLG.fst`, to obtain a search space. A Viterbi decoding is performed on this final graph, by applying beam pruning to reduce the exponential search space [18, 30]. This returns the best decoding path and thus a decoded sequence of words.

2.2.2 End to End (E2E) Architecture

Another major round of revolution in the field of ASR occurred with the advent of an alternate approach of End to End (E2E) ASR. E2E system is a type of ASR model that directly maps the raw audio input to the final text output, without the need for separate acoustic, language and pronunciation modelling stages. E2E models are typically trained on massive datasets (thousands of hours) which are available only to high resource languages. This is because E2E training techniques require thousands of hours of data to learn a mapping of audio directly to characters without any explicit intermediate linguistic representations.

The training techniques that learns sequence to sequence mapping between acoustic feature frames and character sequences has evolved a lot during the past few years [31]. There are explicit alignment approaches like connectionist temporal classification (CTC) [32], recurrent neural network (RNN) [33] transducer and recurrent neural alignment (RNA) [34] and implicit alignment approaches like attention based encoder decoder (AED) methods. In addition to the massive data requirement, the training procedure is compute-intensive with specialised hardware requirement [35].

For low resource languages like Malayalam, where the availability of annotated speech corpora under open licenses is limited to less than 100 hours, training an E2E ASR from scratch is practically impossible. However there is much more text data available to train separate language models. That has motivated the researcher to stick on to the hybrid DNN-HMM architecture, where acoustic models can be trained on the available smaller corpora and leverage the language model to obtain good decoding results [36]. But at the time of writing this thesis, there are many pre-trained E2E models trained using self supervised learning approach [37, 38]. They are trained on many multilingual un-annotated speech datasets, many of which are undisclosed. These pre-trained models can be fine tuned on a small annotated speech dataset, which may be a promising approach for low resource languages.

2.3 Grapheme to Phoneme Conversion Systems

This section focuses on a broad review on automatic G2P conversion systems available in Malayalam and its relation with other world languages. Solutions for automatic G2P conversion in one language may not be the optimal solution applicable for a different language. There are problems with different levels of difficulty that should be solved for each language or language family separately [39].

Malayalam is a morphologically complex low resource language with very little transcribed audio datasets and no openly available pronunciation lexicons. For languages with very little transcribed audio datasets available for speech related tasks, a precise G2P conversion can ensure better acoustic modelling, even in E2E [37] ASR systems.

In many languages, G2P correspondence depends on the relative position of grapheme within a word and a syllable, which makes syllable boundary identification important for phoneme level analysis. Segmenting words to syllables has got further applications in machine translation systems and speech to text systems especially in the context of morphologically complex languages where subword level units improve system performance [36, 40].

Various works on mapping graphemes in Malayalam to phonemes have been reported in literature. There are language-specific as well as language-independent tools. Data driven and knowledge based approaches are the two main G2P strategies. While languages with sufficient amounts of annotated data for training primarily rely on data driven techniques, languages with well documented pronunciation rule sets use knowledge based solutions. In the former, the G2P rules are learned directly from data, whereas in the latter, rules are constructed using linguistic expertise. Most of the G2P tools in Malayalam follow rule-based rather than machine learning approaches. This is primarily due to the simplicity of rule-based approach and secondarily due to unavailability of annotated data for training. Implementation of rule-based approaches require thorough linguistic know-how.

Data driven approaches perform G2P mapping by dictionary lookups [41], decision trees [41], conditional random fields [42], pronunciation by analogy [43] or joint sequence alignments [44]. Recently, deep learning architectures for G2P have been developed based on RNNs [45], convolutional neural network (CNN)s [46] and transformers [47]. Zero shot G2P techniques without explicit training data have been proposed, but they are based on the assumption that similar language families use the same orthography, which is not always true [48]. Phonetisaurus [49] is a data driven tool that learns the mapping rules statistically (joint sequence models) from a training dataset and builds weighted FSTs for G2P conversion. Malayalam does not have a good quality annotated data set for G2P training and a Phonetisaurus model for Malayalam has not been reported yet.

For languages with regular G2P conversion patterns, knowledge based G2P has been reported to produce good results [48, 50]. A set of sequential rewrite rules can be used to achieve this. Agglutinative languages like Turkish [51] and Amharic [52] have reported works on language specific knowledge based G2P conversion using FST technology. Epitran [53], an open source tool using rule based FSTs for G2P conversion of more than 61 world languages recently added Malayalam support, with preliminary mapping between graphemes and phonemes. Hybrid approaches that applies linguistic rules on statistical G2P mappings have been reported for Khmer language [50].

2.3.1 A Review of G2P Tools in Malayalam

This section focuses on G2P tools that works for Malayalam. Being a language with regular orthography, most of the G2P conversion tools in Malayalam follow knowledge based approaches [54–60] rather than data driven methods. The only data driven method is based on encoder-decoder architecture [61] and uses data prepared using an existing knowledge based solution, Unified Parser. Table 2.2 compares the functionalities of different grapheme-phoneme conversion tools available for Malayalam. Here, we examine each of their features and drawbacks in detail.

Table 2.2: Comparing the functionalities and features of G2P conversion tools in Malayalam

Tools	Script Grammar Check	Orthographic Syllabification	Grapheme to Phoneme	Phoneme Delimiter	Phoneme Syllabification	Phoneme to Grapheme	Phonetic Feature Analysis	Open source	Programmable API
Unified Parser [55]			✓	✓	✓			✓	
Espeak [54]			✓	✓	✓			✓	✓
Festvox [56]			✓	✓	✓			✓	
Aksharamukha [57]			✓			✓		✓	✓
Indic NLP [58]		✓	✓					✓	✓
Code-switched [59]			✓	✓					
LTS [60]			✓	✓					
Encoder-Decoder [61]			✓	✓					
Mlphon	✓	✓	✓	✓	✓	✓	✓	✓	✓

- Unified parser [55] is a multi-lingual open source tool for parsing Indian languages and converting it to a common label set of phonemes in syllabified form. In its

language-specific logic, it doesn't take into account any of the Malayalam pronunciation modification rules other than the addition of inherent vowels. This tool does not perform grapheme level syllabification .

- Espeak [54] is an open source speech synthesis system that has a G2P module and it supports Malayalam. Even though phoneme syllabification is supported by Espeak, it does not syllabify graphemes.

- Festvox Indic frontend perform G2P, for TTS systems. It uses X-SAMPA phone set [56]. On analysing the transcription it provides, many contextual rules are observed to be missing for Malayalam and it does not support syllabification of graphemes.

- Aksharamukha [57] script converter is an open source tool that supports G2P for many languages. However language specific contextual logic is lacking for Malayalam. It can not be used to create a pronunciation lexicon, as it does not provide delimiters between phonemes. It syllabifies neither graphemes nor phonemes.

- The Indic NLP library [58], supports syllabification of graphemes and performs G2P. This tool lacks delimiters between phonemes, so it cannot be used to create a pronunciation lexicon.

- FST based G2P mapping for code switched Malayalam-English text has been reported in [59], where English words are phoneme mapped using CMUDict[2] and contextual rule based FST was used for Malayalam. This tool is not open source and hence not freely available for further use, research or analysis.

- Using basic letter to sound (LTS) rules, an automatic pronunciation lexicon creation work was proposed in [60]. It uses a naive Bayes classifier to identify native and English language words and use different set of LTS to perform the G2P mapping. This tool is not openly available for further research and analysis.

- The only deep learning based data driven approach for Malayalam G2P conversion uses Unified Parser for creating the data set for training and testing the model [61]. It has the same features and shortcomings as the Unified Parser tool.

Based on our detailed analysis of the available tools cited above, it was found that none of these tools have full coverage of pronounceable characters defined in Malayalam Unicode. None of these tools could handle the overloading of the letters ഫ (labiodental fricative and and also as labial aspirated plosive) ന (dental nasal and also as alveolar nasal) and their disambiguation. Each of these tools follow different choice of phonetic alphabets and mapping criteria, making them incompatible for a meaningful comparison.

[2]CMUDict- The CMU Pronouncing Dictionary: `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

2.3.2 Ready to Use Pronunciation Lexicon

Availability of a ready to use phonetic lexicon is an essential linguistic resource for ASR and text to speech (TTS) tasks. CMUDict[3] is an open source machine readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations [62]. Similar efforts for creating pronunciation dictionaries for different world languages are reported in literature, namely; Globalphone, providing pronunciation dictionary of 20 world languages [63], The LC-STAR phonetic lexica of 13 different languages [64], Arabic speech recognition pronunciation dictionary with two million pronunciation entries for 526,000 modern standard Arabic words [65], ASR oriented Indian English pronunciation dictionary [66], manually curated Bangla phonetic lexicon of 65k lexical entries prepared for TTS [67] are to mention a few.

However openly available large vocabulary pronunciation lexicon has not been reported for Malayalam, till date. The reported works on Malayalam pronunciation lexicons has mostly been done manually or semi-automatically with a small or medium vocabulary for ASR tasks [68,69]. Agricultural speech and text corpora for Malayalam with 4k manually transcribed phonetic lexicon entries has been reported by Lekshmi et al. [70]. Considering the agglutinative nature of Malayalam language and its practically infinite vocabulary, a manually curated, small sized pronunciation lexicon would be inadequate for general domain speech tasks [71]. Also there could be need for expanding the vocabulary of lexicon as new words get added to the language in the form of proper nouns and loan words. Such lexicons, if available can serve as high quality annotated data sets for bootstrapping data driven G2P training.

2.4 Morphological Complexity Analysis

In linguistic research, measuring the morphological complexity of a language is a crucial area of investigation. Morphology refers to the study of the structure of words, including the formation of words and their grammatical endings, prefixes, and suffixes. The complexity of a language's morphology can be measured by analysing the number and type of morphemes present in its words and examining how they interact with one another to form meaning [72]. The study of morphological complexity is significant because it sheds light on how languages differ from one another and helps us understand the cognitive demands involved in using different languages. Morphological complexity of a language has its impact on applications like ASR where speech to text conversion depends largely on the underlying language model. A measure of the complexity is important for improving and adapting the existing methods of NLP [73]. In this section we discuss what are the different techniques by which morphological complexity are quantified and how are they

[3] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

related.

Morphological complexity can be measured either in terms of the average number of grammatical features getting encoded into a word or in terms of the diversity of word forms occurring in the text corpus of a language. The former approach is called typological analysis and the latter one is called corpus based analysis of morphological complexity [74]. The number of possible inflection points in a typical sentence, the number of inflectional categories, and the number of morpheme types are all morphological complexity indicators [72]. It requires a strict linguistic supervision to analyse each word in terms of its morpheme types to quantify complexity in this manner. The work in [73] suggested estimating the morphological complexity of a language directly from the diverse wordforms over a corpus, which is a relatively easy and reproducible way to quantify complexity without the strict need for linguistic annotated data. A study was conducted to analyse morphological complexity using both human expert judgement and corpus based analysis [74]. Its findings revealed a strong correlation between the two approaches.

Various approaches have been proposed to quantitatively represent the morphological complexity of a language. One such approach, proposed in [75], involves using a text corpus based parameter, namely, type token ratio (TTR) and moving average type token ratio (MATTR), to analyse and compare various European languages. According to the work in [76], MATTR is a reliable measure of linguistic complexity, independent of the total corpus length. Moreover, [75] compared corpus based parameters such as TTR and MATTR with other methods of complexity measures and concluded that both parameters provide a reliable approximation of the morphological complexity of languages. In current research, we rely on the corpus based parameters of TTR and MATTR to quantitatively analyse morphological complexity of Malayalam.

2.5 Subword Based Morphology Aware ASR Systems

ASR is challenging for low resource languages in a morphologically complex setting [7]. Morphological complexity is characterised by productive word formation by agglutination, inflection, and compounding, leading to very long words with phonetic and orthographic changes at morpheme boundaries [77]. Malayalam language is known to have a high level of morphological complexity than many other Indian and European languages in terms of TTR and type token growth rate (TTGR) [9, 71]. This creates a large number of low frequency words and it is practically impossible to build a PL that covers all complex wordforms. Additionally, it introduces the problem of data sparsity in language modeling [36]. Morphology aware ASR systems are designed to improve the accuracy of speech recognition by taking into account the morphological structure of the language by making sufficient modifications in the conventional word based LM and PL.

To build subword level models of pronunciation lexicons and language grammar, the words in conventional LM training corpus has to be separated into smaller units called subwords. The subwords should have some indicators to show, if they belong to a set of subwords that could appear at the end of a word. It is important in ASR decoding because, non word-end subwords should be concatenated with the subsequent subword units to create a complete word [78].

Algorithms to split language model training corpus to morpheme units have been explored for ASR systems in many morphologically complex languages including Finnish, Arabic and Swedish [11, 36]. Data driven algorithms for subword segmentation like byte pair encoding (BPE) and Unigram has also been explored for many other languages including Tamil, Hindi Kannada and Marathi [17, 79, 80]. Different language modelling techniques has been used optimise the performance of automatic speech recognition in Sanskrit [81]. This work compared word level and subword level (byte pair encoding and vowel segmentation) modelling of the text corpus and its impact on OOV word detection and WER [81]. A work on similar lines of Malayalam English code switched ASR has been explored where a hybrid algorithm for subword segmentation was proposed and which reported improved ASR performance [82].

A detailed investigation of various approaches for subword segmentation in Malayalam ASR and its impacts on WER is presented in Chapter 6.

2.6 Malayalam Speech Corpora

Speech corpora with corresponding transcripts is an essential requirement in training ASR systems. We document the list of open licensed Malayalam speech corpora, available for training and testing Malayalam ASR systems.

1. **IIIT Hyderabad (IIITH) corpus**[4] is prepared with the support of the ministry of the Information and Communication Technologies, Government of India. Recorded in a professional recording studio, this dataset contains 1 hour 40 minutes of Malayalam speech by a single male speaker. Each sentence is recorded at 16 kHz and encoded at 16 bits per sample [83].

2. **Indian Institute of Technology Madras (IITM) corpus**[5] is prepared by a consortium of Universities led by the Indian Ministry of Information Technology. It is a corpus primarily aimed for developing TTS for different Indian languages, including Malayalam. Recorded by professional voice talents in an anechoic chamber, the dataset contains 17 hours of Malayalam speech by one male and one female speak-

[4] `http://www.festvox.org/databases/iiit_voices/`
[5] `https://www.iitm.ac.in/donlab/tts/database.php`

ers. Each sentence is recorded at 48 kHz and encoded at 16 bits per sample [84].

3. **Open speech and language resources (Open SLR) corpus** [6] is a crowd sourced, high quality, multi-speaker, multi-language speech dataset by Google. Recorded in a portable 3x3 acoustic vocal booth, the dataset contains 5 hour 30 minutes of Malayalam speech by 24 female and 18 male speakers. Each speech utterance is recorded at 48 kHz and encoded at 16 bits per sample [85].

4. **Malayalam speech corpus (MSC)**[7] is a crowd sourced speech corpus that contains conversational Malayalam sentences recorded by volunteers in natural home and outside environment. The curated collection of MSC contains 1541 speech samples from 75 contributors amounting to 1 hour 38 minutes of speech. Speech files are single channel audio in raw audio format sampled at 48 kHz and encoded with 16 bits per sample [86].

5. **Common Voice**[8] is a crowd sourced collection of speech published by Mozilla foundation. It has about 1 hour of validated speech recorded by 31 speakers and verified by volunteer contributors. The speech files and made available in `mp3` format.

6. **ICFOSS Malayalam speech corpus (IMaSC)**[9] is a studio recorded speech corpus that contains Malayalam sentences read and recorded by 8 speakers [87]. This corpus created with financial support of the International Centre for Free and Open Source Software (ICFOSS) contains of 34473 speech samples from 4 male and 4 female contributors amounting to 50 hours of speech. Speech files are single channel audio in raw audio format sampled at 16 kHz and encoded with 16 bits per sample.

Table 2.3: Details of Speech data sets available under open license so that the derived models can have unrestricted usage.

Corpus	#Speakers	#Utterances	Duration (minutes)	Environment	Year
IIITH [83]	1	1000	98	Studio	2012
IITM [84]	2	8601	838	Studio	2016
Open SLR [85]	44	4025	287	Studio	2019
MSC [86]	75	1541	98	Natural	2020
Common Voice	31	3557	60	Natural	2022
IMaSC [87]	8	34473	30000	Studio	2022

All our experiments on Malayalam speech recognition have used subsets of these corpora for training and testing purposes. The exact usage are clearly indicated in the experiment description section.

[6] `http://www.openslr.org/resources/63`
[7] `https://blog.smc.org.in/malayalam-speech-corpus/`
[8] `https://commonvoice.mozilla.org/en/datasets`
[9] `https://www.kaggle.com/datasets/thennal/imasc`

2.7 Malayalam Speech Recognition Systems

The development of a robust Malayalam ASR system presents several challenges, including the complex phonology and morphology of the language, the wide range of accents and dialects spoken in different regions, and the limited availability of high-quality speech data. To address these challenges, researchers have proposed various approaches for developing Malayalam ASR systems. Several research studies have been conducted in the area of Malayalam ASR, focusing on different aspects of the classical ASR architecture, such as speech data collection, feature extraction, acoustic modelling, language modelling, and grapheme to phoneme conversions. There are recent works on developing ASR systems with modern E2E architecture as well.

Based on the approach used in recognising Malayalam speech, the ASR researches in Malayalam language can be broadly classified as:

1. **Word level Pattern Classification**

   This approach considers speech recognition as a word level pattern classification problem. Features extracted from speech are used as the input and the words to be classified are the output.

   In 2008, Krishnan et al. [88] reported one of the earliest studies on isolated word recognition in Malayalam using an artificial neural network (ANN) based five class classifier. The study utilised five disyllabic words spoken by eight distinct speakers for both training and testing the system. The ANN classifier consisted of 12 input nodes and 6 hidden nodes, with wavelet-based features being fed as the input feature vector. During the testing phase, the system was able to accurately recognise 89% of the presented words.

   In 2010, Cini et al. developed an isolated digit recognition system with a support vector machine (SVM) based classifier that utilised MFCC as the acoustic feature [89]. The system achieved a digit recognition accuracy of 97.6%.

   In 2013, Sunny et al. [90] developed a recognition system for 20 isolated Malayalam words. It was basically a 20 class classifier using ANN with wavelet based features at input nodes. The system demonstrated a word recognition accuracy of 87.5%.

2. **GMM-HMM pipeline ASR**

   This has been the most popular approach in continuous speech recognition that uses HMMs to model phoneme states, which are concatenated to form words and finally sentences. The HMM based phoneme state probabilities with respect to the observed feature vectors can either be generatively trained using GMMs or discriminatively trained using machine learning or deep learning. This section reviews

previous researches based on GMM-HMM approach.

Cini et al. reported a continuous digit recognition system in 2009, which utilised the pipeline architecture with acoustic features such as MFCC and linear prediction coefficient (LPC). The system achieved a high digit recognition accuracy of 99% [91].

In 2011, Cini et al. reported the development of another isolated Malayalam digit recognition system using PLP acoustic features and GMM-HMM based acoustic modelling [92]. The system achieved an accuracy of 99% in digit recognition.

A continuous Malayalam speech recognition system using the pipeline architecture with GMM-HMM for acoustic modelling, n-grams for statistical language modelling and a static PL was reported by Kurian et al. [93] in 2012. The system was developed by training the acoustic models with context dependent triphones as the fundamental speech unit. PLP coefficients were used as the acoustic feature for creating the AM. It was a small vocabulary ASR with 102 words in the PL, created using manual corrections over an automated G2P task. 420 manually transcribed speech samples were created by the authors for the purpose of training and testing. The research reported a word recognition accuracy of 89%.

A dictation system for Open Office Writer was developed by Devi et al. in 2012. It was a continuous speech recognition system trained on 25 hours of an in house speech corpus recorded in an office environment. The ASR system was built using CMU Sphinx toolkit with MFCC as the acoustic feature and GMM-HMM to model the triphone acoustic representation. With a vocabulary of 5000 words, the pronunciation lexicon contained 71 unique phones. However the evaluation of the system in terms of any metric was not reported [94].

An isolated word recognition system proposed by Moneykumar et al., in 2015, is implemented using HTK toolkit. This work compared the effectiveness of GMM-HMM based workflow using HTK toolkit with ANN based classifier and reports a better performance for the latter in the context of recognising 100 isolated words [95].

In 2018, Deekshita et al. [96] reported the development of an in house speech corpora of Malayalam spoken stories that amounted to 203 minutes of audio. This pipeline ASR system was developed using CMU Sphinx toolkit. The GMM-HMM based context independent monophone model was built using MFCC as the acoustic feature. The order of n-gram statistical LM and the vocabulary size of the PL was not specifically mentioned. The system when evaluated on a held out test dataset, reported a WER of 34.8%.

Babu et al. [97] also reported a work on continuous speech recognition system in Malayalam using Kaldi toolkit with pipeline architecture in 2018. It used an in house spoken story database and All India Radio corpora for training and testing. The GMM-HMM based acoustic model was created using monophone and triphone (and its standard variants) speech units. The language models were created on corresponding speech transcripts. The number of entries in the PL was not explicitly reported. The evaluation on different combinations of held out test set demonstrated the impact of domain specific data on the decoding performance. The best WER reported in this system was 34.4%, on maximum likelihood linear transformation (MLLT) based triphone acoustic model.

3. **ANN-HMM pipeline ASR**

   This section reviews research works that used discriminative training criteria to classify phoneme states using ANN framework.

   A continuous Malayalam speech recogniser for a small vocabulary of 540 words were reported in 2012 by Anuj et al [98]. The training and testing was reported on a dataset of 540 words over 180 sentences created by the authors. The system used 25 context independent phonemes (monophones in ASR terminology) as speech units. The acoustic representation of phonemes were modelled using HMM and the hidden state probabilities in it were modelled using multi layer perceptron (MLP) based ANN. Using MFCCs as feature vectors and with no explicit language model, the system reported a WER of 13.3% on the held out test dataset.

4. **DNN-HMM pipeline ASR**

   Here we discuss research works that use discriminative training criteria to classify phoneme states using DNN framework.

   Usage of DNN-HMM acoustic modelling for continuous speech recognition was performed by Moncy et al. in 2020 [99]. It has the highest lexicon size of all previously reported Malayalam ASR of 27500 words. The system has a WER of 3.01% using the the best DNN-HMM acoustic model. The surprisingly low WER be probably due to over-fitting caused by exclusion of the test data set from AM, but not from LM.

   In 2022, open vocabulary ASR system for Malayalam using subword modelling has been explored by Manghat et al. [82]. The experiment was carried out in a private English-Malayalam code switched dataset that resulted in a WER of 39%.

5. **E2E ASR**

   It is difficult to train E2E ASR models from scratch for low-resource languages due

to the limited availability of training data. However, the emergence of transformer models that are pre-trained on large cross-lingual and multi-lingual speech datasets has made it possible to fine-tune them on small labelled target language datasets, resulting in improved E2E ASRs for low-resource languages.

To create an E2E ASR model for Malayalam, Gautham fine-tuned the XLSR- Wav2Vec 2.0 transformer, which was pre-trained on speech data from 53 languages, and achieved a WER of 28% [10].

Another research study by Anoop et al. explored the use of a common phonemic representation label set for Tamil, Telugu, Malayalam, and Sanskrit by fine-tuning pre-trained multilingual speech representation models (Wav2Vec and Indic Wav2Vec) on a small labelled dataset [35]. The study utilised multilingual acoustic models and monolingual language models, resulting in a WER of 19.1% on Malayalam.

The majority of the ASR systems discussed in this section utilised proprietary datasets, while those that used openly accessible datasets did not specify the exact division of their training and testing data. Moreover, due to the lack of a standardised evaluation dataset for Malayalam ASR, comparing the findings of these studies based only on the WER metric would be less meaningful and impractical. However, in the research conducted for this thesis, we have made sure to disclose all the necessary information regarding our training and testing datasets, allowing for future comparisons.

2.8 Summary

In conclusion, this literature review chapter has provided a comprehensive overview of various aspects involved in developing an effective ASR system for the Malayalam language. We started by reviewing different speech recognition architectures and justifying the selection of a hybrid DNN-HMM architecture. We also discussed the importance of G2P conversion systems, pre-built lexicons, and the challenges posed by the high degree of morphological complexity of Malayalam. We explored some of the approaches and techniques that researchers have used to address these challenges and build effective ASR systems for languages with high morphological complexity. We documented the different openly available speech and text corpora in Malayalam for developing speech and language technology applications. Finally, we reviewed previous works on the development of ASR systems for Malayalam, which helped us understand the progress made in this field and the areas that require further attention in developing an effective ASR system for Malayalam. Overall, this literature review will provide the foundation for the subsequent chapters that focus on the development of an ASR system for the Malayalam language.

[10] `https://huggingface.co/gvs/wav2vec2-large-xlsr-malayalam`

Chapter 3

Quantitative Analysis of the Morphological Complexity of Malayalam

3.1 Introduction

To lay the groundwork for the research works that aim to overcome the linguistic challenges in ASR for Malayalam, this chapter takes on the task of quantifying the morphological complexity of the language. To achieve this, we perform a quantitative analysis of the language's morphological complexity on a text corpus containing approximately 8 million words. The analysis utilises key parameters such as TTGR, TTR, and MATTR to determine the extent of the language's morphological complexity. We compare the obtained parameter values with those of other morphologically complex languages, gaining valuable insights into the unique linguistic characteristics of Malayalam. The insights gained from this analysis will provide a better understanding of the linguistic characteristics of Malayalam, which will be useful in developing efficient and accurate ASR systems for the language. By establishing a comprehensive understanding of the morphological complexity of the Malayalam language, this chapter serves as a solid foundation for the subsequent research endeavours in the field of Malayalam ASR.

3.2 Morphological Complexity

Malayalam[1] is a language with complex word morphology. Malayalam words undergo inflections, derivations and compounding producing an infinite vocabulary [10]. As a language with high morphological complexity it has a large number of wordforms derived from a single root word (such as the English words *houses* and *housing*, which stem from the same root word *house*). Morphological complexity can be measured either in terms of the average number of grammatical features getting encoded into a word or in terms of the diversity of word forms occurring in the text corpus of a language. The former approach is called typological analysis and the latter one is called corpus based analysis of morphological complexity [74]. The level of morphological complexity present in a

[1] `https://en.wikipedia.org/wiki/Malayalam`

language can significantly impact applications such as ASR, where the accuracy of speech-to-text conversion largely depends on the underlying language model. Measuring this complexity is crucial in improving and adapting existing methods of NLP to better suit the unique linguistic characteristics of the language [73].

Malayalam has seven nominal case forms (nominative, accusative, dative, sociative, locative, instrumental and genitive), two nominal number forms (singular and plural) and three gender forms (masculine, feminine and neutral). These forms are indicated as suffixes to the nouns. Verbs in Malayalam get inflected based on tense (present, past and future), mood (imperative, compulsive, promissive, optative, abilitative, purposive, permissive, precative, irrealis, monitory, quotative, conditional and satisfactive), voice (active and passive) and aspect (habitual, iterative, perfect) [10, 100]. The inflecting suffix forms vary depending on the final phonemes of the root words. Words agglutinate to form new words depending on the context [101]. Table 3.1 gives examples of a few complex word formation in Malayalam.

Table 3.1: Complex morphological word formation in Malayalam

Malayalam Word	**English Translation**	**Remark**
പെട്ടിയിൽ (peṭṭijil)	in the box	Nominal locative suffix to the word പെട്ടി (peṭṭi, box)
കുട്ടിയോട് (kuṭṭijoːṭ)	to the child	Nominal sociative suffix to the word കുട്ടി (kuṭṭi, child)
ആനക്കുട്ടി (aːnakkuṭṭi)	baby elephant	Compound word formed by agglutination of nouns ആന (aːna, elephant) and കുട്ടി (kuṭṭi, baby)
ആനക്കുട്ടികളോട് (aːn̪akkuṭṭikaɭoːṭ)	to the baby elephants	Nominal sociative suffix to the plural form of the compound word ആനക്കുട്ടി (aːnakkuṭṭi, baby elephant)
ഉണർന്നിരിക്കണ്ട (uɳaṛn̪n̪irikkaɳṭa)	do not stay awake	Negative imperative mood of the verb ഉണരുക (uɳaṛuka, be awake)
പാടിക്കൊണ്ടിരിക്കും (paːṭikkoɳṭirikkum)	will be singing	Future tense iterative aspect of the verb പാടുക (paːṭuka, to sing)

The productive word formation and morphological complexity of Malayalam are documented qualitatively in the domain of grammatical studies. However a quantitative study on the same is not yet available for Malayalam language. Adoption of general NLP solutions of high resource languages like English is not feasible in the setting of morphologically complex languages. A functional morphology analyser, Mlmorph addresses the morphological complexity of Malayalam applying grammatical rules over root word lex-

icon [10]. Quantification of linguistic complexity is important to adapt and improve various NLP applications like automatic speech recognition, parts of speech tagging and spell checking [102–105]. This study aims at quantifying the morphological complexity of Malayalam in terms of corpus linguistic parameters.

3.3 Dataset

This study is performed on Malayalam running text from Wikipedia articles. The Malayalam Wikipedia dump is curated and published by Swathanthra Malayalam Computing (SMC) as SMC Corpus [106]. It consists of 62302 articles. The Malayalam running text often has foreign words, punctuation and numerals present in it. The corpus is first cleaned up to eliminate non Malayalam content and punctuation. It is then Unicode normalised [107]. The cleaned up corpus contained 8.14 million Malayalam words. The nature of the text is formal encyclopedic Malayalam.

3.4 Experimental Details

An element of the set of distinct wordforms in a running text is called a type. Every instance of a type in the running text is called a token. For example, in the sentence, *To be or not to be is the question*, there are 7 types and 9 tokens. The types *to* and *be* repeat two times each. The relationship between the count of types and tokens is an indicator of vocabulary richness, morphological complexity and information flow [73]. The TTR is a simple baseline measure of morphological complexity [75]. TTR is calculated by the formula defined in equation 3.4.1, where V is the count of types and N is the count of tokens.

$$TTR = \frac{V}{N} \tag{3.4.1}$$

The type count gets expanded due to productive morphology and higher values of TTR correspond to higher morphological complexity [74]. However TTR is affected by the token count, N [76]. Larger the corpus, it is more likely that the new tokens belong to the types that have occurred already. The value of TTR gets smaller with the increase in token count. Computing TTR over incrementally larger corpus can indicate how the TTR varies with the token count. In this study, TTR is computed with different token counts starting with 1000 and increasing up to the entire corpus size. This has enabled comparison of Malayalam with the morphological complexity of other languages whose TTR values are available in literature for different token counts.

The TTGR curve is obtained by plotting the graph of token count vs. type count. It indicates how many new types appear with the increase in the token count. If the slope of

the growth rate curve reduces and approaches a horizontal line, at a lower value of token count, it indicates a simple morphology [9]. For a morphologically complex language, the type count continues to grow with the token count [108].

The MATTR computes the relationship between types and tokens that is independent of the text length. Its efficient implementation by Covington et al. has been used by Kettunen to compare the morphological complexity of different European languages [75, 76]. The algorithm to compute MATTR is as described in Algorithm 1 [109]:

Algorithm 1 Computation of MATTR

Require: A text Corpus, C

procedure MATTR
    N ← length of corpus
    L ← length of window (L<N)
    start ← initial position of window
    i = start ▷ index of window position
    while $i \leq (N - L + 1)$ **do**
        $V_i = type\ count\ in\ the\ window\ [i, i + L - 1]$
        $TTR(i) = \frac{V_i}{L}$
        $i = i + 1$
    end while
    $MATTR(L) = \frac{\sum_{i=1}^{N-L+1} TTR(i)}{N-L+1}$
end procedure

The corpus with N tokens is divided into the overlapped subtexts of the same length, say L, the window length. Window moves forward one token at a time and TTR is computed for every window. MATTR is defined as the mean of the entire set of TTRs [76]. In this work, L is chosen as 500, enabling comparison with other languages in the study by Kettunen, where the window length is 500 [75].

3.5 Result and Discussion

Counting the types and tokens on SMC Corpus [106], TTGR and TTR curves are plotted. Fig. 3.1 shows the TTGR curve on the left and the TTR on the right. TTGR curve shows a steep rise initially. As the token count reaches 8 million, the type count is around 1.2 million. But the curve does not flatten even at that token count. This pattern is a common property of Dravidian languages as many unseen wordforms appear as the corpus size is increased [9]. TTR is very high at around 0.82 when the token count is 1000. TTR reduces to around 0.44 when the token count is 0.1 million and finally flattens to a value of 0.16 for the full corpus of 8 million tokens.

To compare the TTR obtained for Malayalam with that of other languages, we have used the results reported for European languages by Kettunen and for Indian languages by Ku-

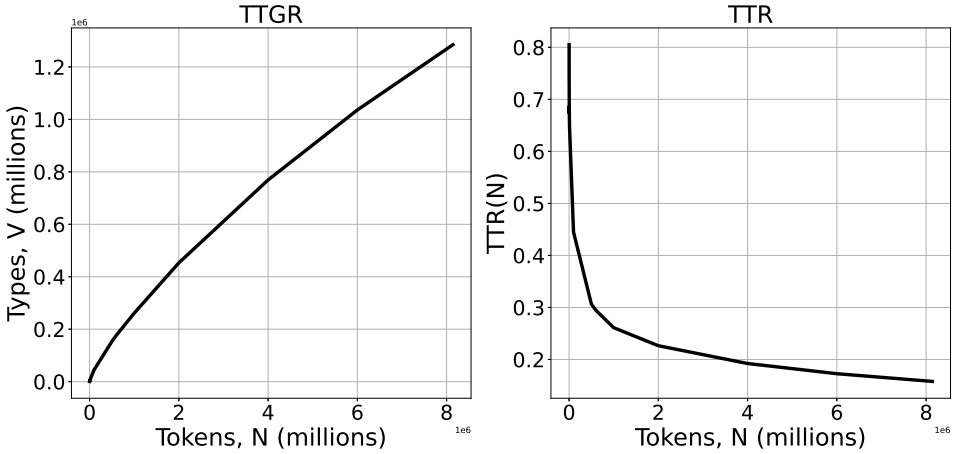
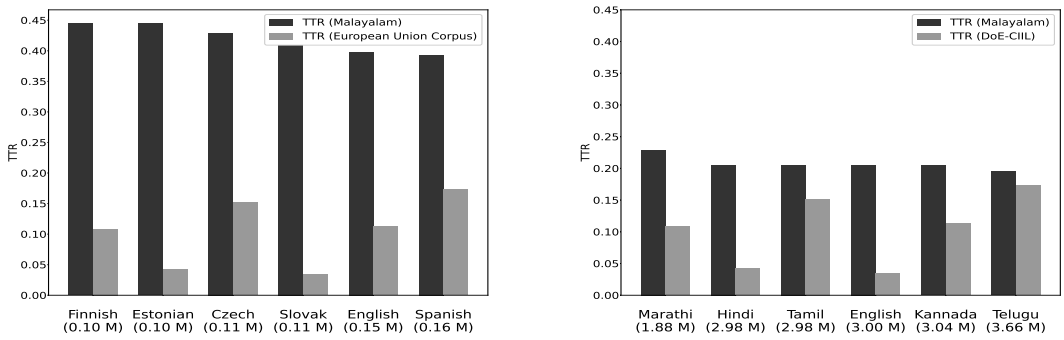


Figure 3.1: TTGR and TTR plot of Malayalam for SMC Corpus of Wikipedia text



(a) **(b)**

Figure 3.2: Comparison of Malayalam TTR with that of European Union Constitution Corpus and DoE-CIIL Corpus

mar et al. [9, 75]. Fig.s 3.2a and 3.2b illustrates the comparison. Only those languages with the highest reported TTRs in the respective works and English are used for comparison. The token size (in millions) used for computing TTRs is indicated for each language. Malayalam clearly shows more morphological complexity than the European languages, Finnish, Estonian, Czech, Slovak, English and Spanish in terms of TTR values. TTR values for Malayalam when compared with other Indian languages Marathi, Hindi, Tamil, Kannada and Telugu clearly indicates a higher level of morphological complexity for Malayalam.

MATTR is computed with window length, $L = 500$ over different segments of the SMC corpus. TTR values for the segments with window position index 1-1000, 5001-6000, 15001-16000 and 18001-19000 are shown in Fig. 3.3. These segments gave MATTR values 0.834, 0.839, 0.836 and 0.800 respectively. Computing MATTR with 0.1 million tokens of SMC corpus resulted in a value 0.806 for Malayalam. Kettunen has reported MATTR values on European Union constitution corpus with each language having a token

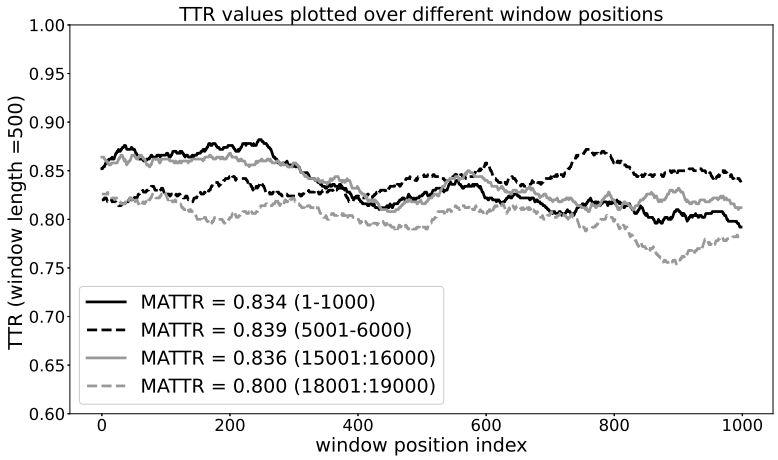


Figure 3.3: TTR plotted at different segments of the SMC corpus for 1000 window positions

count slightly above 0.1 million [75]. The MATTR values reported by Kettunen with the values obtained for Malayalam is plotted in Fig. 3.4. It clearly indicates a higher degree of morphological complexity for Malayalam in terms of MATTR on a formal text corpus. An equivalent comparison with other Indian languages could not be done due to non availability of reported studies.

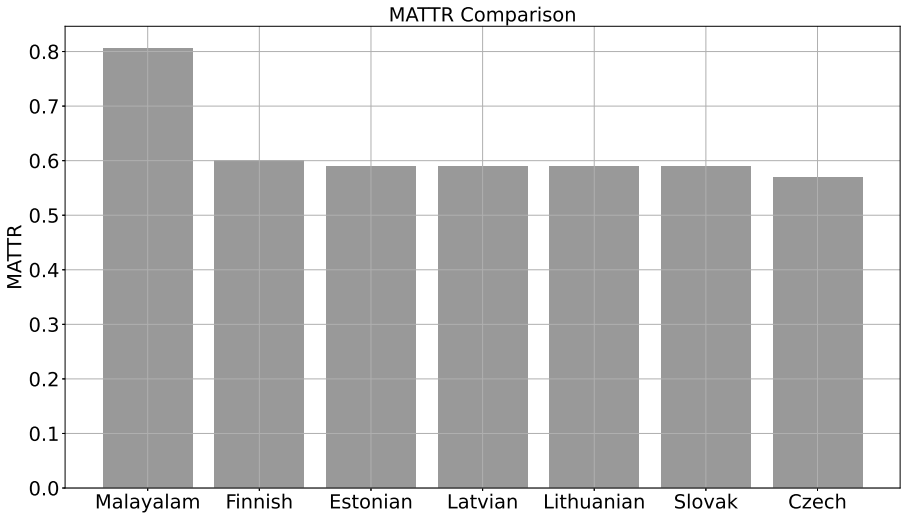


Figure 3.4: Comparison of MATTR values computed for Malayalam on SMC Corpus with that of European Union Constitution Corpus

3.6 Summary

To conclude, this chapter provides a quantitative analysis of the morphological complexity of Malayalam language on a formal text corpus of approximately 8 million words. The analysis has revealed a high degree of morphological complexity in Malayalam, as evidenced by the values of TTR and MATTR. It is essential to consider this aspect of the language's complexity while developing natural language processing applications, such as ASR, spell-checking, and part-of-speech tagging, for Malayalam. To accomplish this, we must create subword-based LMs and PLs for ASR, and perform a morphological analysis of words for POS tagging and spelling correction. By taking these crucial steps, we can develop more efficient and accurate NLP tools that cater to the unique linguistic characteristics of Malayalam.

Chapter 4

Finite State Transducer based Grapheme to Phoneme Conversion

4.1 Introduction

Precise text processing taking care of intricate linguistic details is a pre-requisite for many downstream NLP tasks. This chapter presents the motivation and steps involved in the development of a knowledge based computational linguistic tool, Mlphon, that can solve multiple text processing problems closely associated with speech related as well as some general purpose NLP tasks. Mlphon is built on finite state transducer (FST)s to perform multiple functions including G2P and phoneme to grapheme (P2G) conversions, syllabification on graphemes as well as phonemes, phonetic feature analysis, and script grammar check for Malayalam language. The features of Mlphon are accessible through a programmable Python application program interface (API), that can be integrated with the development process of ASR and TTS systems.

The need to perform precise G2P conversion on demand, to perform syllabification on graphemes as well as phonemes, and to create a programmable API for integrating these functionalities on downstream NLP tasks prompted us to develop the multifunctional tool Mlphon. Our decision to use a knowledge-based approach was driven by the availability of adequate linguistic descriptions. Even though there has been many previous attempts to address one or more of these problems as discussed in section 2.3.1, Mlphon offers certain unique features compared to these prior works which makes it a well suited tool for ASR related tasks in Malayalam.

In this chapter, we will begin by exploring the importance of FSTs in pronunciation modelling and provide an overview of the Mlphon toolkit's architecture. Following that, we will conduct an intrinsic evaluation of the Mlphon toolkit on a gold standard lexicon. We will then showcase our publication of a large vocabulary pronunciation lexicon, consisting of more than 100,000 words. Finally, we will demonstrate the practical applications of the Mlphon toolkit including text sanity check, assisted pronunciation learning and phoneme diversity analysis. This chapter aims to provide a comprehensive understanding of the effectiveness of FSTs in pronunciation modelling for Malayalam.

4.2 Malayalam Grapheme to Phoneme Correspondence

A grapheme is the smallest functional unit of the writing system of a language and a phoneme is the smallest distinguishable sound unit of a language [110, 111]. The correspondence between the two, largely depends on the nature of the writing system. For G2P and P2G conversion tasks, most alphasyllabary languages have precise rule sets, unlike non-phonemic scripts like English [53, 55]. The possible exceptions in rule sets, if any, could be handled by exception dictionaries. For languages with non-phonemic writing systems, when a sufficient amount of annotated high-quality training data is available, data driven solutions are generally preferred to extract linguistic information that is too fuzzy and difficult to be captured by a finite set of rules.

Malayalam is a language spoken predominantly in the state of Kerala in southern India, with about 38 million native speakers. It belongs to the Dravidian language family, and has an alphasyllabary writing system [112]. Though Malayalam script is largely phonemic in nature, there are some unique characteristics like: (i) consonants with and without inherent vowel, (ii) consonant clusters with pronunciation different from the consonants present in them, (iii) special symbol *virama*, that contextually chooses its function depending on its position in a word and (iv) graphemes being overloaded with non-native sounds in loan words. A detailed analysis of the characteristics of Malayalam graphemes, phonemes and the relation between the two are discussed in Appendix I.

4.3 FSTs for Pronunciation Modelling

Rule based mappings between graphemes and phonemes are basically context-sensitive rewrite rules. Each rule specifies how a set of symbols get mapped to another set. FSTs provide efficient methods for performing the composition of such rule sets to single mega rule as described in [113] and [114]. This has made FST popular in many fundamental NLP applications [10, 51–53, 115]. The rule set of Mlphon is written in Stuttgart finite state transducer (SFST) formalism and compiled to FSTs [116].

Given that Malayalam linguistic literature contains well-established pronunciation modelling rules [117–119], we computationally model these rules in a deterministic manner using finite state transducers, which are ideal for this task [113]. FSTs are apt for morphological as well as phonological parsing of natural languages [113]. An FST maps between two sets of symbols. Formally a finite state transducer T can be defined [30] as a set of seven parameters (q, Σ, Γ, I, F, δ, σ) where

q *is a finite set of states.*

Σ *is a finite set of input symbols*

Γ *is a finite set of output symbols*

I *is set of initial states, a subset of* q

F *is a set of final states, a subset of* q

$\delta : q \times \Sigma \rightarrow q$ *is the transition function*

$\sigma : q \rightarrow \Gamma^*$ *is the output function*

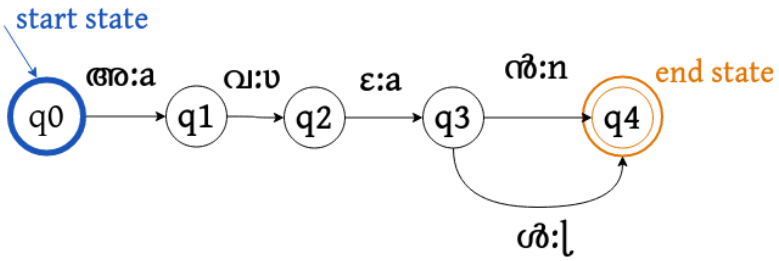


Figure 4.1: An FST representing a simple pronunciation mapping that accepts two words അവൻ and അവൾ.

In an FST, every state, qi, has a finite number of transitions to other states. An input and output symbol is used to label each transition. According to the transition function, the FST emits an output symbol for each symbol in the input string after changing its state, starting from the initial state. When it enters the final state, the FST would have accepted every symbol in the input string. The output string is made up of all the emitted symbols at that point [120]. For the cases where the number of symbols in input or output strings mismatch, a 'null symbol', ε is introduced in the transition mapping.

The FST described in Fig. 4.1, generates pronunciations for two words അവൻ /avan/ and അവൾ /avaɭ/. The states are represented as circles and marked with their unique number. The initial state is represented by a bold circle and final states by double circles. An input symbol i and an output symbol o are marked on the corresponding directed arc as i : o. A special symbol ε indicates the generation of an output corresponding to an empty input string. Here the inherent vowel a is inserted at the transition from the state q1 to q3. Its parameters are defined in Table 4.1.

FSTs satisfy closure property, such that the inversion and composition of transducers are two natural consequences. According to the composition property, if transducer T_1 maps from input symbols I_1 to output symbols O_1 and transducer T_2 maps from O_1 to O_2, then the composition $T_1||T_2$ maps from I_1 to O_2 [30]. The composition of a series of transducers perform the mapping from an input string to output string, passing through the states defined by the constituting transducers. The inversion, T^{-1} of a transducer T, reverses the input and output symbols. This inversion property has enabled the development of Mlphon as a bidirectional G2P converter.

Table 4.1: Parameters of FST illustrated in Fig. 4.1 is defined in this table.

Parameters	Definition
q	{q0, q1, q2, q3, q4}
Σ	{അ, വ, ൻ, ൾ, ε}
Γ	{a, ʋ, n, ⌊}
I	{q0}
F	{q4}
δ	δ(q0, അ) = q1; δ(q1, വ) = q2; δ(q2, ε)= q3; δ(q3, ൻ) = q4; δ(q3,ൾ) = q4
σ	σ(q0, അ) = a; σ(q1, വ) = ʋ; σ(q2, ε)= a; σ(q3, ൻ) = n; σ(q3,ൾ) = ⌊

Mlphon, the tool we introduce is developed using SFST. SFST is programming language for FSTs, written in C++ language [116]. It has a user-friendly Python API[1], freely available under the GNU public license. SFST provides efficient mechanisms for defining the input and output symbol sets for FSTs and the rules for contextually mapping an input string to output string. SFST has been employed in the development of state of the art morphological analysers for Turkish [115], German [121], Latin [122] and Malayalam [10].

The ruleset of Mlphon can be adapted with the necessary script modifications to other Dravidian languages with a similar script nature. The rulesets and graphemes must be adjusted to fit the target language. To enable this, we have made sure the source code is accessible, well-documented, and freely licensed to allow for adaptations[2].

4.4 Architectural Description

The system architecture of Mlphon is described in Fig. 4.2. We follow a modular approach in the design of Mlphon. The mapping from Malayalam script to IPA is carried out in eleven steps, where each step represents an FST. In Mlphon, FST parameters are not directly defined. They are instead compiled from SFST programs. An SFST program is essentially a regular expression. They represent context sensitive rewrite rules. When the programs are compiled, we get eleven transducers shown in the architectural diagram in Fig. 4.2. Each solid rectangular box in this figure represents an FST that maps between two sets of symbols. They are composed at compile time to give final FSTs in dotted rectangular boxes. Mlphon Python library provides programmable access to these final FSTs.

The SFST programs corresponding to the transducers are simplified and described in Algorithms 2 - 5. In the algorithmic description we use the SFST syntax, where '|' indicates

[1]SFST Python library: `https://pypi.org/project/sfst/`
[2]https://github.com/kavyamanohar/mlphon

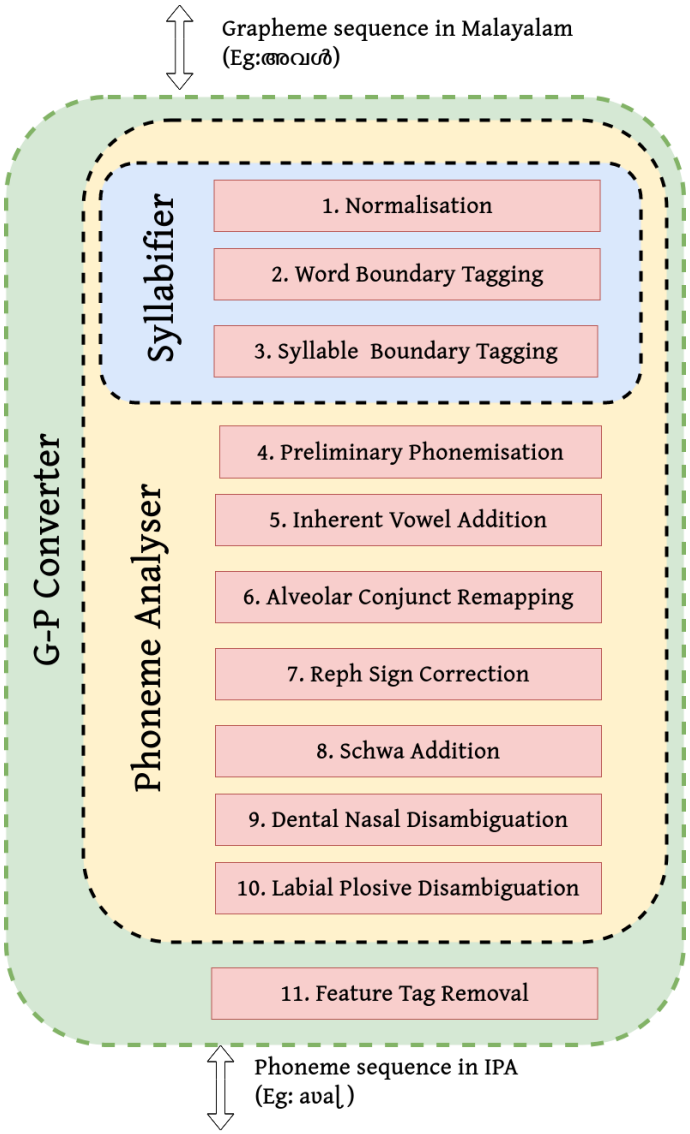


Figure 4.2: The system architecture of Mlphon

the union operation, '||' indicates composition operation, '←' indicates the mapping of the right hand side input symbol sequence to left hand side output symbol sequence. They are represented in the form: D ← [A] B [C], where B is the input with an optional left context A and a right context C, being mapped to the output D. A, B, C, and D can represent a single symbol or a sequence of symbols. The individual symbols in a sequence is separated by a '+', for enhanced readability.

For transducers that carry out complex tasks, the expressions might be quite complicated. In order to create complex expressions from simpler ones, variables are defined [123]. The SFST program is structured as a combination of (i) one-to-one and one-to-many mappings from input symbols to output symbols, (ii) contextual mappings of input symbol sequence to output symbol sequence, and (iii) self mappings where input symbols are passed as such to the output. Additional information is provided in comments in the algorithmic

description. The individual FSTs are composed at compile time to the final FST structures namely:

1. Syllabifier
2. Phoneme analyser
3. Grapheme-Phoneme (G-P) converter

These three FSTs are bundled into Mlphon Python library along with various utility functions and released under MIT license. The programmable Python API enables its integration with many downstream NLP tasks as demonstrated in section 4.10. The functionalities of the individual FSTs are described in sections 4.4.1 to 4.4.11. Wherever it is essential to communicate the functionality, state transitions in each FST are diagrammatically represented.

4.4.1 Normalisation

This FST accepts all Malayalam characters and invisible zero width characters[3]. Characters that do not require normalisation are self mapped. Character sequences that essentially represents the same graphemes are normalised to a standard form.

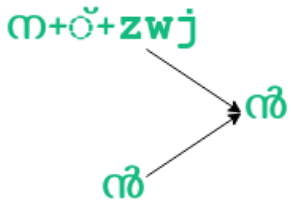


Figure 4.3: Two alternate representations of ൻ at input is being mapped to the normalised form at the output.

Specifically this FST converts chillus represented as the sequence `base consonant`, *virama* (്), `zwj` to atomic forms as shown in Fig. 4.3. It also converts ന്റ /nt̪a/ represented as the sequence [ൻ, *virama* (്), ഠ] to [ന, *virama* (്), ഠ]. Fig. 4.4 provides an example indicating the state transitions happening in an FST that performs this. The word final *chillu* grapheme represented as ന, ്, `zwj` is normalised to a common form of single atomic character, ൻ, by passing through states from q2 , q3, q4 and q5. If the word were already in normalised form, that character is self mapped as indicated in other transitions. The procedural description is provided in Algorithm 2.

[3]Zero Width Joiner: `https://en.wikipedia.org/wiki/Zero-width_joiner`

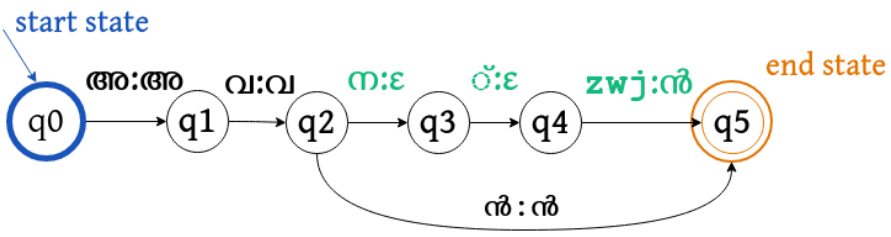


Figure 4.4: Normalisation of the word അവൻ, indicating the two possible input sequences generating the normalised output.

4.4.2 Word Boundary Tagging

This FST accepts all Malayalam characters. The token passed to Mlphon for analysis is considered as a word. Tags in angle brackets <BoW> and <EoW> are added to indicate the beginning of word and the end of word respectively by this FST and is returned to the output. The procedural description is provided in Algorithm 2.

4.4.3 Syllable Boundary Tagging

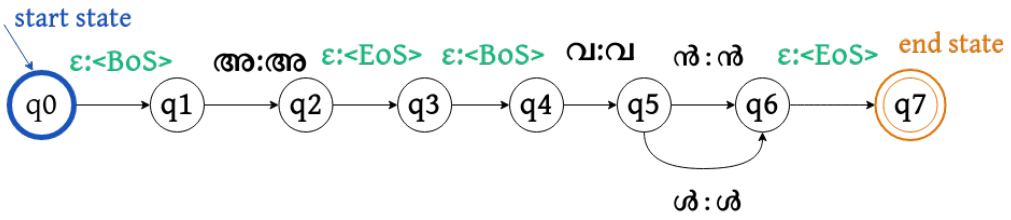


Figure 4.5: Insertion of syllable tags, <BoS> <EoS> are indicated by transitions in green colour. All other symbols are mapped to themselves. Word boundary tags inserted in previous FST is also shown.

This FST accepts all Malayalam characters, along with word boundary tags. As discussed in section I.4, some character sequences are invalid according to Malayalam script grammar. The syllabifier FST checks for validity of character sequences to form syllables. An invalid sequence of Malayalam characters will not find a path from the start state of this FST to the end state and will summarily be rejected. On valid input strings, it inserts tags - <BoS>, <EoS> - at appropriate positions to indicate the beginning and end of the syllables. The syllable boundary tags are essential for pronunciation analysis. This procedure is explained in Algorithm 2. An example for the insertion of syllable tags is indicated in Fig. 4.5.

4.4.4 Preliminary Phonemisation

This FST accepts valid sequence of Malayalam characters separated by word and syllable boundary tags. The transitions defined by this FST maps every grapheme to phonemes

Algorithm 2 Normalisation, Word Boundary Tagging, Syllable Boundary Tagging

procedure Normalisation
    `chillunorm_fst`: chillu ← base consonant+virama+zwj ▷ Define a named FST for chillu normalisation
    `ntanorm_fst`: ന+്+റ ← ൻ+്+റ
    return `chillunorm_fst | ntanorm_fst` ▷ It is the union of two predefined FSTs
end procedure
procedure Word Boundary Tagging
    return <BoW>+token+<EoW> ← token ▷ Insert boundary tags to input word token
end procedure
procedure Syllable boundary tagging
    c_v ← consonant + virama
    syl_end = [anuswara, visarga, chillu] ▷ syl_end is a variable, that can take any value in the list
        ▷ Four types of character sequences that constitute a syllable is defined in the following lines
    Type 1 ← <BoW>+vowel+syl_end? ▷ ? indicates optionaity
    Type 2 ← consonant+vowelsign?+syl_end?
    Type 3 ← c_v * + consonant ▷ * indicates one or more occurence
    Type 4 ← c_v ? + consonant + ◌്? + virama + <EoW>
    syllable ← Type 1 | Type 2 | Type 3 | Type 4 ▷ A syllable is any of the 4 types
    return <BoS> + syllable + <EoS> ← syllable ▷ Insert boundary tags to syllables
end procedure

as per tables in Appendix I I.1-I.3 along with phonetic or graphemic feature tags. The preliminary mapping carried out by this FST will be modified by subsequent FSTs based on contexts. The boundary tags are self mapped, so that they will be retained as such in the output. An example of mapping the graphemes ന and ഩ to its phoneme with phonetic features is described in Algorithm 3.

Algorithm 3 Preliminary Phonemisation

```
procedure Preliminary Phonemisation
    g2p_1: s+<fricative>+<alveolar> ← ന
    g2p_2: m+<labial>+<nasal> ← ഩ
    ...
    ...                                          ▷ Basic G2P mappings
    return g2p_1 || g2p_1 . . .
                                                 ▷ Composition of basic G2P mappings
end procedure
```

4.4.5 Inherent Vowel Addition

Algorithm 4 outlines the rules for adding the inherent vowel /a/ after consonant phonemes in certain circumstances. Specifically, if the consonant phoneme is at the end of a syllable position, or if it is followed by the *anuswara*, *visarga*, or a *chillu*, then the inherent vowel /a/ is added.

4.4.6 Alveolar Conjuncts Remapping

The alveolar consonant clusters ന്റ /nt̪a/ and ട്ട /tt̪a/ are composed of the dental nasal ന /n̪a/ and the alveolar trill ര /ra/, which have distinct pronunciations. Therefore, the grapheme sequence ന /n̪a/ followed by a virama and ര /ra/ can be mapped to /nt̪a/ instead of /n̪ra/. Similarly, the grapheme sequence ര /ra/ followed by a virama and ര /ra/ can be mapped to /tt̪a/ instead of /rra/. To achieve this unambiguous mapping, an FST is used to check the context and remap these phonemes, as described in Algorithm 4.

4.4.7 Reph Sign Correction

If the final consonant in a cluster is the alveolar tap ർ /ɾa/, its pronunciation gets modified to ര /ra/ depending on the preceding consonants. The ർ /ɾa/ sound is retained only if the preceding consonant of the cluster is voiced velar or dental plosive (ഗ /ga/ or ദ /d̪a/) as described in Algorithm 4.

Algorithm 4 Inherent Vowel Addition, Alveolar Conjuncts Remapping and Reph Sign Correction

procedure Inherent Vowel Addition
    pre_context = consonant+<tags> ▷ Define a variable that is a sequence of consonants and tags
    post_context = [<chil>, <anuswara>, <visarga>, <EoS>] ▷ Define a variable that takes any value in the list
    return [pre_context] a+<inherentvowel> [post_context] ← [pre_context] ε [post_context]
end procedure ▷ <tags> - represent the sequence of phonetic feature tags
procedure Alveolar conjuncts remapping
    `tta_fst`: [<BoS>,<virama>]+t̪+<tags>+t̪+<tags> ← [<BoS>,<virama>]+r+<tags>+r+<tags>
    `nta_fst`: <BoS>+n̪+<tags>+t̪+<tags> ← <BoS>+n+<tags>+r+<tags>
    return `tta_fst || nta_fst` ▷ Composition of two FSTs
end procedure
procedure Reph Sign Correction
    return [g,d̪]+<tags>+<virama>+ɾ+<flapped>+<reph> ← [g,d̪]+<tags>+<virama>+r+<trill>+<reph>
end procedure

4.4.8 Schwa Addition (*Samvruthokaram*)

Samvruthokaram is a unique feature of Malayalam. Whenever there are consonants followed by *virama* at word ends, a half-u sound of mid-central vowel schwa is added at word end as described in Algorithm 5. This FST basically disambiguates the function of *virama*. Loan words get adapted to native pronunciation by schwa addition at word ends. eg: ബാങ്ക് /baːŋkə/ (*bank)*

4.4.9 Dental Nasal Disambiguation

The dental nasal grapheme ന /n̪a/, is pronounced as the alveolar nasal /na/ in the following contexts [119]:

1. When a morpheme medial syllable starts in ന and is followed by a vowel sound.

   eg: ആന /aːna/ (*elephant*), ഗാനം /gaːnam/ (*song*), അനുജൻ /anuȷan/ (*younger brother*)

2. When ന is the starting character in a consonant cluster followed by യ /ja/, വ /ʋa/ or മ /ma/.

   eg: നന്മ /n̪anma/ (*virtue*), ന്യായം /njaːjam/ (*justice*), അന്വേഷണം /anʋeːʂaɳam/ (*enquiry*)

3. When ന is the second character in a consonant cluster, beginning with ക /ka/, ഘ

/gʱa/, പ /pa/, മ /ma/, ശ /ʃa/ and സ /sa/.

eg: വിഘ്നം /vigʱnam/ (*blockage*) , സ്വപ്നം /svapnam/ (*dream*), പ്രശ്നം /praʃnam/ (*issue*), സ്നേഹം /sneːɦam/ (*love*)

These three rules are implemented by identifying the context of appearance of ന in terms of the surrounding consonants and syllable boundaries etc. as described in the Algorithm 5.

4.4.10 Labial Plosive Disambiguation

The unvoiced aspirated labial plosive grapheme ഫ /pʰa/ is used to represent the labio-dental fricative /f/ in non-native words. On analysing a corpus of 100k most frequent Malayalam words [124], only 6% of words that contained the letter ഫ were native. All those native words had the letter ഫ, either preceded by the letter സ് or followed by ല. This graphemic context is used as the parameter to determine the word origin and remap fricative to plosive as described in Algorithm 5.

Algorithm 5 Schwa Addition, Dental Nasal Disambiguation, Labial Plosive Disambiguation

procedure Schwa Addition
    `schwa_1`: ə+<schwa>+<EoS> ← u+<v_sign>+<virama>+<EoS> ▷ Define named FSTs for schwa addition
    `schwa_2`: ə+<schwa>+<EoS> ← <virama>+<EoS>
    return `schwa_1 || schwa_2` ▷ Return the composition of two FSTs
end procedure
procedure Dental Nasal Disambiguation
    `nasalrule_1`: <BoS>+n+<alveolar+<virama>+[j,ʋ,m] ← <BoS>+n̪<dental+<virama>+[j,ʋ,m] ▷ Define named FSTs
    `nasalrule_2`: <EoS>+<BoS>+n<alveolar>+[vowel] ← <EoS>+<BoS>+n̪+<dental+[vowel]
    `nasalrule_3`: [k,gʱ,p,m,ʃ,s]+<tags>+<virama>n<alveolar> ← [k,gʱ,p,m,ʃ,s]+<tags>+<virama>n̪<dental>
    return `nasalrule_1 || nasalrule_2 || nasalrule_3` ▷ Return the composition of three FSTs
end procedure
procedure Labial Plosive Disambiguation
    `fa_1`: <BoW>+<BoS>+pʰ<plosive>+a+<EoS>+<EoW> ← <BoW>+<BoS>+f<fricative>>+a+<EoS>+<EoW>
    `fa_2`: s+<fricative>+<alveolar>+<virama>+pʰ+<plosive> ← s+<fricative>+<alveolar>+<virama>+f<fricative>
    `fa_3`: pʰ<plosive>+a+<EoS>+<BoS>+l ← f<fricative>+a+<EoS>+<BoS>+l
    return `fa_1 || fa_2 || fa_3` ▷ Return the composition of three FSTs
end procedure

4.4.11 Feature Tag Removal

The tag-removal FST removes the boundary tags and phonetic feature tags, by mapping them to the null symbol ε. It will leave just the IPA symbols at the output.

4.5 Syllabifier FST

The composition of the series of FSTs from 4.4.1 to 4.4.3 results in a very useful module that performs syllabification of Malayalam text. We compose these FSTs to get the Syllabifier FST and provide programmable access to it in the Mlphon Python library. This module has interesting applications like developing subword level language modelling for ASR as described in section 4.10. An illustration of this module accepting Malayalam text as input and generating output with syllable boundary tags is shown in Fig. 4.6.

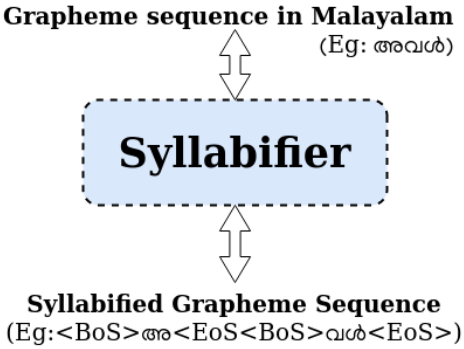


Figure 4.6: Syllabifier performing syllabification on the word അവൾ. Boundary tags for words and syllables are demonstrated.

If the token passed to the syllabifier is അവൾ /avaɭ/, it returns the syllabified string <BoS>അ<EoS> <BoS>വൾ<EoS>. The Python interface to the FST for syllabification, parses the boundary tags and returns the sequence of syllables.

4.6 Phoneme Analyser FST

Phoneme analyser FST is compiled as a composition of 10 FSTs described in sections 4.4.1 to 4.4.10 and indicated in Fig. 4.2 of Mlphon architecture. This FST accepts a grapheme sequence as input and returns phoneme sequence, tagged with their phonetic features. If the token അവൾ is passed to the phoneme analyser FST, it returns the string <BoS>a<vowel><EoS><BoS>ʋ<approximant><labiodental>a <inherentvowel>ɭ<chil><EoS> as illustrated in Fig. 4.7. This module can play crucial role in the context of linguistic learning providing pronunciation information regarding the graphemes in a word.

Fig. 4.8 illustrates the state transitions and the insertion of tags in the phoneme analyser

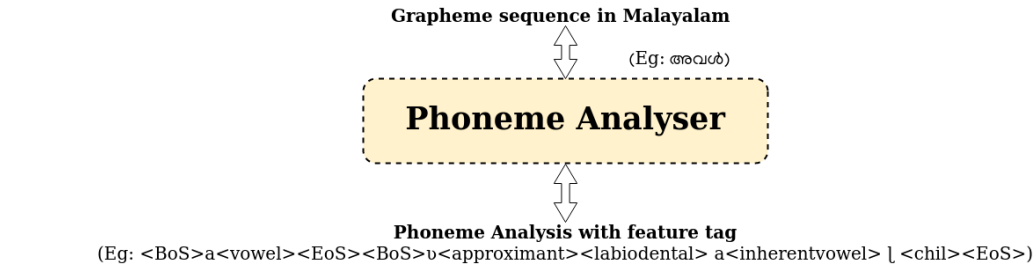


Figure 4.7: Phoneme analyser performing analysis on the word അവൾ. It returns the sequence of phonemes in its pronunciation along with articulatory feature tags in angle brackets.

FST when input tokens passed are: അവൾ /ʌʋaɭ/ and അവൻ /ʌʋan/. The Python interface of Mlphon utilises this FST to analyse the Malayalam word and return the sequence of phonemes and phonetic feature tags like place and manner of articulation.

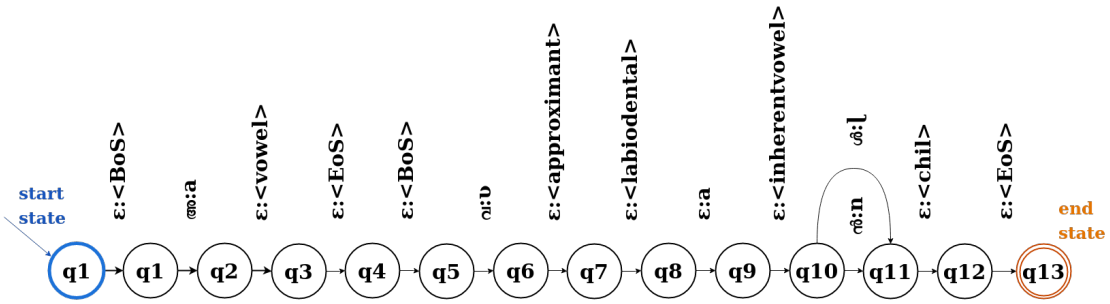


Figure 4.8: Phoneme analyser FST, showcasing grapheme to phoneme conversion on the word അവൾ.

4.7 G-P Converter FST

A transducer that takes in a grapheme sequence and gives out its pronunciation as IPA in analysis mode and does the reverse in generate mode is the bidirectional grapheme-phoneme converter FST. It is marked as G-P converter in Fig. 4.2. All the FSTs previously discussed are bidirectional. However the bidirectionality property is particularly useful when there is need to convert graphemes to phonemes and vice-versa. Fig. 4.9 demonstrates an input and output symbol sequence of G-P[4] Converter FST.

This FST, parses the words അവൾ /ʌʋaɭ/and അവൻ /ʌʋan/ in analysis mode as shown in the Fig. 4.10 (i). When operated in generate mode, it converts a valid phoneme sequence into graphemes. For example, in generate mode, it can parse the inputs ʌʋaɭ and ʌʋan as shown in Fig. 4.10 (ii).

[4]We have used G-P and not G2P, as this FST is bidirectional and can perform P2G as well.

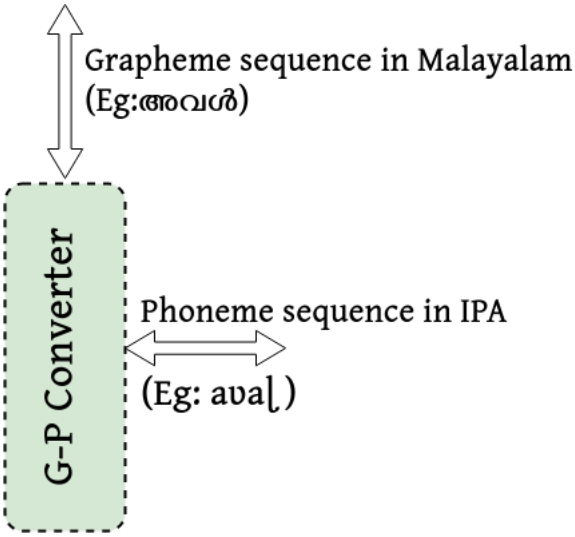


Figure 4.9: G-P Converter FST, performing phoneme analysis on the word അവൾ.

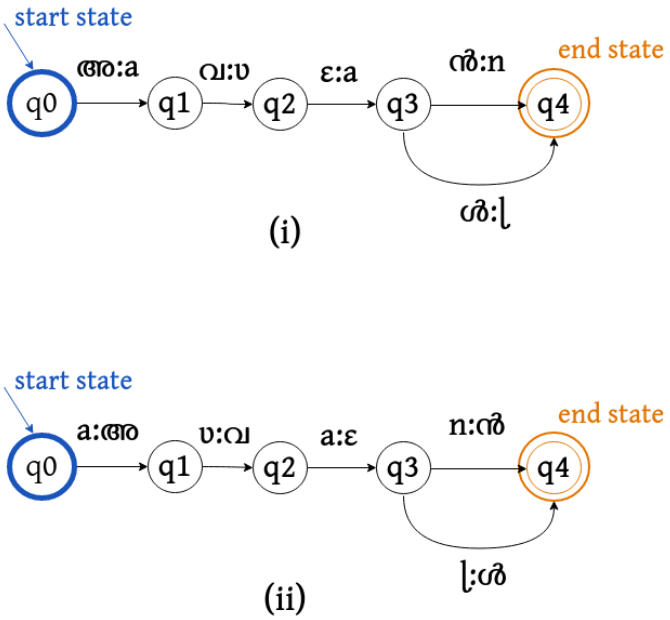


Figure 4.10: G2P converter FST performing (i) grapheme to phoneme conversion on the words അവൾ and അവൻ in analysis mode and (ii) phoneme to grapheme conversion on avaɭ and avan in generate mode.

4.8 The Python Library: Mlphon

The core functionalities of Mlphon is written in SFST and compiled into different finite state transducers. SFST compiles the rules to form minimised FSTs which are very much memory optimised [28]. The python binding of SFST provides access to these transducers for high level programming. Mlphon Python library is very compact with 21 kB of total file size.

One of the major motivation behind this work is to provide pronunciation lexicon for integrating with ASR and TTS applications. The pronunciation lexicon may require the transcriptions to have delimiters between phonemes and/or syllables depending on the application. The utility functions `split_as_phonemes` and `split_as_syllables` provided with Mlphon python library can parse the phonemic analysis to a sequence of phonemes or sequence of syllables separated by spaces. Additionally the function `phonemise` accepts the delimiters defined by the user to separate phonemes and syllables.

The Mlphon library also provides a command line utility for the tasks of syllabification, phoneme analysis and conversion between graphemes and phonemes. See Listing 4.1 for its usage and the list of optional arguments. The entire development process was guided by a set of unit tests to ensure expected functionalities.

Listing 4.1: Command line utility for Mlphon library

```
mlphon [-h] [-s] [-a] [-p] [-pe string] [-se string] [-g] [-i INFILE]
        [-o OUTFILE] [-v]

optional arguments:
-h, --help            Show this help message and exit
-s, --syllablize      Syllablize the input Malayalam string
-a, --analyse         Phonetically analyse the input Malayalam string
-p, --tophoneme       Transcribe the input Malayalam grapheme to phoneme
                      sequence
-pe string, --phoneme_end string
                      String to be inserted at end of phoneme
-se string, --syllable_end string
                      String to be inserted at end of syllable
-g, --tographeme      Transcribe the input phoneme sequence to Malayalam
                      grapheme
-i INFILE, --input INFILE
                      Source of analysis data
-o OUTFILE, --output OUTFILE
                      Target of generated strings
-v, --verbose         Print verbosely while processing
```

4.9 Intrinsic Evaluation of Mlphon

Evaluating a script analysis toolkit like Mlphon is not straight forward due the absence of any baseline ground truth linguistic resource. A gold standard manually annotated data, which can serve as a reference is an important part of any quantifiable evaluation [115]. A gold standard for G2P conversion contains a list of words annotated with their true phoneme transcription. A gold standard for syllabifier is annotated as a sequence of syllables. If a word has multiple possible annotations, all of those should be present in the gold standard lexicon. Before we explain the evaluation, the following section presents the design of gold standard lexicon. It follows a similar procedure and the number of entries as in [115], used for creating a gold standard annotations for Turkish morphological analyser.

4.9.1 Design of Gold Standard Lexicon

The lexical entries in gold standard lexicon are chosen from the IndicNLP Corpus[5] [124] which is a list of words with frequency information. These words belong to a general domain, web crawled Malayalam text corpus of 167.4 million tokens with 8.8 million types.

The manually verified gold standard annotations were created semi-automatically. The process began with the syllabification of 1000 of the most common words in this corpus using Mlphon. It gave back a list of words, some of which had the proper syllabification and others of which had none at all. A small portion of the words that couldn't be syllabified were incorrectly spelled, which is typical of a corpus compiled from web crawls. Misspelt words were manually corrected in the corpus and syllabified. All the syllabifications performed by Mlphon were also manually verified and corrected by expert linguists, if found to be wrong. For the remaining words, which could not be syllabified, manual annotation was performed. Thus the gold standard lexicon for syllabification was obtained.

The gold standard lexicon for G2P mapping followed a similar procedure. Mlphon was used to phoneme map the spelling corrected 1000 words. The returned results were manually corrected for deletion, substitution and insertion errors. The manual corrections were performed, following all the rules and descriptions in the reference books [118, 119]. The removal of word final schwa (*samvruthokaram*) in proper nouns was suggested in a consultation with linguists. The final annotations on the gold standard lexicon were approved by them.

The lexical entries in the gold standard lexicon constitutes 26% of the total number of tokens in the said corpus according to the computation shown in (4.9.1). The frequency profile in Fig. 4.11 illustrates this. The coverage of tokens in the gold standard lexicon

[5] `https://github.com/AI4Bharat/indicnlp_corpus`

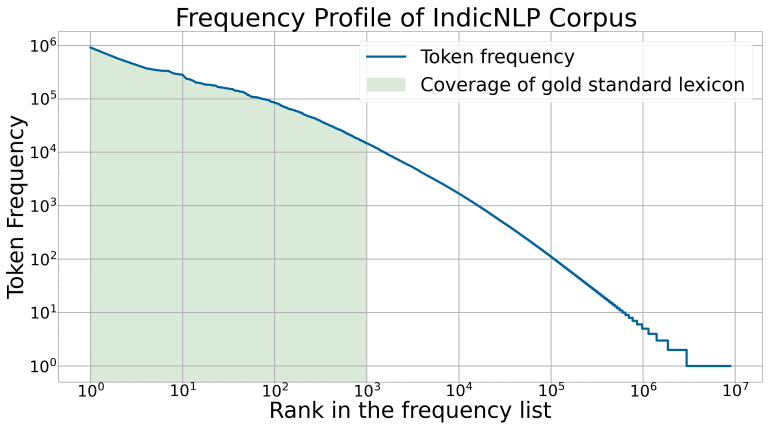


Figure 4.11: The lexical entries of gold standard lexicon is chosen from the most frequent thousand words from the IndicNLP corpus [124] such that these words cover 26% of the 167.4 million tokens present in this corpus.

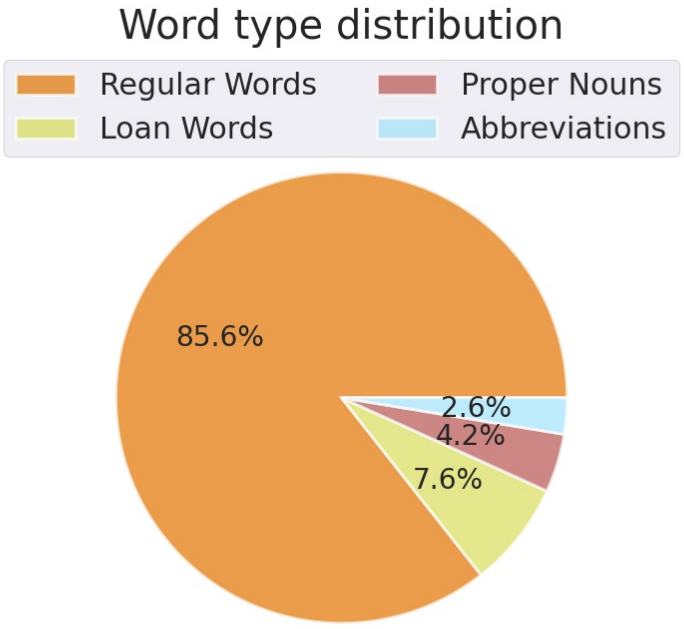


Figure 4.12: Distribution of word types in gold standard lexicon

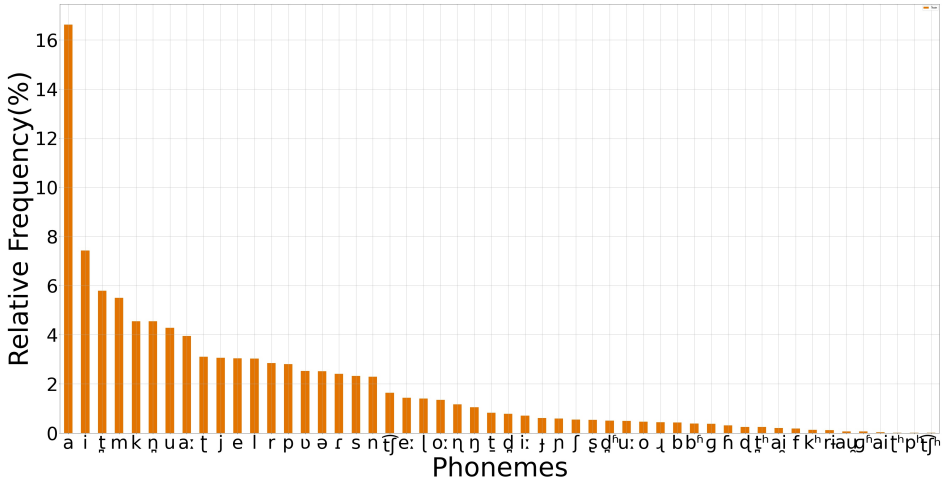


Figure 4.13: Phoneme diversity analysis in gold standard lexicon

with respect to the corpus is computed as:

$$Coverage = \frac{\sum Frequency\ of\ top\ 1000\ tokens \times 100}{Total\ token\ count\ in\ corpus}$$

$$\approx \frac{44321494 \times 100}{167.4\ million} \tag{4.9.1}$$

$$\approx 26\%$$

The gold standard lexicon covers many regular words, loan words, proper nouns and abbreviations as per the distribution illustrated in Fig. 4.12.

A phoneme diversity analysis of the gold standard lexicon was performed and plotted in Fig. 4.13. The relative frequency of phonemes in gold standard lexicon follows the same pattern as previously reported values of phoneme diversity in Malayalam speech corpora [86].

4.9.2 Syllabification

The syllabification results of Mlphon is evaluated on the gold standard reference. Even though the syllabification rules are deterministic there has been few deletion errors as illustrated in the Table 4.2 and analysed in detail in section 4.9.2.

Table 4.2: Comparing the syllabification provided by Mlphon with gold standard reference.

Word	Reference	Mlphon	Error
ഈ	ഈ.	ഈ.	-
ഉന്നത	ഉ.ന്ന.ത.	ഉ.ന്ന.ത.	-
എന്ന	എ.ന്ന.	എ.ന്ന.	-
ഒരു	ഒ.രു.	ഒ.രു.	-
കഫേ	ക.ഫേ.	ക.ഫേ.	-
തോമസ്	തോ.മ.സ്.	തോ.മ.സ്.	-
നമ്പർ	ന.മ്പർ.	ന.മ്പർ.	-
ഫലം	ഫ.ലം.	ഫ.ലം.	-
ഫോട്ടോ	ഫോ.ട്ടോ.	ഫോ.ട്ടോ.	-
സിഐഡി	സി.ഐ.ഡി.		Deletion

Syllabification Accuracy

Accuracy is defined as a ratio between the correctly classified samples to the total number of samples. Precision represents the proportion of positive samples that were correctly classified to the total number of positive predicted samples. Recall of a classifier represents the positive correctly classified samples to the total number of positive samples. The harmonic mean of precision and recall is the F1 score [125]. The evaluation metrics averaged over all syllables and represented as percentage has the values as listed here.

```
Accuracy        : 99%
Precision       : 99%
Recall          : 99%
F1 Score        : 99%
```

Syllable Error Rate

We measure the syllable error rate (SER) based on the number of insertions, deletions, and substitutions for every syllable present in the gold standard lexicon.

```
Total Words: 1000
Total Syllables:  2891
Syllables deleted: 18
Syllables Inserted: 0
Syllables Substituted: 0
Syllables Error Rate: 0.62%
```

Syllable Error Analysis on Different Word Types

Among the words in gold standard lexicon, all syllabification errors were concentrated in words that are English abbreviations directly transliterated to Malayalam without any

delimiters in between. Such words have violated the script grammar of Malayalam with vowels in word medial positions. Some Arabic loan words are among the other valid words that defy script grammar and cannot be syllabified by Mlphon.

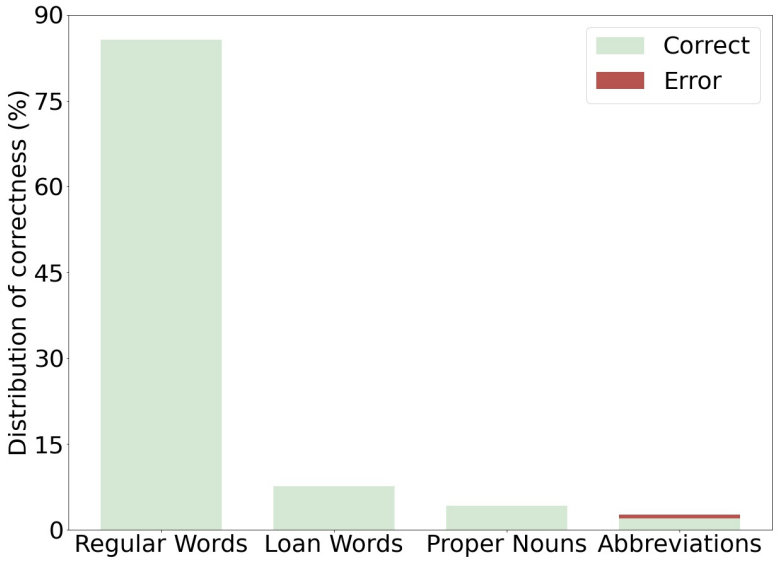


Figure 4.14: Distribution of syllabification errors in different word types in gold standard lexicon

When the word level syllabification errors were examined, 23% of the abbreviations were incorrectly syllabified. This makes up about 0.6% of words in the entire gold standard lexicon. It is illustrated in Fig. 4.14.

4.9.3 Grapheme to Phoneme Conversion

The G2P conversion of Mlphon is evaluated, comparing its output with gold standard phoneme transcriptions. Evaluation involves phoneme level alignment of the transcription provided by Mlphon with that of the gold standard lexicon and counting the number of insertions, deletions and substitutions. We use the toolkit Kaldialign[6] to perform the same. A sample of gold standard transcriptions with the phoneme sequence output provided by Mlphon is shown in Table 4.3.

G2P Conversion Accuracy

Comparing the true phonemes in gold standard lexicons to the transcription provided by Mlphon, we present the phoneme transcription accuracy in the form of a confusion matrix

[6]Kaldialign library `https://pypi.org/project/kaldialign/`

Table 4.3: Comparing the G2P transcription provided by Mlphon with gold standard reference.

Word	Reference	Mlphon	Error
ഈ	iː	iː	-
ഉന്നത	u n n a t̪ a	u n n a t̪ a	-
എന്ന	e n̪ n̪ a	e n̪ n̪ a	-
ഒരു	o ɾ u	o ɾ u	-
കഫേ	k a f eː	k a f eː	-
തോമസ്	t̪ oː m a s	t̪ oː m a s ə	Insertion
നമ്പർ	n a m p a r	n̪ a m p a r	Substitution
ഫലം	pʰ a l a m	pʰ a l a m	-
ഫോട്ടോ	f oː ʈ ʈ oː	f oː ʈ ʈ oː	-
സിഐഡി	s i aɪ̯ ɖ i	s i aɪ̯ ɖ i	-

in Fig. 4.15. For all phonemes other than those listed in Table 4.4, the accuracy, precision, recall, and F1 scores were computed to be 100%.

Table 4.4: Precision, Recall and F1 Scores of phoneme transcription by Mlphon. For all other phonemes, these metrics are evaluated to be 100%.

Phoneme	Precision (%)	Recall (%)	F1 Score (%)
n	100	84	91
n̪	92	100	96
ə	93	100	97

Except for the ന disambiguation rules, all contextual rule sets operate flawlessly without a single error when evaluated on gold standard lexicon. The unintentional insertion of *samvruthokaram* into non native proper names and abbreviations transliterated from English was the cause of all the insertion errors. Insertion is mapped to the empty symbol '#' in the gold standard transcription. The top row of the Fig.4.15 shows insertion of 'ə'. Since the mostly ambiguous grapheme ഫ was G2P mapped with 100% accuracy on the gold standard lexicon, we increased the evaluation space to include 100k common Malayalam words. According to the confusion matrix in Fig. 4.16, the transcription accuracy of ഫ dropped to 99% in the expanded evaluation set.

The overall evaluation metrics averaged over all phonemes in the gold standard lexicon has the values in percentage as listed here.

```
Accuracy        : 99%
Precision       : 98%
Recall          : 98%
F1 Score        : 98%
```

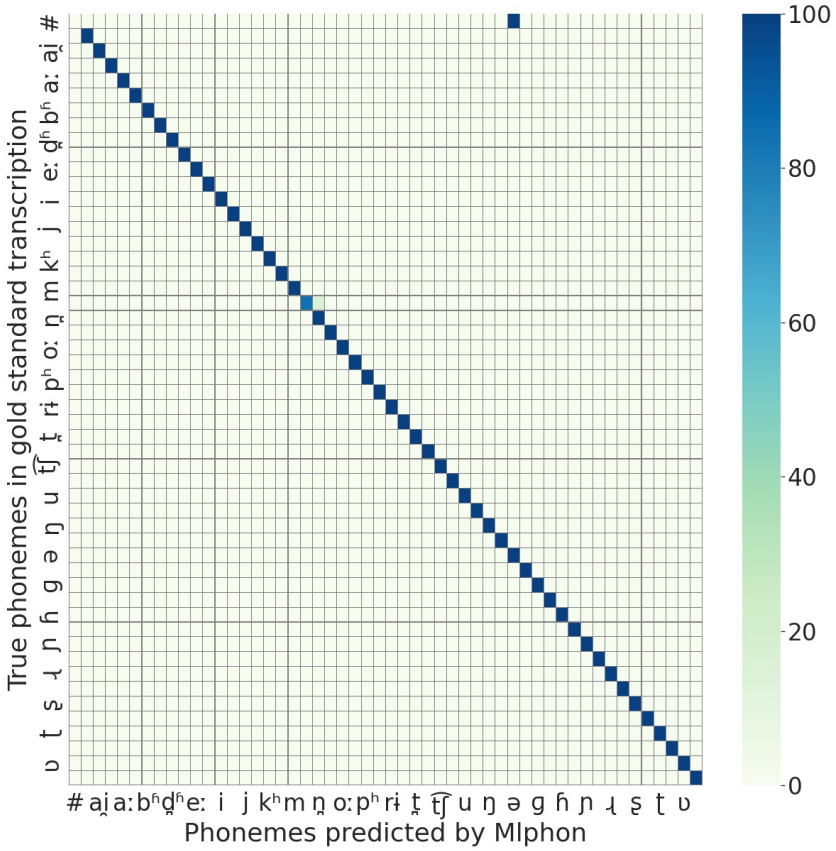


Figure 4.15: Confusion matrix comparing Mlphon transcription with gold standard transcription. The values are normalised and represented as percentage.

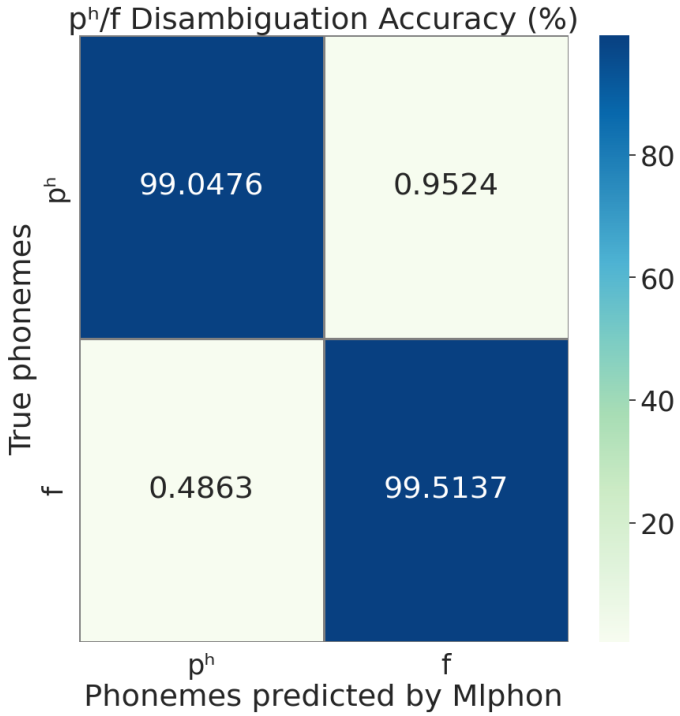


Figure 4.16: In an evaluation space of 100k tokens, we computed the accuracy of transcribing ഫ. The two possible pronunciations /f/ and /pʰ/ were accurately identified in more than 99% of the cases as shown in this confusion matrix.

Phoneme Error Rate

As an alternate metric to measure the phoneme transcription quality, we evaluate the phoneme error rate (PER). It is computed based on the number of insertions, deletions, and substitutions for every phoneme present in the gold standard lexicon.

```
Total Words: 1000
Total Phonemes:  6755
Phonemes deleted: 0
Phonemes Inserted: 12
Phonemes Substituted: 25
Phoneme Error Rate = 0.55%
```

G2P Error Analysis on Different Word Types

We performed a detailed analysis of G2P errors on different types of words in the gold standard lexicon. 1.4% of regular words and 1.3% of loan words had substitution errors. About 23% of proper nouns and 15% of abbreviations had insertion errors due to unintended *samvruthokaram* at word ends. All the erroneous words account for 2.6% of the total words in the gold standard lexicon. It is illustrated in Fig. 4.17.

The correction of substitution and insertion errors involve morphologically analysing the

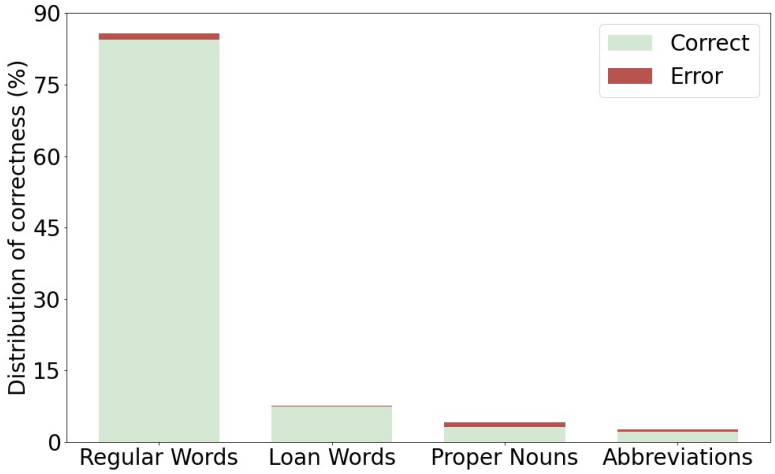


Figure 4.17: Distribution of G2P errors in different word types in gold standard lexicon.

words, which is currently beyond the scope of this work. Even with these limitations, the PER on the gold standard lexicon that covers about 26% of words from 167 million tokens is only 0.55%

4.10 Applications of Mlphon

In this section we describe some potential application of Mlphon. The applications include phonemic diversity analysis of speech corpora, assisted pronunciation learning, text sanity check and correction and creation of large vocabulary pronunciation lexicon. These are described in the following subsections.

4.10.1 Phonemic Diversity Analysis of Speech Corpora

Speech corpus used in developing ASR and TTS systems has to be phonemically balanced and rich to ensure proper acoustic modelling [126]. We transcribed the speech corpus transcript to phonemised text using Mlphon. The phoneme diversity of the resulting text is then analysed. The graph in Fig. 4.18 illustrates the phonemic richness of the corpora used in the ASR experiments described in chapter 5. The phoneme with the highest number of appearances is the inherent vowel /a/, followed by the vowel /i/ in all the corpora under consideration. The most frequent consonant phoneme is the dental plosive /t̪/ in Indic TTS [84] corpus while it is /n̪/ in OpenSLR [85] and Festvox IIITH [83] corpora. The statistical analysis of phonemes could potentially be used to design corpora with phonemically balanced content [127].

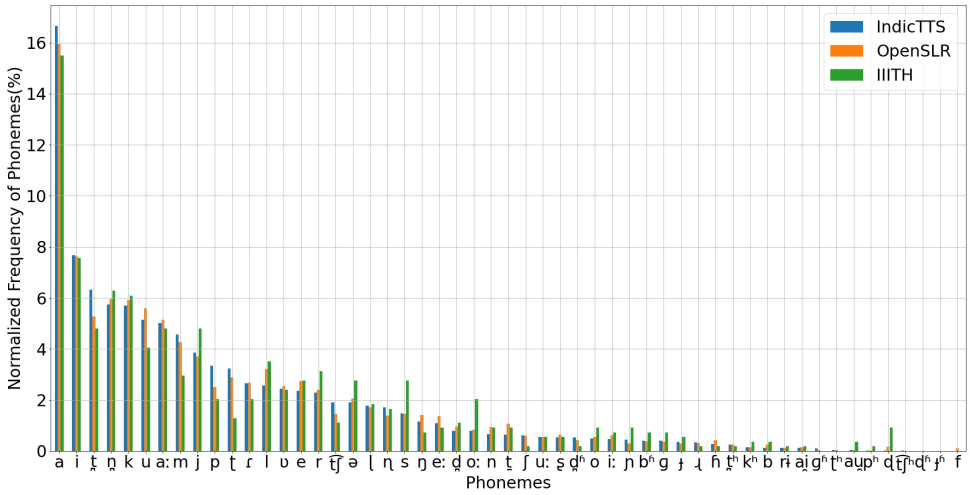


Figure 4.18: Phonemic diversity analysis of various speech corpora used in ASR Experiments. It indicates relative frequency of each phoneme.

4.10.2 Assisted Pronunciation Learning

It is important for a new script learner to understand the pronunciations correctly and get a comprehensive idea of phonetic features of the text. Mlphon provides phonetic feature tags corresponding to every phoneme. A web interface[7] has been developed for user friendly access to Mlphon features. As demonstrated in Fig. 4.19, the graphical user interface accepts a word in Malayalam script and provides syllabification, phonetic analysis and IPA transcription. This interface can aid even a non-native linguistic researcher to analyse and understand the nuances of Malayalam script and pronunciation.

4.10.3 Text Sanity Check and Correction

Large body of text (web crawled, crowd sourced, curated, transcribed or annotated) is the backbone of training and testing modern NLP solutions of large language models, part of speech taggers, text to speech and speech to text systems. Mlphon can perform a script grammar check on the text corpora under consideration and give pointers for manually correcting possibly corrupt Malayalam text content due to presence of invisible characters, foreign scripts, wrong script order etc.

The Table 4.5 lists the number of tokens flagged as errors after script grammar check using Mlphon in various Malayalam speech corpora. These flagged errors were corrected before feeding them for training in ASR experiments explained in chapter 5. However, errors which do not violate the script grammar rules can not be detected by Mlphon.

[7]Mlphon Web Interface `https://phon.smc.org.in/`

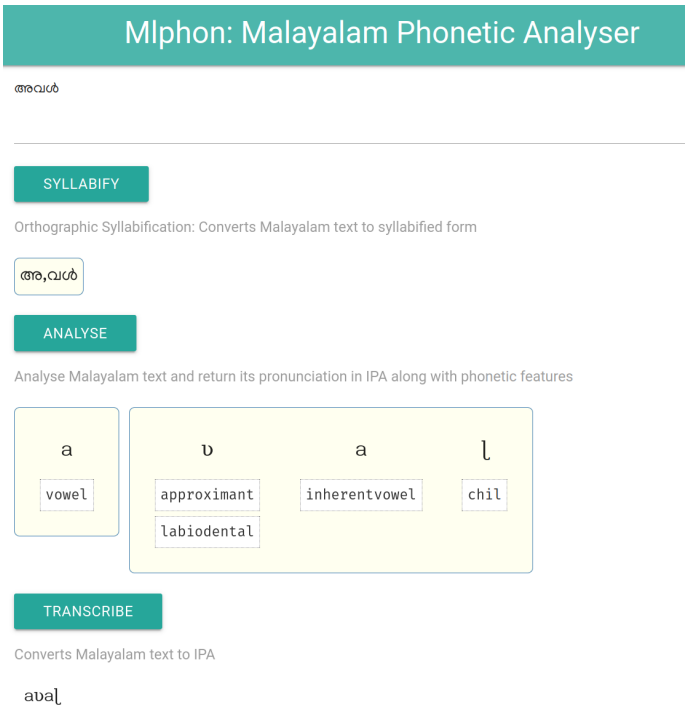


Figure 4.19: Web interface for Mlphon. Features of syllabification, phonetic analysis and IPA transcription are shown.

Table 4.5: Number of word tokens flagged as invalid by Mlphon on different transcribed speech corpus and corresponding error rates.

Speech Corpora	Error Count	Token Error Rate
Indic TTS [84]	1013	1.2%
OpenSLR [85]	83	0.3%
Festvox IIITH [83]	22	0.3%
Indic Speech [128]	4337	4.3%

4.10.4 Creation of Large Vocabulary Pronunciation Lexicon

One of the potential usages of Mlphon is its effectiveness in creating large vocabulary pronunciation lexicon efficiently. A detailed discussion on the creation of such a resource for Malayalam is discussed in Chapter 5 along with a comparison with similar tools qualitatively as well as quantitatively.

4.11 Summary

In this chapter, we presented a FST based G2P conversion system for Malayalam, called Mlphon. The system consists of three FSTs: a syllabifier FST, a phoneme analyser FST, and a G2P converter FST. We also described the various normalisation, tagging, and correction rules used in Mlphon to handle complex phonological processes in Malayalam, such as inherent vowel addition, alveolar conjuncts remapping, and dental nasal disambiguation.

Our intrinsic evaluation of Mlphon showed that the system achieved an accuracy of 99% in syllabification and G2P conversion. We also discussed several potential applications of Mlphon, such as creating large vocabulary pronunciation lexicons, assisted pronunciation learning, and phonemic diversity analysis of speech corpora.

Overall, the development of Mlphon has contributed to the advancement of NLP for Malayalam, and we believe that it can be further improved and expanded in the future to better serve the needs of researchers, developers, and users in the field.

Chapter 5

Large Vocabulary Pronunciation Lexicons for Malayalam ASR

5.1 Introduction

This chapter discusses the development of an LVCSR system for Malayalam using the pipeline approach, which includes building a large vocabulary pronunciation lexicon. A comprehensive and robust PL should be able to handle the language's complex phonology. To achieve this, we utilise an automated approach using Mlphon, an FST-based toolkit developed as part of this research and described in detail in Chapter 4.

Using Mlphon we publish a large vocabulary pronunciation lexicon for Malayalam in both phonemized and syllabified forms. It contains more than 100,000 words covering different categories of words. It is the first of its kind resource available in Malayalam language. To evaluate the effectiveness of the PL created using Mlphon, we compare it with those created using other automated tools such as Unified Parser and Espeak. We perform a qualitative comparative analysis of the PL generated by different tools, while also evaluating the word processing speed of each.

To build the Malayalam LVCSR model, we combine the lexicons generated using automated tools with a statistical LM and a hybrid DNN-HMM acoustic model. Additionally, we conduct a comparative analysis of the effectiveness of the pronunciation lexicon created using Mlphon on the LVCSR task.

5.2 Large Vocabulary Pronunciation Lexicons

In the development of ASR and TTS systems, pre-built PLs are a crucial resource available for numerous languages, as mentioned in section 2.3.2. However, in the case of Malayalam, aside from manually or semi-automatically created small pronunciation lexicons for specific experiments, as discussed in section 2.7, there are no openly available pronunciation lexicons. To address this issue, we present a large vocabulary pronunciation lexicon for Malayalam, created using Mlphon, which is now publicly available.

The published lexicons consist of different categories of words as described in Table 5.1. The tokens in common words pronunciation lexicon are extracted from a general domain

Table 5.1: Pronunciation lexicons of different word categories.

Category	**Number of lexical entries**	**Remarks**
Common words	100000	Most commonly used 100k Malayalam word forms. They are arranged in the order of decreasing frequency.
Verbs	3895	Malayalam verbs in the citation form arranged in alphabetic order
Nouns	59763	Malayalam nouns arranged in alphabetic order
Proper nouns	6751	Common Malayalam person names, place names and brand names arranged in alphabetic order
Foreign words	4350	Sanskrit and English borrowed words commonly used in Malayalam arranged in alphabetic order

Table 5.2: An excerpt from pronunciation lexicons.

Lexicon	Word	Phonemised transcription	Syllabified transcription
Common words	ഒരു	o ɾ u:	o ɾu
	ഈ	i:	i:
	എന്ന	e n̪ n̪ a	e n̪n̪a
	തന്നെ	t̪ a n̪ n̪ e	t̪a n̪n̪e
	പറഞ്ഞു	p a r a ɲ ɲ u	pa ra ɲɲu
	എന്നാൽ	e n n a: l	e nna:l
	എന്നാൽ	e n̪ n̪ a: l	e n̪n̪a:l
Verbs	അകലുക	a k a l u k a:	a ka lu ka
	അഞ്ചുക	a ɲ t͡ʃ u k a	a ɲt͡ʃu ka
	അടക്കുക	a ʈ a k k u k a	a ʈa kku ka
Nouns	അടക്ക്	a ʈ a k k ə	a ʈa kkə
	അടങ്കൽ	a ʈ a ŋ k a l	a ʈa ŋkal
	അടച്ചുര	a ʈ a t͡ʃ t͡ʃ u t̪ u r a	a ʈa t͡ʃt͡ʃu t̪u ra
	അടച്ചുവാറ്റി	a ʈ a t͡ʃ t͡ʃ u ʋ a: t̪ t̪ i	a ʈa t͡ʃt͡ʃu ʋa: t̪t̪i
Proper Nouns	അക്ബർ	a k b a r	a kbar
	അക്ഷയ	a k ʂ a j a	a kʂa ja
	അക്ഷര	a k ʂ a ɾ a	a kʂa ɾa
	അഖില	a kʰ i l a	a kʰi la
	അഖിലൻ	a kʰ i l a n	a kʰi lan
Loan words	അക്കാഡമി	a k k a: ɖ a m i	a kka: ɖa mi
	അക്കൗണ്ട്	a k k a u̯ ɳ ʈ ə	a kkau̯ ɳʈə
	അക്വേറിയം	a k ʋ e: r i j a m	a kʋe: ri jam
	അങ്കിൾ	a ŋ k i ɭ	a ŋkiɭ

text corpus of 167 million types covering the fields of business, entertainment, sports, technology etc. as described in Indic NLP corpus [124]. The rest of the categories are curated word lists from the Malayalam morphology analyser, Mlmorph [10]. Since Mlphon fails to syllabify and phoneme map abbreviations that contain word medial vowels, a work around script has been written to split such words at the position of vowels and obtain the right G2P results.

These pronunciation lexicons are published in two separate formats; one with phoneme level transcription where pronunciation is described as a sequence of phonemes and the other with syllable level transcription where pronunciation is described as a sequence of syllables. The sequences are separated with a blank space in between. The lexicons are published in a two column, tab separated values (tsv) format. Multiple pronunciations of the same word are provided wherever applicable. Table 5.2 gives an excerpt from different categories of pronunciation lexicons. These lexicons are publicly available for download and usage under CC-BY-SA License [1].

The development of a pronunciation lexicon for use in speech tasks like ASR is one of the crucial applications of a G2P conversion tool. Among all the tools previously reported in literature, only Unified Parser and Espeak are freely available to create Malayalam pronunciation lexicons on demand, with delimiters between phonemes. We build lexicons with Unified Parser and Espeak in order to compare and contrast Mlphon's performance with those tools.

Each of the automated tools used to create the pronunciation lexicon, namely Unified Parser, Mlphon, and Espeak, employ different phoneme alphabets and have different G2P mapping criteria. While Unified Parser and Mlphon focus on converting graphemes into phonemes, Espeak generates allophones of phonemes. This means that a single phoneme may be represented by several phonetic symbols in Espeak, depending on its position within a word. In the lexicons created for our ASR experiments, Mlphon has a set of 56 phonemes, whereas Unified Parser and Espeak have 61 and 62 phonemes, respectively. Unified Parser has a higher phoneme count because it differentiates phonemes if they come from different graphemes. In contrast to the other two tools, Espeak uses distinct phonetic alphabets for allophonic variations, giving it a higher phoneme count. Consequently, it is impossible to compare the output produced by these tools in a straightforward, direct, and automated manner.

Sample entries from the pronunciation lexicons created using these tools, are presented in Table 5.3. On analysing these lexicons, following observations can be made:

1. Unified Parser ignores all other contextual rules discussed in appendix I, except inherent vowel deletion in the context of *virama* and other signs.

[1] `https://gitlab.com/kavyamanohar/malayalam-phonetic-lexicon`

2. Espeak implements most of the contextual rules discussed in appendix I, except reph sign, dental nasal and labial plosive disambiguation.

3. Espeak additionally considers allophonic variations due to co-articulation effects. Mlphon does not consider this because this is not a phonemic change.

The carefully crafted pronunciation rules in Mlphon has close to perfect G2P mappings. It makes Mlphon suitable for the creation pronunciation lexicons and for linguistic learning purposes. The unattended contextual rules make Unified Parser less suitable for such tasks. Espeak is suitable to identify allophonic variations. The impact of these lexicons on ASR task is experimentally analysed and presented in this chapter.

Table 5.3: A qualitative linguistic comparison between the lexicons produced by Mlphon and freely accessible automated tools

No.	Word	Unified Parser	Espeak	Mlphon	Remarks
1	കല്പന	k a l p a n a	kɐlbənɐ	k a l p a n a	The third phoneme produced by Unified Parser is different in rows 1 and 2, which should
2	കൽപന	k a l w p a n a	kɐlbənɐ	k a l p a n a	have been same as produced by Espeak and Mlphon
3	അവന്	a w a n	ɐvəniː	a ʋ a n ə	Espeak and Mlphon ensures word end vowel (*Samvruthokaram*) is rightly inserted, corresponding to *virama* at word ends. But Unified Parser does not handle this.
4	നിനക്ക്	n i n a k k	n i n ə kː iː	n̪ i n a k k ə	Only Mlphon disambiguates the dental (n̪) and alveolar nasal (n) pronunciations of ന.
5	കറന്റ്	k a r x a n r x	kɐrəndiː	k a r a n t̪ ə	Unified Parser fails to contextually change the pronunciation of റ, while Espeak and Mlphon handles this correctly

Comparison of Processing Speed

Word processing speed (WPS) is one indicator of a G2P algorithm's effectiveness [39]. The WPS for the applications, Unified Parser [55], Espeak[2] and the proposed tool Mlphon was estimated by measuring the time required to convert the 100k common words in Malayalam listed in Indic NLP corpus [124] as per (5.2.1). Mlphon with a WPS of 69142 words per second is at least ten times faster than Espeak and 1000 times faster than Unified Parser as per the values computed and listed in Table 5.4. This faster processing

[2]Using Phonemizer library `https://pypi.org/project/phonemizer/`

speed of Mlphon makes it particularly suitable for integration with other real time NLP applications.

$$WPS = \frac{100000\ [words]}{Processing\ time\ [minutes]} \tag{5.2.1}$$

The reason for the time efficiency while using Mlphon can be attributed to the computationally fast determinised and minimised FSTs [28], upon which Mlphon is built. Unified Parser is prohibitively slow due to the additional memory management requirement[3].The measurement of G2P conversion speed was performed on a PC workstation with 2 $\times$ AMD CPU @ 2.250 GHz and 4 GB of RAM.

Table 5.4: Comparison of the word processing speed of the proposed tool Mlphon with other openly available tools.

Tool	WPS ($words/minute$)
Unified Parser	42
Espeak	6722
Mlphon	69142

5.3 Experimental Setup for Malayalam LVCSR

With reference to the pipeline architecture of ASR decoder described in Fig. 2.1, we need to build an AM, LM and a PL and compose them into a WFST graph to get the LVCSR model for Malayalam. The acoustic model is trained from an annotated speech corpus and the language model is trained from a huge corpus of text. We have a ready to use PL, which we will adapt for our training speech corpus.

5.3.1 Dataset

We rely on publicly available open licensed read speech datasets [83–85] for Malayalam. Every audio recording in the dataset is associated with a textual transcript in Malayalam script. We divide the available speech into train and test datasets, ensuring non-overlapping speakers and speech transcripts.The train datasets described in Table 5.5 are combined to get approximately 19 hours of audio for acoustic modelling.

To create the language model, we use the sentences from the speech transcripts and combine it with the curated collection of text corpus published by SMC [106]. From this, every sentence that appeared in our test dataset is explicitly removed to prevent over-fitting. The

[3]Solution for segmentation fault error suggested in the discussion forum `https://groups.google.com/g/indictts/c/YUhHfr3Ysug/m/xcflHJTkAQAJ`

Table 5.5: Details of Speech data sets used in our experiments.

Name	Corpus	#Speakers	#Utterances	Duration (minutes)
1	Indic TTS, IITM [84]- Train	2	8601	838
2	Open SLR Malayalam [85] - Train	37	3346	287
T1	Open SLR Malayalam [85] - Test	7	679	48
T2	Festvox IIITH [83] - Test	1	1000	98

resulting text corpus contains 205k unique sentences, 1325k word types, and 356k unique word tokens.

5.3.2 Language Model

Language model predicts the probability of a word to follow a sequence of previous words. Apart from the transcripts of speech which amount to 7924 unique sentences, we have utilized the curated collection of text corpus published by SMC [106] amounting to 205k unique sentences for language modelling. After combining these, we explicitly removed all sentences that are present in our test audio transcripts. Bigram language model is prepared on this language modelling corpus using SRILM toolkit [129]. Back-off probabilities are estimated using Kneser-Ney algorithm to avoid the zero-probable word sequence problem [27].

5.3.3 Pronunciation Lexicon

The entries in the pronunciation lexicon is chosen from the list of words with at least four occurrence in the sentence corpora used for language modelling, ensuring all words in the training speech transcripts are present in the lexicon. The pronunciation lexicon for the ASR are prepared using Unified Parser, Espeak and Mlphon. Due to the differences in grapheme to phoneme mapping approach in different G2P tools,there are differences in the phoneme label set for each of these PLs as shown in Table 5.6.

Table 5.6: Comparison of pronunciation lexicons created by different automated tools

Word	Pronunciations		
	Mlphon	Unified Parser	Espeak
അകം	a k a m	a k a q	ɐ g ə m
അകലം	a k a l a m	a k a l a q	ɐ g ə l ə m
അവൻ	a ʋ a n	a w a nn	ɐ v ə n
അവന്	a ʋ a n ə	a w a n	ɐ v ə n ɨ
എന്നാൽ	en̪ n̪ a: l	e n n aa lw	ʲ e n n a: l
എന്നാൽ	e n n a: l	-	-

In the lexicons with a vocabulary of 69k words, Mlphon has a set of 56 phonemes, whereas

Unified Parser and Espeak have 61 and 62 phonemes, respectively. Unified Parser has a higher phoneme count because it differentiates phonemes if they come from different graphemes. In contrast to the other two tools, Espeak uses distinct phonetic alphabets for allophonic variations, giving it a higher phoneme count. Since acoustic modelling is to learn the relationship between the acoustic features and the phoneme label set, corresponding to every PL, separate acoustic modelling has to be carried out.

5.3.4 Acoustic Modelling

Kaldi toolkit [3] is used for our experiments on ASR. We split the openly available transcribed Malayalam speech corpora from various sources [83–85], into training and test datasets, ensuring non-overlapping speakers and speech transcripts as listed in Table 5.5. The series of steps involved in training a hybrid DNN-HMM acoustic model with LF-MMI training criteria is described here.

Preprocessing

The speech sampling rates of different sources are converted to a sampling frequency of 16 kHz prior to feature extraction. As the acoustic features, we have used 13 dimensional MFCCs with delta and double delta coefficients computed over a window (Hamming) size of 25 ms with an overlap of 10 ms. To reduce the effects of environmental influences on the cepstral features, a cepstral mean and variance normalisation (CMVN) operation is performed on the MFCC feature vectors before training and testing [130].

Context Independent (Monophone) GMM-HMM Training

Monophone acoustic model is a simple GMM-HMM system trained on speech data by extracting 13 MFCCs and their first-order and second-order derivatives from 25 ms speech frames at 10ms intervals. Each phoneme is modelled by a single HMM with three states. The parameters of GMM-HMM (number of Gaussians, means and covariences of Gaussians, transition probabilities between states) as shown in Fig. 2.3, are estimated by iterative Baum-Welch algorithm which belongs to the category of expectation maximisation (EM) algorithms [30].

In each iteration, the Baum-Welch algorithm re-estimates the parameters of the GMM-HMM model, improving itself over previous iterations. If we have the speech, associated transcript and a pronunciation for every word in the transcript, then a flat-start model training starts with the assumption that each phoneme occupies an equal audio space. With this initial guess of flat start alignment, the GMM-HMM parameters are estimated. Starting with one Gaussian per state, we target for a maximum of 1000 Gaussians distributed over all phoneme states.

Re-alignment being an expensive operation, it is performed for every iteration only for the initial 10 iterations; after which re-alignment is performed once for every 2 iterations for the next 10 iterations; followed by once every 3 iteration for the remaining iterations. The total number of iterations is set to 40. We call this the context independent GMM-HMM (CI GMM-HMM) model. After 40 iterations, the resulting alignment is called the monophone alignment and is passed on as input to the next acoustic modelling.

Context Dependent (Triphone) GMM-HMM Training

The acoustic realisation of a phoneme depends largely on its context of occurrence. If we use a single phoneme label to model all the different realisations, the final model would average out all the differences, ultimately reducing the model quality. An alternate approach is to model every phoneme with every possible left and right contexts. If there are N unique phonemes, then there would by N^3 context dependent (CD) phonemes or triphones. In any real speech corpora, all the possible triphones may not be present to train an acoustic model.

A practically feasible middle path is to cluster all similar sounding HMM states together. This is done using phonetic decision trees [20]. A phonetic decision tree groups these triphones into a smaller amount of acoustically distinct units, thereby reducing the number of parameters and making the problem computationally feasible. Starting with single Gaussian per leaf node (tied senone state), we train the context dependent triphone model to get a maximum of 150 leaf nodes and a total of 12000 Gaussians in 35 iterations with Baum-Welch algorithm using MFCC and its derivatives. Re-alignment is done after every 10 iterations. These hyper-parameters were experimentally determined to get the best WER on the test dataset. Once the final context dependent GMM-HMM (CD GMM-HMM) model is trained, the entire corpus is re-aligned to get triphone alignments which are then passed on to the next training stage.

CD GMM-HMM (Triphone) Training with LDA Features and MLLT Transform

An improvement over simple CD GMM-HMM modelling would be to perform feature based linear discriminant analysis (LDA) and model based maximum likelihood linear transformation (MLLT) adaptation. Linear discriminant analysis (LDA) reduces dimensionality and hence compresses the data, and improves inter class separability of phonemes [131]. During this process, 13-dimensional MFCC vectors across 5 frames are spliced resulting in 65-dimensional feature vectors. Then LDA is applied to reduce the vector dimensionality to 40 and increase inter class phoneme variability. MLLT is a transformation on the GMM-HMM model parameters such that it increases the likelihood of training data by performing similar transforms on similar phoneme classes during training and testing [132].

While training LDA-MLLT triphone model, the number of Gaussians is set to 17000 and number of leaves are set to a maximum of 400. The training is done for 35 iterations and alignments are updated every 10 iterations. The resulting alignments improve ASR performance and this is used to bootstrap the next triphone training.

CD GMM-HMM (Triphone) Training with fMLLR Features

A speaker adaptive training (SAT) model is trained by applying speaker specific feature space maximum likelihood linear regression (fMLLR) adaptation on the top of LDA-MLLT transforms [133]. It suppresses speaker variability and hence can build speaker adapted models. To train SAT-fMLLR triphone model, the number of Gaussians is set to 18000 and number of leaves are set to a maximum of 550. The training is done for 35 iterations and alignments are updated every 10 iterations. The resulting alignments improve ASR performance and this is used to bootstrap the TDNN training.

CD DNN-HMM (TDNN) Training

This acoustic model is trained to get the maximum posterior likelihood of the tied CD senone states, given an observation vector. Our implementation of this acoustic model is based on a time delay neural network (TDNN) [23] architecture in Kaldi toolkit [3]. It is trained using frame-level state labelling obtained from CD GMM-HMM SAT acoustic model. State labels are used as targets to train the TDNN acoustic model.

Acoustic features used in TDNN training are: (i) 40-dimensional high-resolution MFCCs extracted from frames of 25 ms length and 10 ms shift and (ii) 100-dimensional i-vectors [19] computed from chunks of 150 consecutive frames. Three consecutive MFCCs vectors and the i-vector corresponding to a chunk are concatenated, obtaining a 220-dimensional feature vector for a frame. i-vectors are a kind of feature containing speaker characteristic information and they have a role in improving the system's adaptation to the specific speech of a speaker [4]. We have used audio data augmentation by speed and volume perturbation before i-vector extraction [134]. When we provide i-vectors at each time-step to a TDNN classifier, we are effectively providing this speaker-specific information so that the system can learn regularities in different kinds of voices, and adjust its phoneme classification depending on them. LDA is applied on this feature vector to decorrelate the components, while the dimensionality is preserved. Consequently, the TDNN input is a 220-dimensional feature vector.

There are 16 layers of TDNNs, each working with different temporal contexts. The TDNN layers 2 to 4 process the input vectors at time indices t-1, t, t+1. The TDNN layer 6 to 13 process the input vectors at time indices t-3, t, t+3. All other layers use no temporal context information. The layers 2 to 13 use factored form of TDNN with subsampled connection

between layers. Each layer is a succession of typical DNN operations, such as affine transforms, ReLU activations and batch normalisations. This acoustic model is trained simultaneously with two discriminative training criteria, one based on cross entropy loss and the other based on LF-MMI. This is effectively a multi-task learning scenario where training is performed efficiently on a GPU, by replacing word level LM with phone level LM [26]. The dimension of the output layer is determined automatically, based on the number of tied phoneme states. The model is trained for 5 epochs where every layer uses L2 normalisation to avoid over-fitting. The model is trained on a single NVIDIA Tesla T4 GPU [4].

5.3.5 Decoding Lattice

The decoding graph is a WFST (`HCLG.fst`) that is composed of the weighted FSTs corresponding to the context dependent HMM acoustic model, lexical model and the grammar model. The final lattice creation for every utterance happens by composing the weighted FSA for that utterance (U) with the `HCLG.fst` [17, 29].

5.4 Result Analysis

All the ASR models use the same bigram language model, with different acoustic models and pronunciation lexicons. The performance of ASR models are evaluated in terms of WER. WER is computed based on the number of words inserted, deleted and substituted in the predicted speech transcript when compared to the ground truth transcript.

$$WER = \frac{(Insertions\ +\ Deletions\ +\ Substitutions)\ \times\ 100}{(Number\ of\ words)} \tag{5.4.1}$$

The out of vocabulary (OOV) rates and dataset characteristics have a significant impact on the ASR results. It is also largely influenced by the domain of text used in language modelling and OOV rates. The OOV rate is the proportion of words in a given speech sample that are not present in the vocabulary of the ASR lexicon. OOV words can not be recognised by the word based ASR decoder. We evaluate our ASR models on two different test datasets namely, T1 (14% OOV) and T2 (1% OOV) derived respectively from OpenSLR [85] and Festvox IIITH [83] corpora that contains 48 and 98 minutes each of speech data. The test data set with lower OOV rate performs better as expected. The resulting WER produced by the lexicons created using all tools under investigation are reported in Table 5.7. The best WERs on T1 and T2 are 34.6% and 9.6% respectively, and they are both given by Mlphon lexicon.

[4] `https://gitlab.com/kavyamanohar/asr-malayalam`

Table 5.7: Comparing WER (%) obtained in Malayalam ASR experiments with lexicons created using the proposed tool, Mlphon, and other openly available tools.

		T1 (14% OOV)			T2 (1% OOV)		
Acoustic Models	**Lexicon Creator**	*Unified Parser*	*Espeak*	*Mlphon*	*Unified Parser*	*Espeak*	*Mlphon*
Monophone (MFCC)		60.9	**58.4**	58.7	25.0	**20.9**	21.8
Triphone (MFCC)		49.9	48.3	**47.4**	21.0	17.5	**17.1**
Triphone (LDA+MLLT)		43.8	**41.2**	43.7	18.4	14.3	**13.9**
Triphone (fMMLR)		43.6	41.2	**41.0**	14.3	11.1	**10.6**
TDNN (MFCC+ivectors)		35.7	34.9	**34.6**	10.7	9.7	**9.6**

These results can be used to deduce some interesting insights about the impact of phoneme transcription quality on WER. It has been found that Mlphon lexicon performs the best with most of the acoustic models, closely followed by Espeak lexicon. The meticulously crafted pronunciation rules have an effect on this improved WER. The context-free monophone acoustic model works well with the Espeak lexicon. This might be as a result of the contextual co-articulation effects being already included in the Espeak lexicon. The Unified Parser Lexicon performs poorly in terms of WER because it ignores the majority of the contextual rules highlighted in Appendix I. This analysis is an indicator of the importance of precise G2P conversion required for speech tasks.

Although there have been previously published works on ASR for continuous Malayalam speech [69, 97, 135], each one was tested using private datasets described in respective papers. The lexicon creation process was not explicitly explained. Additionally, some of these works did not mention the sizes of the pronunciation lexicon and OOV rates, which have a significant impact on the WER. Nevertheless we present a comparison of these previously reported WERs with ours. It is observed that, on two different test datasets of OOV rates 14% and 1%, the proposed ASR with Mlphon lexicon provides similar or better WERs when compared with previously reported WERs as listed in Table 5.8.

5.5 Publication of Malayalam LVCSR Model

The speech recognition model in the form of a WFST graph, `HCLG.fst`, created using the procedures mentioned in this chapter is made publicly available in a format that could be integrated with the open source speech recognition toolkit, Vosk [5].

[5] `https://alphacephei.com/vosk/`

Table 5.8: Comparison of WER from previously reported works on Malayalam ASR. The ASR we built using the lexicon created with Mlphon performs at par with the previously reported works.

ASR Model	Lexicon size	OOV Rate (%)	WER (%)
Deekshita et al. [69]	29k	8	34.2
Lavanya et al. [97]	-	-	34.4
Lekshmi et al. [135]	-	-	10.0
TDNN + Mlphon lexicon	69k	14	**34.6**
TDNN + Mlphon lexicon	69k	1	**9.6**

For the purpose of demonstration, we have integrated the Malayalam ASR model, into a browser based demo page [6]. The model could be loaded to the browser and used for converting speech to text, without passing the speech to server side. A screenshot of the demo page is in Fig. 5.1.

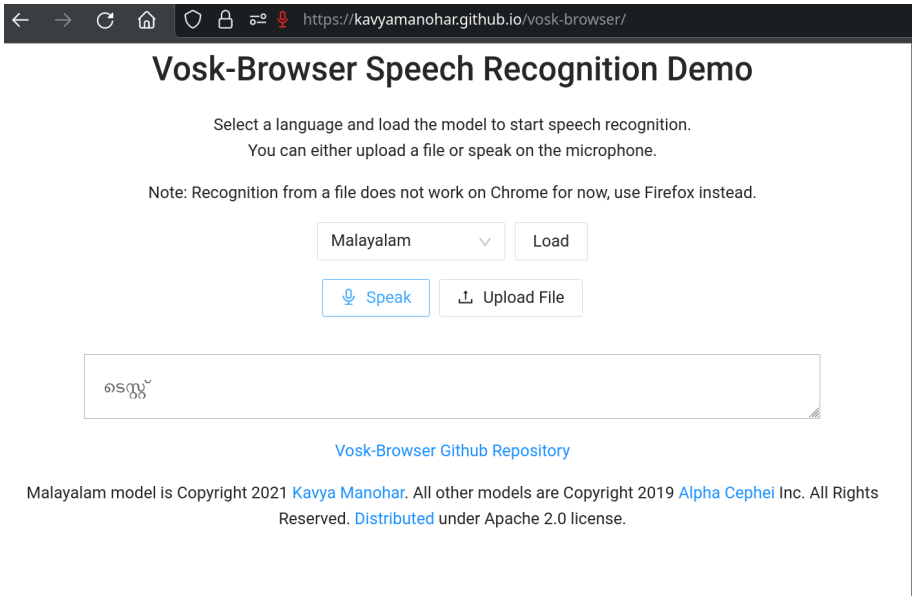


Figure 5.1: The Website demonstrating the ASR developed as per the description in this work

5.6 Summary

This chapter describes the development of an LVCSR model for the Malayalam language. The development of an automated tool called Mlphon for G2P conversion has made it possible to create a pronunciation lexicon for the language. The acoustic model was trained using LF-MMI discriminative training criteria with a TDNN, which is the first attempt of its kind for Malayalam. The resulting LVCSR model, when combined with the pronunciation lexicon built using Mlphon, achieved the best WER on two different test datasets.

[6]Vosk Demo: `https://kavyamanohar.github.io/vosk-browser/`

However, the performance of the model was affected by the OOV rate, indicating the need for alternative language modelling techniques based on subword segmentation. The following chapter discusses these alternative language modelling techniques, which can potentially improve the performance of the LVCSR model for Malayalam.

Chapter 6

Subword based Open Vocabulary Continuous Speech Recognition for Malayalam

6.1 Introduction

In this chapter we shift our focus to subword segments instead of words to generate the language models and pronunciation lexicons. We discuss how subword models would be beneficial to Malayalam LVCSR. We propose two linguistically motivated subword segmentation algorithms for Malayalam and compare their performance with language independent subword segmentation algorithms in the context of open vocabulary pipeline ASR.

6.2 Open Vocabulary ASR

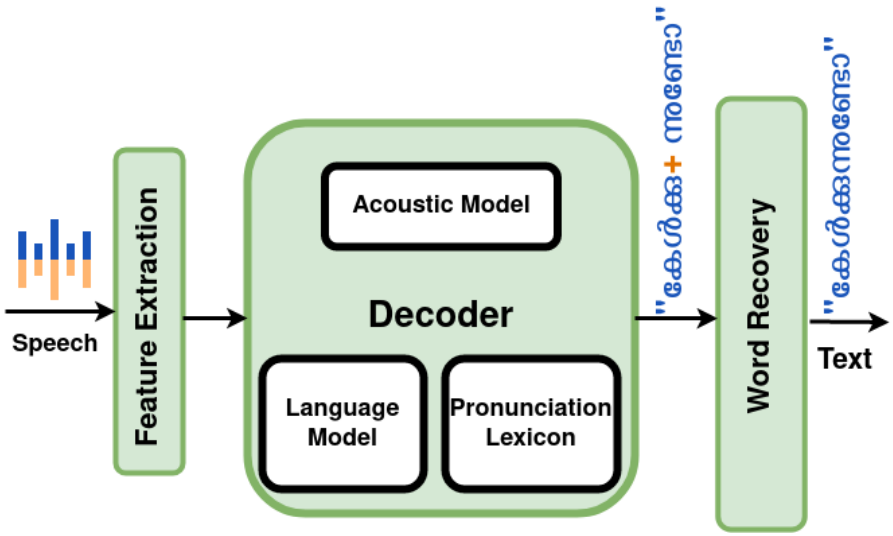


Figure 6.1: Block schematic representation of hybrid ASR system, with subword based language model and pronunciation lexicon

ASR is challenging for low resource languages in a a morphologically complex setting [7]. Morphological complexity is characterised by productive word formation by agglutination, inflection, and compounding, leading to very long words with phonetic and ortho-

graphic changes at morpheme boundaries [77]. Malayalam language is known to have a high level of morphological complexity than many other Indian and European languages in terms of TTR and TTGR [9, 71]. This creates a large number of low frequency words and it is practically impossible to build a pronunciation lexicon that covers all complex wordforms. Additionally, it introduces the problem of data sparsity in language modelling [36].

Deep learning based acoustic modelling improves speech recognition accuracy for languages with abundant training data [136]. Statistical GMM-HMM or hybrid DNN-HMM methods are best suited for languages with limited labelled audio [4]. When much more text data is available than audio data, hybrid models are the preferred choice [5, 6]. Hybrid ASR models require explicit pronunciation lexicons and language models as shown in Figure 1.1. Language modelling of morphologically complex languages can not achieve the word sequence prediction capabilities of simple morphology languages [137, 138]. Additionally, finite sized word vocabulary of a pronunciation lexicon can not cover complex wordforms and loan words that appear in a real world testing. As a result, there is a difficulty in recovering words that are not in the lexicon. [36].

The OOV rate is the proportion of words in a given speech sample that are not present in the vocabulary of the ASR lexicon. OOV words can not be recognised by the conventional word based ASR decoder. Large number of OOV words, and data sparsity are the natural consequences of word based language models in ASR for morphologically complex languages [36]. Segmenting words to appropriate subword units before processing, and later reconstructing them to the whole words is a viable approach to solve the issues of data sparsity and OOV rates. When subword segments are used for language modelling, a dummy symbol is added to identify the positions where the segments can be glued together to form words [78]. When subwords replace words, the ASR vocabulary contains morphemes, syllables, or other character sequences that together can be used to create an unlimited number of words [82, 139].

Subword ASR decoder requires an additional module to reconstruct words from the decoded subword units as shown in Figure 6.1. Language modelling on subword segments in ASR for morphologically complex languages has been extensively studied [139–143] in the context of hybrid ASR systems. Subword segmentation based language modelling has been proposed for applications in automatic speech recognition [36, 81, 82, 139, 144], statistical machine translation [40], neural machine translations [145, 146] and handwriting recognition [147].

The reconstruction from subwords to words is facilitated by adding a dummy marker symbol [36]. In the experiments we perform in this work, we use continuity marker "+" to the right side of the subwords, to indicate another subword has to follow it. In this ap-

proach, reconstruction is straightforward, as the marker indicates the positions for joining the following subword. The Table 6.1 illustrates the usage of continuity marker in our experiments.

Table 6.1: Segmentation illustrating the usage of continuity marker symbol '+'.

Original Text	Segmented text
അവൻ വള ഇടുകയില്ല	അ+ വൻ വള ഇടുക+ യില്ല

6.3 Graphemic and Phonemic Pronunciation Lexicon

For languages where space and punctuation marks act as delimiters between words, segmenting raw text of the language into words is pretty straightforward. However to segment text to subword units, there are data driven as well as linguistically informed algorithms [7, 36]. Morfessor [139, 148, 149], Byte pair encoding (BPE) [145, 150] and Unigram [146] are a few data driven algorithms in popular use. These algorithms do not ensure that subword segmentation happens at valid pronunciation boundaries. This makes precise representation of its pronunciation as a sequence of phonemes impossible.

Table 6.2: Phonemic Lexicon

Word	Pronunciation
SOPHIA	s o ʊ f i ə

For example if the word **SOPHIA** /soʊfiə/ is segmented as **SOP+ HIA**, the pronunciation can not be segmented in a valid way. Then what is viable is to represent each subword segment in the lexicon with its pronunciation described as a grapheme sequence. Tables 6.2 and 6.3 indicate how these entries would be represented in a phonemic and graphemic lexicon respectively. In this work, we use graphemic lexicons and graphemes would be mapped to acoustic features during acoustic model training. Subword based lexicon has the continuity marker "+" indicating it will be followed by another subword segment to complete a word.

Table 6.3: Graphemic Lexicon

Word	Pronunciation
SOPHIA	S O P H I A

Subword	Pronunciation
SOP+	S O P
HIA	H I A

In this work, we propose two linguistically motivated subword segmentation algorithms. First one is a purely linguistic syllabification approach, tailored specifically for Malayalam. The second one is a hybrid subword segmentation algorithm, S-BPE, combining

linguistic syllabification rules with the data driven segmentation method of BPE. We design our experiments to answer two related research questions:

1. How does segmentation method affect the language modelling complexity in a morphologically complex setting?
2. How does the choice of segmentation strategy affect the ASR performance in terms of WER, OOV-WER and model memory requirement?

Due to the low resource nature of Malayalam with limited audio training data but comparatively higher amount of text data, we experiment on hybrid DNN-HMM based ASR for Malayalam, where acoustic and language models are trained separately [5, 6].

6.4 Subword Segmentation Strategies - A Review

A language model estimates the likelihood of word or subword sequences to form a valid sentence. To estimate these likelihood, the raw text of the language has to be segmented into words or subwords. This section explains various segmentation strategies proposed in literature. The suitability of subword segmentation depends on the task (speech recognition, machine translation, text prediction etc.) under consideration. Segmentation techniques that could be adapted for Malayalam language are used in the ASR experiments performed in this research work.

Orthographic syllable based segmentation of text was proposed by Kunchukuttan et al. for statistical machine translation applications [40]. Splitting the tokens based on vowels and adjacent consonants, named Vowel segmentation, was proposed by Adiga et al. and employed in the context of Sanskrit speech recognition [81]. These two methods segment text to syllable-like units at valid pronunciation boundaries.

Several approaches for segmenting Malayalam text to meaningful morpheme units incorporating linguistic knowledge are reported in literature. But for the reasons listed below, none of these could be used for the language modelling task required for ASR. For Malayalam morphological segmentation, earlier studies have used probabilistic, rule-based suffix-stripping, machine learning and dictionary-based approaches [151–154]. The most recent deep learning technique uses Romanised Malayalam text and require annotated data for training [155]. However, none of these research offers an API that can be programmed to perform morphological segmentation for use in downstream applications. The only tool with a programmable interface that works with Malayalam script performs morphological analysis[1] and not morphological segmentation [10]. For example we need the compound word ആനകൾ /a:naka|/ to be segmented as ആന+ കൾ /a:na+/ /ka|/,

[1]Mlmorph: `https://pypi.org/project/mlmorph/`

while its morphological analysis returns മരം<noun><plural>. Morphological analysis is not appropriate for an ASR task, as we expect to piece together the original word from morpheme segments by concatenation. For the ASR task, we therefore do not rely on any linguistically informed morpheme segmentation in Malayalam.

The data driven segmentation methods are designed only based on word spellings and do not have access to pronunciation information. It is therefore possible for these algorithms to break a word sequence into units that do not imply well-formed correspondence to phonetic units. Pronunciation assisted subword modeling (PASM) is an algorithm proposed to solve this issue by using a pronunciation dictionary as an aligner to determine the positions for segmentation [144]. To perform this task, PASM needs a pronunciation dictionary. It uses CMUDict to report the results on English language. But many low resource languages do not have a ready to use pronunciation dictionary. In the comparative analysis performed in this work, we use several segmentation methods that breaks the pronunciation flow. So to be fair in the comparison, we use only graphemic lexicons and not phonemic ones. In this scenario, PASM for segmentation is not used.

For ASR applications in morphologically rich languages, a segmentation technique that ensures rare subwords are included in the vocabulary was proposed by Manghat et al. [82]. This is the first work that uses subword based language modelling in Malayalam - English code switched ASR. Since the algorithmic implementation is not publicly available, we have not attempted its usage in the experiments in this work.

The segmentation techniques already reported in literature and that could be adapted for Malayalam language and employed in the experiments reported in this work are described in the following subsections.

6.4.1 Word Segmentation

In Malayalam, the technique of word segmentation is simple. After removing punctuation, the raw text corpus is divided up by spaces.

6.4.2 Morpheme Segmentation - Using Morfessor

Morfessor is a language independent, data driven method of subword segmentation. The Morfessor baseline algorithm is based on minimum description length principle [148]. It is an unsupervised technique in which frequently occurring sub strings in several different word forms from the raw training text corpus are proposed as morphs (or morpheme-like units) and the words are then represented as a concatenation of morphs [139]. Its current version, Morfessor2.0, has a Python interface that may be customised and it supports annotated training data as well [149]. The Morfessor method guarantees neither segmentation at appropriate pronunciation boundaries nor segmentation into meaningful units.

6.4.3 BPE Segmentation

BPE is a data driven algorithm that determines the optimal set of subword tokens through an iterative process. It was originally proposed as a data compression algorithm [150]. BPE algorithm splits the training data into characters and create an initial vocabulary. During further iterations, the most frequent character bigrams are determined, merged into a single token and added to the vocabulary. The process is continued until a desired number of merge operations are performed. The final vocabulary size is the sum of initial vocabulary and the number of merge operations, which is a hyper-parameter. The subword tokens in the learnt vocabulary are later used to segment any text. BPE ensures that the most common words are represented in the pronunciation dictionary as a single token while the rare words are broken down into two or more subword segments [145]. BPE segmentation algorithm available in `subword-nmt` python library is used in the experiments described in this work[2].

6.4.4 Unigram Segmentation

Unigram segmentation [146] is another language independent segmentation algorithm. It makes the assumption that the subwords of the sentence are independent of one another. The vocabulary of desired size is built from a heuristically large vocabulary by retaining only $\eta\%$ (say $\eta = 80$) of the subwords in each iteration, and discards the rest. The top 80% of subwords are obtained by ranking all subwords according to the likelihood reduction on removing it from the vocabulary. The most probable segmentation of an input sentence is determined by the Viterbi algorithm. The Unigram segmentation algorithm is available in the open source python library, `sentencepiece`[3], which is used in the experiments performed in this work.

6.5 Proposed Subword Segmentation Algorithms for ASR

In this section, we present the details of our proposed subword segmentation algorithms. The first one is the FST based syllabification algorithm and the other is the hybrid algorithm combines the data driven approach of BPE with linguistic information about syllables.

6.5.1 Syllable Segmentation

To performs segmentation that aligns with the Malayalam script, it is important to analyse the nature of grapheme inventory of Malayalam. The graphemes in Malayalam script are

[2]subword-nmt: `https://pypi.org/project/subword-nmt/`
[3]sentencepiece: `https://pypi.org/project/sentencepiece/`

classified as: (i) Vowels, (ii) Vowel signs, (iii) Regular Consonants, (iv) Special consonants : *anuswara*, *visarga* and *chillu*, and (v) Multi-functional character: *virama*. Vowel graphemes occur only at word beginnings. Regular consonants inherently have the vowel /a/ present in them. Vowel sounds at positions other than word beginnings are represented by vowel signs. Vowel signs modify the inherent vowel sound of the consonants. A consonant cluster, also known as a conjunct, in Malayalam is a sequence of consonants separated by *virama* in between, where *virama* kills the inherent vowel from the preceding consonant [118]. *Chillus* are special consonants that do not have inherent vowel associated with them. The characteristics of other special consonants and *virama* are marked in Table 6.4. The syllabification process will make use of these linguistic rules.

Table 6.4: Special consonants and Virama sign in Malayalam

Character	Properties
Anuswara	Represents /m/ at syllable ends
Visarga	Introduces aspirated glottal stop
Chillu	Dead consonants with no inherent vowel
Virama	Kills Inherent vowel in conjuncts and inserts schwa at word ends.

A syllabification algorithm tailored for Malayalam script using finite state transducers has been introduced in Chapter 4. The linguistic rules for syllable segmentation has been computationally implemented as in Algorithm 6 and made available as part of the `Mlphon` Python library[4]. We use Mlphon, to perform syllable segmentation. This results in a variable length subword segments where each segment is a syllable with valid pronunciation.

Algorithm 6 FST based Syllabification Algorithm

procedure Syllable boundary tagging
    c_v ← consonant + virama
    Type 1 ← <BoW> + vowel+[anuswara, visarga, chillu] ? ▷ ? indicates optionality
    Type 2 ← consonant + vowelsign ? + [anuswara, visarga, chillu]?
    Type 3 ← c_v * + C ▷ * indicates one or more occurence
    Type 4 ← c_v ? + consonant + ◌്? + virama + <EoW>
    syllable ← [Type 1, Type 2, Type 3, Type 4] ▷ Defines a syllable
    `SyllableBoundaryTagger`: <BoS>+ syllable + <EoS> ← syllable
end procedure

6.5.2 S-BPE Algorithm

Syllable-BPE (S-BPE) is a hybrid algorithm that takes into account syllables as irreducible units in the same way that BPE takes into account characters. S-BPE algorithm needs to learn the syllable sequence frequencies from a training corpus, before it can be applied

[4] `https://pypi.org/project/mlphon/`

to segment words. During training, it creates a symbol vocabulary of most frequent syllable sequences. During segmentation, the algorithm compares the text with the symbol vocabulary file and segments the words into subwords that are most frequent sequence of syllables. The algorithmic implementation has been adapted from the original BPE algorithm in `subword-nmt` python library, and made available under MIT License [5].

Training

To begin with, the words in the S-BPE training corpus are syllabified using the Algorithm 6. A symbol vocabulary, V is created by populating it with the unique syllables. The words in the training corpus is searched to determine the most frequent symbol bigram and that bigram is then merged and added as a new entry to the symbol vocabulary. All the occurrences of this bigram will be replaced in the training corpus using the newly merged symbol bigram. This step is repeated for a desired number of times, k as described in Algorithm 7. The number of merge operations is set to $k = 10000$, in our experiments.

Algorithm 7 S-BPE Training Algorithm

Require: Set of strings D, Number of merges k

procedure S-BPE(D, k)
    Vocab ← All unique syllables in D
    $merge\_counter$ = 0
    while $merge\_counter$ < k **do** ▷ Merge frequent bigrams
        S_L, S_R ← Most frequent bigram in D
        S_{new} ← S_L+S_R ▷ Merge most frequent symbols
        Vocab ← Vocab + S_{new}
        Replace each occurrence of S_L, S_R with S_{new}
        $merge\_counter$ = $merge\_counter$ + 1
    end while
end procedure

Segmentation

First, the text to be segmented is syllabified using Algorithm 6. Then every instance of the syllable sequence S_L, S_R in the corpus to be segmented is replaced with S_{new}, in the order in which those symbols were learnt and added to the vocabulary, Vocab. S-BPE ensures that the most common words are represented in the vocabulary as a single symbol while the rare words are broken down into two or more subword segments while guaranteeing each segment has a valid pronunciation.

This entire process effectively combines the knowledge based syllabification with the data driven BPE. The syllabification algorithm is tailored for Malayalam script. However S-BPE algorithm can be extended to any language that can be syllabified.

[5] `https://github.com/kavyamanohar/subword-syl-bpe-ml/tree/sbpe`

6.6 Experimental Setup

This section presents the details of our experiments[6]. We begin with a description of the datasets followed by a discussion on building components for hybrid ASR system. We will describe in detail about the segmentation of the text corpus for language modelling, creation of segmented pronunciation lexicons and the process of acoustic modelling.

6.6.1 Datasets

We rely on publicly available open licensed read speech datasets [84, 85] for Malayalam. Every audio recording in the dataset is associated with a textual transcript in Malayalam script. We divide the available speech into train and test datasets, ensuring non-overlapping speakers and speech transcripts. The train datasets described in Table 5.5 are combined to get 1125 minutes of audio for acoustic modelling. The training dataset is same as that of the experiment described in chapter 5. Since the focus of this chapter is on the usage of open vocabulary ASR to recover OOV words, we have not used the test dataset with fewer than 1% OOV words.

Table 6.5: Details of Speech data sets used in our experiments.

Name	Corpus	#Speakers	#Utterances	Duration (minutes)
1	Indic TTS, IITM [84]- Train	2	8601	838
2	Open SLR Malayalam [85] - Train	37	3346	287
T1	Open SLR Malayalam [85] - Test	7	679	48

To create the language model, we use the sentences from the speech transcripts and combine it with the curated collection of text corpus published by SMC [106]. From this, every sentence that appeared in our test dataset is explicitly removed to prevent overfitting. The resulting text corpus contains 205k unique sentences, 1325k word types, and 356k unique word tokens.

6.6.2 Hybrid ASR components

The hybrid ASR decoder consists of three modules as described in Figure 1.1. The functions of these modules are listed below:

1. The acoustic model predicts the posterior likelihood $Pr(X|Q)$ of CD tied phone HMM states $Q = Q_0, Q_1, ...Q_K$ given the acoustic feature vectors $X = X_0, X_1, ..., X_N$. Modern acoustic models are usually implemented with deep neural networks. Deep

[6]The Kaldi Experimental Setup: `https://gitlab.com/kavyamanohar/ml-subword-asr/-/tree/master/`

neural network training relies on the frame level alignment of audio and phoneme labels obtained from a previously trained GMM-HMM acoustic model [156].

2. The pronunciation lexicon maps words into sequence of phonemes. The acoustic model training module would need to look up the pronunciation lexicon to convert the word-level transcripts into phoneme sequences.

3. The language model predicts the conditional likelihood $Pr(w_{i+1}|w_0, w_1, ...w_i)$ of the next word w_{i+1} given the previous words.

All these components are composed into a weighted finite-state transducer framework [157] and the most likely word sequence is retrieved using graph search methods. This word based system would serve as the baseline for our experiments.

Subword based ASR, as shown in Figure 6.1 is very much like word based ASR system, except that (i) the language model represents the conditional probability of subword sequences, instead of words and (ii) the pronunciation lexicon is composed of subword segments. The word boundary marker is chosen so that the predicted subword segments can be easily concatenated to form words. We use the the segmentation methods described in 6.4 and 6.5, and compare them with the standard word-based ASR to answer the research questions. The creation of segmented text corpus for subword language model training and the creation of segmented pronunciation lexicons are explained in the following subsections.

6.6.3 Creating Segmented Text Corpora

This section explains the text corpus segmentation procedure. Morfessor, BPE and Unigram are data driven segmentation algorithms while S-BPE is a hybrid one that additionally relies on linguistic knowledge. We use a subset of the text corpus (7.5k sentences) to train the data driven and hybrid models.

1. Words are separated by spaces in the text corpus and are easily segmented.

2. Using the default settings with Morfessor 2.0 [149] we trained Morfessor on the training set and applied it to create morpheme segmented text corpus.

3. BPE [145] learns the vocabulary from the training dataset. The initial vocabulary is formed by the Malayalam characters in the training dataset. The number of merge operations is set to 10000. This results in a BPE model which is used to obtain the BPE segmented text corpus.

4. Unigram [146] model is trained by the `sentence piece` library using the training dataset with a vocabulary of 15000. The trained Unigram model is used to get the Unigram segmented text corpus.

5. Being a rule based algorithm, syllabifier requires no training. Algorithm 6 is directly applied on the text corpus to obtain syllable segmented corpus.

6. S-BPE model is trained using the Algorithm 7 so that the vocabulary is learnt. The initial vocabulary is formed by the Malayalam syllables present in the training dataset. The number of merge operations is set to 10000. This results in a model which is used to obtain S-BPE segmented text corpus.

Samples of text segmented using these methods are presented in Table 6.6.

Table 6.6: Examples for different segmentation strategies. Space is used as delimiter between segments.

Method	Example	Segment count
Word	അവൻ വള ഇടുകയില്ല /avan ʋaɭa iʈukajilla/	3
Morfessor	അവ+ ൻ വള ഇട+ ുക+ യില്ല /ava+ n ʋaɭa iʈa+ uka+ jilla/	6
BPE	അവൻ വ+ ള ഇടു+ കയ+ ില്ല /avan ʋa+ ɭa iʈu+ kaja + illa/	6
Unigram	അവ+ ൻ വള ഇട+ ുകയില്ല /ava+ n ʋaɭa iʈa+ ukajilla/	5
Syllable	അ+ വൻ വ+ ള ഇ+ ടു+ ക+ യി+ ല്ല /a+ ʋan ʋa+ ɭa i+ ʈu+ ka+ ji+ lla/	9
S-BPE	അവൻ വള ഇടു+ കയില്ല /avan ʋaɭa iʈu+ kajilla/	4

6.6.4 Language modelling

Because highly inflected words can be divided into smaller pieces, segmentation into subwords can lessen the impact of rich morphology [138]. Statistical n-grams serve as a simple and powerful tool to capture language modelling information. The order of n-gram needed to capture this information depends largely on the properties of the segments used. The segmentation strategy determines the properties of the segmented training text - affecting the total number of segments in the text, the number of characters within each segment, the number of segments in a word and the frequency of segments.

For a sentence W formed by sequence of N segments $W = w_1, w_2...w_N$, the probability $Pr(W)$ of the sentence is given by the following formula applying the chain rule of probability.

$$Pr(W) = Pr(w_1, w_2, ...w_N) \tag{6.6.1}$$

$$= Pr(w_1)P(w_2|w_1)...Pr(w_N|w_{N-1}, w_{N-2}...w_1) \tag{6.6.2}$$

Based on the Markovian assumption of n-gram language modelling, probability of each word depends only on the previous $n-1$ words. This makes the sentence probability to be computed as:

$$Pr(W) = \prod_{i=1}^{N} Pr(w_i|w_{i-1}, w_{i-2}..w_{i-(n-1)}) \tag{6.6.3}$$

Perplexity, PPL of a sentence, is the inverse probability normalised by the number of segments, N. Normalisation ensures, longer sentences are not heavily penalised.

$$PPL(W) = \left(\frac{1}{Pr(W)}\right)^{\frac{1}{N}} \tag{6.6.4}$$

Perplexity can be interpreted as the weighted average branching factor of a language. The branching factor of a language is the number of possible next words that can follow any word. Higher perplexity is positively correlated with the difficulty in language modelling [158]. Since the number of segments, N is largely determined by the segmentation method, perplexity measure that is dependent of this parameter can not be used to compare modelling complexity across different segmentation methods [137]. The negative log likelihood (NLL) of the segment probability distribution effectively removes this dependency as described in the following equation.

$$NLL(W) = -log_2\left(Pr(W)\right) \tag{6.6.5}$$

$$= Nlog_2\left(PPL(W)\right) \tag{6.6.6}$$

The metric $NLL(W)$ is called surprisal [138]. By scaling down the $NLL(W)$ by the number of characters, M in a sentence, we can obtain the log character level perplexity, $log\ PPL_c(W)$ as in equation 6.6.7. The number of characters in a sentence is independent of the segmentation method and like surprisal, this metric can be used to compare language modelling complexity across segmentation methods.

$$log\ PPL_c(W) = \frac{N}{M} log_2(PPL(W)) \tag{6.6.7}$$

Alternatively log word level perplexity, $log\ PPL_c(W)$, where the scale factor is the number of words in a sentence can also be used for comparison across segmentation methods.

For language modelling complexity comparisons across segmentation methods, surprisal is employed in [138], $log\ PPL_c(W)$ is used in [159], and $PPL_w(W)$ is used in [36]. To compute the surprisal per sentence (SPS) of a corpus that contains k ($k = 680$) number of sentences, we use the equation 6.6.8.

$$SPS = \frac{1}{k}\sum_{i=1}^{k} NLL(W_i) \tag{6.6.8}$$

In the experiments performed in this work, we report both $PPL(W)$ and SPS for language modelling complexity. Statistical n-gram language modelling is performed on the segmented text corpus. SRILM toolkit is used for the training and evaluation of models [129]. To avoid zero probability assignment to unseen word sequences, the probability weights are redistributed by a process known as smoothing. We use the modified Kneser-Ney smoothing algorithm [27] to create n-gram language models of orders 2 to 6 for every segmentation strategy. The models are trained to predict the next segment based on the previous n-gram context. The SRILM toolkit can evaluate the test dataset and return the log-likelihood values with respect to base 10 logarithms and the perplexity. Surprisal values are computed by converting these values to base 2 logarithms.

6.6.5 Creating Segmented Lexicons

Graphemic lexicon describes the pronunciation using the language's native alphabets, or graphemes. Since BPE, Unigram and Morfessor segmentation algorithms in our experiments do not have access to pronunciation information, the segmentation can happen at locations that break the pronunciation flow. So, it was decided to use a graphemic pronunciation lexicon, instead of a phonemic one [36] for all the segmentation methods to ensure fair comparison.

Algorithm 8 Subword Lexicon from Word Lexicon

Require: PL_{word}

procedure Create-Subword-Lexicon(PL_{word})
    `subwords` $= \{\}$ ▷ Define an empty list
    for $word$ in PL_{word} **do**
        `subwords` $+=$ **Get-Subwords**($word$) ▷ Expand list
    end for
    `vocabulary` $\leftarrow$ **Unique**(`subwords`) ▷ List of unique subwords
    $PL_{subword} \leftarrow$ **Generate-Pronunciation**(`vocabulary`)
    return $PL_{subword}$
end procedure

For the baseline ASR, the word pronunciation lexicon is prepared by using all the words in

the text corpus with at least three occurrences. It is then expanded to include all the words in the training speech transcript. This word lexicon is referred to as PL_{word} and has 79947 entries. Subword lexicons are obtained by segmenting every word entry in PL_{word} as per the segmentation strategy under consideration and choosing the list of unique segments as described in Algorithm 8. The number of entries in these lexicons are described in Table 6.7.

Table 6.7: Lexicon sizes of different segmentation

Segmentation	Lexicon Size
Word	79947
Morfessor	10545
BPE	9986
Unigram	19564
Syllable	6279
S-BPE	15926

6.6.6 Acoustic modelling

The speech sampling rates of the two train speech corpora are converted to 16kHz prior to feature extraction. During acoustic model training the pronunciation of every word present in the transcript is looked up in a graphemic pronunciation lexicon.

We use the Kaldi chain acoustic model in our experiments. It is based on a TDNN [23]. It is trained using frame-level phoneme state labeling obtained from GMM-HMM speaker adaptive triphone acoustic model. State labels are used as targets to train the TDNN acoustic models. Acoustic features used in TDNN training are: (i) 40-dimensional high-resolution MFCCs extracted from frames of 25 ms length and 10 ms shift and (ii) 100-dimensional i-vectors [19] computed from chunks of 150 consecutive frames. Three consecutive MFCC vectors and the i-vector corresponding to a chunk are concatenated, obtaining a 220-dimensional feature vector for a frame. This acoustic model is trained on a single NVIDIA Tesla T4 GPU.

6.6.7 Summary of Experimental Investigations

The acoustic models are built and combined with language models and pronunciation lexicons using Kaldi toolkit [3]. From six ways segmented text corpus, we construct language models with n-gram orders of 2 to 6. The language modelling effectiveness is then measured using corpus level and information theoretic metrics. Keeping the Kaldi based TDNN acoustic model fixed across segmentation methods, we use segmented lexicons and corresponding language models to create 30 (1 acoustic model × 6 segmented lexicons × 5 n-gram orders) different ASR decoders. These decoders are then tested on a

multispeaker test dataset described in Table 5.5.

6.7 Results

In this section, we present the findings from our experiments. We first perform a corpus linguistic analysis on the segmented corpora. After that, we analyse the language model. This is done in terms of the metric surprisal, which roughly indicates the length, complexity, and overall difficulty that the model has in predicting sentences [137]. Finally, we analyse the ASR results. The WER, OOV-WER, lexicon size, and overall model size are used to measure this.

6.7.1 Corpus Linguistic Analysis

Words can be broken down into smaller pieces that are likely to convey similar meanings in different contexts by segmenting them into subwords, which can lessen the impact of rich morphology. The examples of segmentation done by different segmentation method given in Table 6.6 indicates how the number of segments per sentence varies with the method of segmentation. It can also be observed from the table that the mean segment length (MSL) also depends on the segmentation method. The evaluation of these parameters over the speech transcripts in the test corpus is described next.

Linguistic Validity of Segments

The segments given by different methods, as exemplified in Table 6.6, does not necessarily comply with linguistic correctness. The word segments are orthographically and phonetically valid linguistic units. The segmentation given by Morfessor tool are not true morpheme segments. The Morfessor segments break the orthographic flow as in ഇടുക /iṭuka/ being segmented as ഇട+ ുക /iṭa+ uka/. In the second segment, the vowel sign ു, occurs without a consonant preceding it, which is an invalid orthographic usage. Similar invalid orthographic usages can be observed in BPE and Unigram segmentation methods too.

Syllable segmentation method, by its design, always gives orthographically valid subword units. S-BPE method also gives orthographically valid subword units, which are longer than syllable segments. But none of the methods are capable of providing linguistically meaningful subword segments. However, unlike machine translation applications, this is not an essential requirement for building an ASR system.

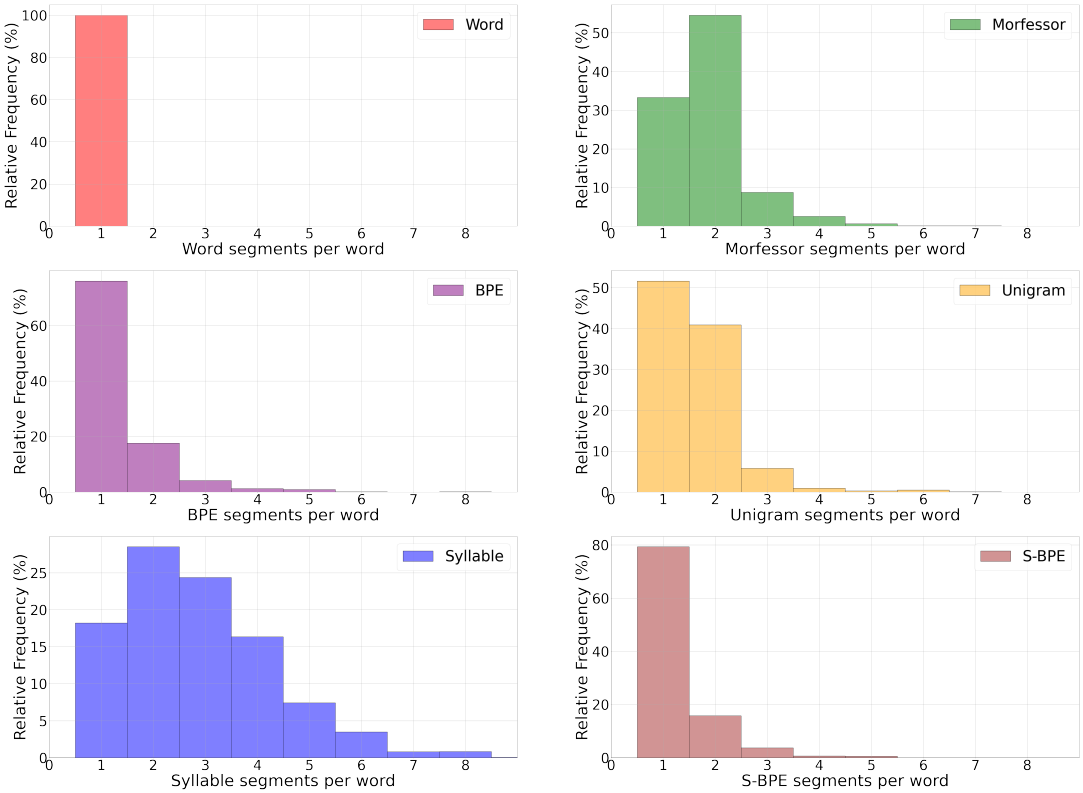


Figure 6.2: Distribution of the number of subword segments per word

Segment Count per Word and per Sentence

The distribution of number of segments per word in the test dataset is illustrated in Fig. 6.2. Word segmentation does not beak down the words, resulting in a single bar graph. In BPE, Unigram and S-BPE segmentation methods, more than 50% of the words remain unsegmented, followed by words being segmented into two subwords. In Morfessor segmentation, the distribution shows more than half the words are segmented into two, followed by words remaining unsegmented. The percentage of words that get segmented into more than two segments are rare in all these methods. However in syllable segmentation, about 28% and 24% of words get segmented into two and three subwords respectively. The segment length per word is more broadly distributed in syllable segmentation.

Table 6.8: Sentence length statistics in terms of the number of segments per sentence.

Segmentation	Minimum	Maximum	Mean
Word	5	14	6.4
BPE	5	26	8.5
Morfessor	6	29	11.7
Unigram	5	29	10.1
Syllable	8	49	19.9
S-BPE	5	25	8.1

On analysing the segmentation statistics over sentences, we get the values reported in Ta-

ble 6.8. It describes the minimum, maximum and mean number of segments per sentence. Syllable segmented sentences contain on an average 19.9 subword segments, which is the highest count of all. Sentences that contain large number of segments would need longer n-gram language model context to guide the decoding [36]. We will analyse its impact on ASR later in section 6.8.

Mean Segment Length

The MSL, defined as the average number of characters in a segment, depends on the segmentation strategy. The distribution of segment lengths, in the form of box plots is shown in Figure 6.3. The glsmsl is the highest for words (8.3) as expected and the smallest is for syllables (2.2). The MSL values of other segmentation methods are 3.9, 4.3, 4.5 and 4.8 respectively for Morfessor, Unigram, BPE and S-BPE.

Comparatively smaller box for syllable, indicates the length is distributed closely about the median value, with very little outliers. However for word segmentation, the length of box plot is larger, indicating the segment lengths vary widely.

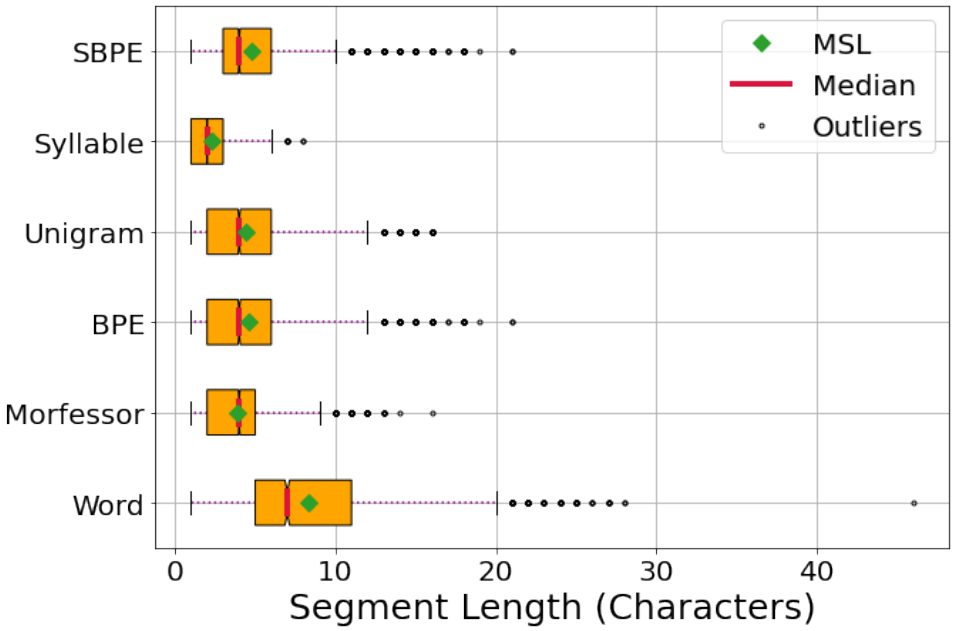


Figure 6.3: Segment Length distribution in the datasets

6.7.2 Language modelling Complexity

The complexity of a language model is related to its difficulty in determining the next segment from the previous n-gram context. The higher order n-grams extract more context for the occurrence of a segment and generally reduces language modelling complexity and hence perplexity. However raising the n-gram order beyond a limit reintroduces the data sparsity problem, resulting in unimproved perplexity and SPS values [159]. Subword

language models require higher order n-grams to capture the context, than word based ones [80]. In our experiments we create language models of orders $n-gram = 2$ to 6 and analyse their complexity in terms of perplexity and SPS.

Table 6.9: Language modelling Complexity in terms of Perplexity

n-gram	Word	BPE	Morf.	Uni.	Syl.	S-BPE
	Perplexity					
2	2266	334	162	287	44	387
3	2188	245	111	211	19	308
4	2187	242	107	208	16	307
5	2188	241	107	207	16	307
6	2188	242	107	207	16	307

Table 6.9 presents the values of perplexity over different segmentation and n-gram orders. For any given n-gram order, it can be observed from this table that the perplexity value follows the pattern: Word $>$ S-BPE $>$ BPE $>$ Unigram $>$ Morfessor $>$ Syllable. This is inversely correlated with the mean number of segments per sentence, N, tabulated in Table 6.8. Lower the value of N, higher is the perplexity. However the perplexities are not comparable across different segmentation methods [160]. But for a specific segmentation, the perplexity can be compared along the variations in n-gram orders. So it can be concluded that the fourth order syllable level language model with lower perplexity is an improved model than the corresponding second order language model with higher perplexity. But it does not necessarily imply the fourth order syllable level language model with lower perplexity is better than fourth order word level language model with higher perplexity.

Table 6.10: Language Modelling Complexity in terms of SPS

n-gram	Word	BPE	Morf.	Uni.	Syl.	S-BPE
	SPS					
2	70	98	99	99	124	96
3	69	93	92	94	98	92
4	69	93	91	93	93	92
5	69	93	91	93	92	92
6	69	93	91	93	92	92

The SPS values obtained in our experiments are shown in Table 6.10. As expected, the SPS reduces initially and then stabilises with increase in n-gram order for every segmentation method. However, it is interesting to note that the best set of SPS values are obtained for word segment based language model. Syllable segments of lower n-gram orders show higher SPS values than all other segmentation methods. For the test corpus under evaluation, all subword segment based language models exhibit a comparable SPS value of ≈ 92, at higher n-gram orders ($n \geq 4$). To conclude, word level language model is better

off than all the segmented language models in terms of SPS. This pattern is previously observed in [36], where the complexity is computed in terms of $PPL_w(S)$.

The impact of subword segment based language modelling on ASR decoder needs to be evaluated in terms of its ability to recover OOV words and corresponding reduction in WER, which is attempted in the following section. However, lowering the language model complexity does not always ensure an improvement in automatic speech recognition accuracy [36, 161].

6.7.3 Evaluation of ASR performance

To begin with, we present the ASR error rate which is computed as WER. It is based on the number of words inserted, deleted and substituted in the predicted speech transcript when compared to the ground truth transcript according to (6.7.1).

$$WER = \frac{(Insertions + Deletions + Substitutions) \times 100}{(Number\ of\ words)} \tag{6.7.1}$$

The evaluation is performed on a multi-speaker studio recorded dataset from the same domain as the train dataset. About 14% of words in this test dataset are OOV words, which can not be recovered by word based ASR. According to [162], it has been shown that the presence of an OOV word in test set can result in approximately two errors during ASR decoding.

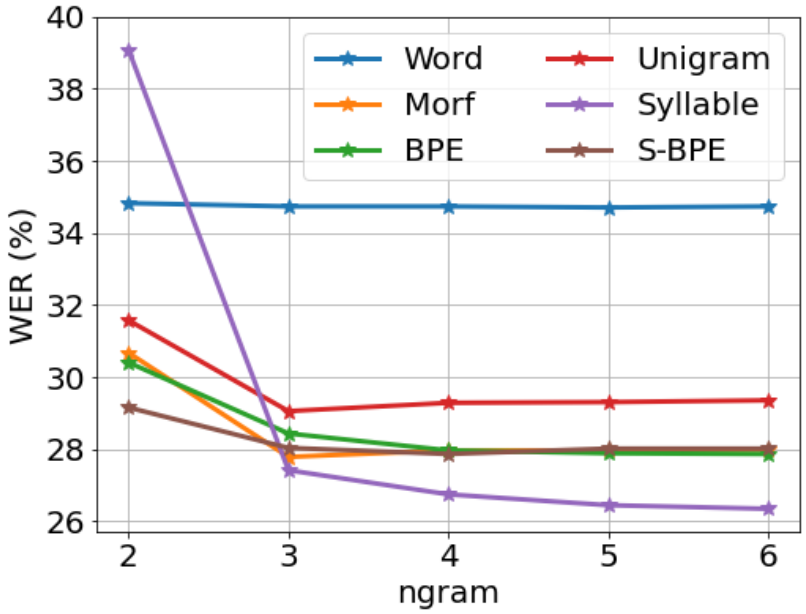


Figure 6.4: ASR performance evaluation in terms of the WER (lower is better)

The WER of different segmentation methods is shown in Fig. 6.4. The baseline word

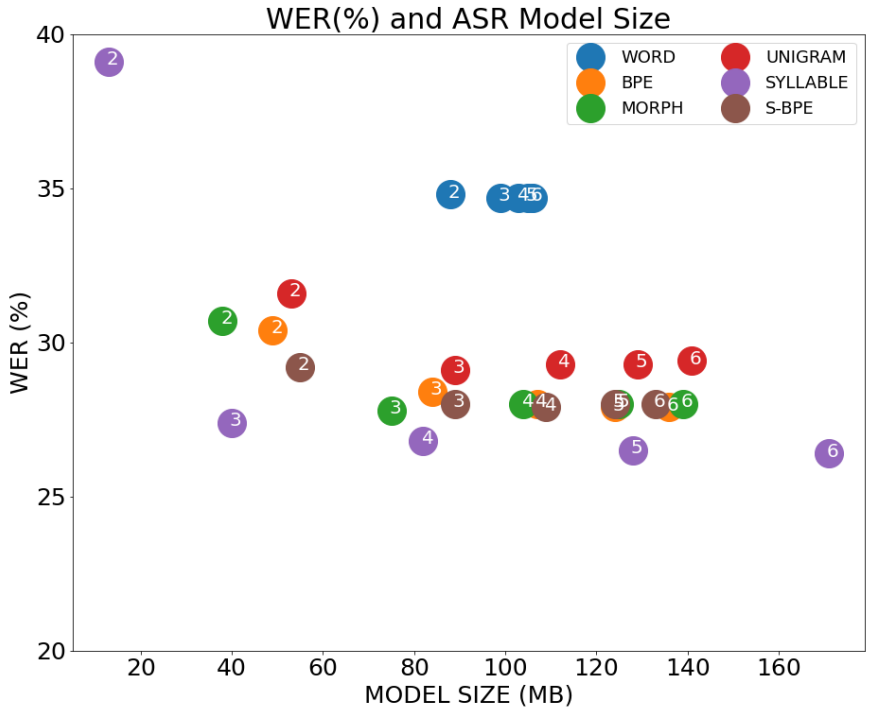


Figure 6.5: Trade-off between ASR performance and Model Memory Requirement (n-gram order is shown as labels)

model with a lexicon size of 79k has a word error rate (WER) of $\approx 35\%$ for all n-gram orders. The performance of all subword models is better than the baseline, except for bigram syllable language model. However, for n-gram orders 3 and above, syllables demonstrate an improvement in WER compared to all other segmentation methods and n-gram orders. Syllables with the smallest MSL among all segmentation methods, require more n-gram context to make a reasonable prediction about the next subword segment. But once it has enough context, the predictions become more reliable as exemplified in the WER reduction in Fig. 6.4.

Syllable is the only segmentation method that show consistent improvement in WER, the best being 26.4%, at the n-gram order $n = 6$. But the relative improvement diminishes with the n-gram order. For all other segmentation methods, the WER does not show any improvement beyond n-gram$= 4$. The performance of S-BPE, BPE and Morfessor remains closer at about 28% WER, for n-gram order of 4 and beyond. The best WER provided by Unigram segmentation is 29%.

WER of OOV Words

To study the performance of subword based ASR compared to the baseline word model on OOV recovery, we compute the WER, specifically for OOV words. The OOV words in

the test set are determined with respect to the word level lexicon. The results are tabulated in Table 6.11.

Table 6.11: WER on OOV words

Method	OOV-WER(%)
Word	100.0
Morfessor	55.3
BPE	53.1
Unigram	55.0
Syllable	49.9
S-BPE	50.2

Since OOV words can not be recovered by a word segment based ASR, The OOV-WER is 100% for the baseline word based ASR. To analyse the extend of OOV-WER in subword segmented ASR, we use the `texterrors` Python library [163]. The OOV-WER values are computed corresponding to the best WER values reported in section 6.7.3. Providing the list of OOV words in the test set along with the true speech transcripts, this library computes the OOV-WER of the subword ASR model. The syllable segments perform the best, which has reduced the OOV-WER to 49.9%, closely followed by S-BPE segments. Purely data driven segmentation techniques of BPE, Morfessor and Unigram could also recover the OOV words, but the WER are not as promising as the Syllable and S-BPE segments.

Trade-off between WER and Model Memory Requirement

The order of n-gram impacts the memory requirement of the ASR model. To study the the model memory requirement, we compute the size of weighted FST graph (*HCLG.fst*) used for decoding. In the trade-off diagram shown in Fig. 6.5, the model size of the word based baseline ASR model does not change significantly with the n-gram order. However, the error rate of the word-based baseline model is higher than all subword-based models, except for the syllable bigram ASR. Although the syllable bigram ASR has the smallest model size, its error rate is so high that it is not practical to use and is therefore not considered in further analysis.

For lower-order language models ($n \leq 4$), the subword-segmented ASR models have significantly smaller sizes compared to the word-based models, while also having a lower error rate. Syllable segments require less memory at smaller n-gram orders, but the model size increases quickly with the increase in n-gram order, while the corresponding reduction in error rate is relatively small. From the figures 6.4 and 6.5, it can be seen that the 4-gram syllable language model, which has a better error rate than all other subword-segmented models, has the smallest model size.

6.7.4 Comparison with Other Reported works

Since the previously published work on subword ASR for Malayalam [82], was tested on a private dataset, the comparison of results is not meaningful and hence not attempted. The current study is a detailed expansion of the experiments reported in [164], with the difference that graphemic lexicons are used instead of phonemic ones, focusing on a medium OOV test dataset where the WER improvement is notable and within a reasonable rage.

6.8 Findings and Discussions

In this section we analyse the experimental results to draw the relationship between segmentation methods and language modelling and their impact on ASR.

6.8.1 Language modelling Complexity

To compare language modelling performance across different segmentation methods, we use the SPS metric [160]. The major observations are listed below:

1. For all segmentation methods, language modelling complexity decreases with n-gram order initially, and then stabilises.
2. Raising the n-gram order beyond a limit (different for different segmentation methods) reintroduces the data sparsity problem, resulting in unimproved perplexity values [159]. This limit is 3 for words, BPE and S-BPE, 4 for Morfessor and Unigram and 5 for syllables as presented in Table 6.10.
3. In terms of SPS, the language modelling complexity is the least for words than all other segmentation methods.
4. The lower LM complexity in terms of SPS does not guarantee improved WER [36].

6.8.2 ASR results

Syllable segments give the best WER among all other segmentation strategies, and it happens for n-gram orders 3 and above. It could recover more than 50% of the OOV words correctly. This performance improvement can not unambiguously be attributed to the segmentation happening at valid orthographic boundaries. The proposed S-BPE segmentation did not bring in much reduction in WER, and has a performance similar to that of BPE and Morfessor segmentation methods and better than word and Unigram segmentation.

The WER result obtained for Malayalam ASR is not correlated with the language modelling complexity of the segmentation method. This pattern was previously observed and

reported in [36] for Finnish, Swedish and Arabic ASR where word segments show least language modelling complexity but the highest WER.

The subword lexicons used in our experiments were derived from the word lexicon used in our baseline experiments by the procedure explained in section 6.6.5. In our experiments, we compared various segmentation methods in terms of their lexicon size and WER. The Unigram segmentation method produced the largest subword lexicon, with approximately $19k$ entries, but it also yielded the highest WER of 29%. On the other hand, the BPE, S-BPE and Morfessor segmentation methods resulted in lexicons with around $10k$ to $15k$ entries and a similar WER of approximately 28%. However, the syllable segmentation method with the smallest lexicon size of about $6k$ entries had the best WER of 26%. Overall, our results suggest that smaller lexicon sizes can result in better WERs, when the subword lexicons are derived from a common pool of words.

An optimal choice of n-gram order helps to build ASR models for memory constrained environments while maintaining a good recognition accuracy. Syllable segments with n-gram orders 3 and 4 prove to provide best WER at the lowest reported model memory requirement.

6.9 Summary

The study presented in this chapter is the first investigation on the diverse aspects of improving speech recognition system in the morphologically complex Malayalam language using subword language modelling techniques. We have presented the detailed investigation of the usage of different subword segmentation strategies to build statistical n-gram language models for usage in hybrid ASR task. The language modelling complexity is evaluated using perplexity and surprisal. We build graphemic pronunciation lexicons of these subword segments. The ASR decoders built by combining these language models and lexicons with a common acoustic model are used to evaluate a multi speaker Malayalam speech dataset having 14% of OOV words with respect to the baseline word pronunciation lexicon. The ASR performance is evaluated based on the WER, OOV-WER and the model memory requirement.

It has been demonstrated that the linguistically informed syllable segments perform exceptionally well with a 26% WER, which is an 8% improvement from the baseline word level ASR. The hybrid segmentation strategy proposed in this paper, S-BPE reports a WER of 28%, but it could not beat the syllable subword segments in terms of WER.

To conclude, the syllable subword segments have the advantage of requiring less memory for the model, especially at n-gram orders of 3 and 4, while still maintaining a low error rate. Syllable subwords also have the lowest error rate for out of vocabulary words,

showing that they are effective at correctly identifying about half of these words. Overall, syllable segmentation provides benefits in terms of reduced model memory requirements and a lower error rate, despite having higher language modelling complexity compared to word-based language models.

Chapter 7

Conclusion

In this thesis we have demonstrated that the morphological complexity of Malayalam is much higher than that of many other Indian and European languages which are known for their complex morphological structure. Considering this morphological complexity, a general purpose ASR would require a large PL. We designed and developed an FST based bidirectional grapheme to phoneme conversion tool Mlphon, which can additionally perform syllabification as well. The speed and accuracy of Mlphon is evaluated and compared with other openly available lexicon creation tools.

Then we have presented the building process of LVCSR system for Malayalam. The resultant model is evaluated in terms of WER. The data used for building the model, model creation scripts, lexicons and the resultant model are made publicly available under open license for reproduction and reuse purposes.

We have also presented an alternate approach to address the morphological complexity of Malayalam in ASR task by building open vocabulary ASR. Existing subword segmentation strategies are compared with two proposed algorithms for subword modelling in Malayalam and the resulting WER are evaluated.

7.1 Major Contributions

1. Analysis of the morphological complexity of Malayalam language.
2. Documentation of the graphemic and phonemic inventory of Malayalam and the correspondence between the two.
3. Development of an algorithmic description of the grapheme-phoneme correspondence in Malayalam and implement it into a modular toolkit.
4. Publication of large vocabulary pronunciation lexicon of more than hundred thousand words belonging to different word categories.
5. Development of a LVCSR model for Malayalam and publication of an open licensed Malayalam ASR model that could be integrated to various applications.
6. Exploration of subword segmentation strategies suited for Malayalam considering its morphological complexity.

7. Development of subword based open vocabulary ASR model to reduce WER and model memory requirement.

Furthermore, open dissemination of the source codes and the models is essential to ensure reproducibility, reusability and research continuity. This aspect of has been given utmost priority at every stage of this research work.

7.2 Limitations

Many large speech corpora in Malayalam published at the time of the writing of this thesis are documented in the literature review. However they were not utilised in the experiments of the research work, which were completed before the publication of those resources.

The experiments in the current research uses the best approach in hybrid DNN-HMM acoustic modelling, using TDNN with long temporal context and MMI based sequence discriminative training [23–26]. Over the past few years, the field of ASR has undergone significant changes, particularly with the introduction of pretrained speech transformers trained on speech-only data [37] or combined with annotated speech data [38] from multiple languages, which can be fine-tuned for ASR tasks using small annotated speech datasets. While the use of such models has not been extensively explored in the low-resource ASR task of this thesis, they might have the potential to significantly improve the performance of ASR systems.

7.3 Future Scope

The Mlphon G2P tool developed in this study doesn't currently offer transcriptions at the allophone level that indicate co-articulation effects. However, there's potential to expand the tool to include these effects in the future.

As a future avenue of research, it would be valuable to explore methodologies that effectively isolate the impact of varying phonetic lexicons in acoustic models from the presence of language models. This could facilitate a more refined assessment of phoneme recognition efficiency within diverse acoustic models when using distinct pronunciation lexicons. Furthermore, investigating techniques to disentangle these influences could yield insights into the interplay of pronunciation and language modeling in speech recognition systems.

To enhance the open framework for building ASR introduced in this thesis, it would be beneficial to increase the speech dataset used for acoustic modeling. With the availability of new speech datasets for Malayalam, an improved model could be integrated into desktop or mobile applications for speech-based typing and interacting with conversational agents.

The subword modeling techniques proposed in this research for language modeling could also find application in E2E ASR systems for acoustic modeling. However, further investigation is needed to explore this possibility.

Appendix I

Characteristics of Malayalam Orthography

Malayalam is a language spoken predominantly in the state of Kerala in southern India, with about 38 million native speakers. It belongs to the Dravidian language family, and has an alphasyllabary writing system [112]. Though Malayalam script is largely phonemic in nature, there are some unique characteristics like: (i) consonants with and without inherent vowel, (ii) consonant clusters with pronunciation different from the consonants present in them, (iii) special symbol *virama*, that contextually chooses its function depending on its position in a word and (iv) graphemes being overloaded with non-native sounds in loan words.

I.1 Grapheme Inventory of Malayalam

Malayalam belongs to the family of Brahmic writing systems that is alphasyllabary in nature [112]. In this writing system, consonant - vowel sequences are written as a unit; each unit is based on a consonant character, and the vowel notation is secondary. The basic components in Malayalam orthography belong to three classes of characters: namely vowels, consonants and signs. Additionally there are complex graphemes derived from these basic characters.

I.1.1 Vowels

There are 16 vowels in Malayalam. It includes 5 short vowels, 5 long vowels, 2 diphthongs and 4 vocalics [118]. Independent vowels occur only at word beginnings. Vowels that follow consonants in word medial or end positions are indicated by dependent vowel signs. Consonants generally have the vowel /a/ inherent in them, eliminating the need for specialized vowel sign for അ /a/. See Table I.1 for the list of all vowels in Malayalam and their IPA representations.

I.1.2 Consonants

There are 38 regular consonant graphemes in Malayalam [118]. This includes 21 plosives classified by their aspirational and voicing characteristics, 6 nasals, 4 fricatives, 3

Table I.1: Short and long vowels in Malayalam and their IPA representations

Vowels	അ a	ആ aː	ഇ i	ഈ iː	ഉ u	ഊ uː	ഋ rɨ	ൠ rɨː	ഌ lɨ	ൡ lɨː	എ e	ഏ eː	ഐ ai	ഒ o	ഓ oː	ഔ au
Vowel Signs		ാ	ി	ീ	ു	ൂ	ൃ	ൄ	ൢ	ൣ	െ	േ	ൈ	ൊ	ോ	ൗ

Table I.2: Consonants in Malayalam and their IPA representations

Place of Articulation	**Manner of Articulation**									
	Plosive[a]	Plosive[b]	Plosive[c]	Plosive[d]	Nasal	Trill	Tap	Fricative	Approximant	Lateral
Velar	ക ka	ഖ kʰa	ഗ ga	ഘ gʰa	ങ ŋa					
Palatal	ച ca	ഛ cʰa	ജ ɟa	ഝ ɟʰa	ഞ ɲa			ശ ɕa	യ ja	
Retroflex	ട ʈa	ഠ ʈʰa	ഡ ɖa	ഢ ɖʰa	ണ ɳa			ഷ ʂa	ഴ ɻa	ള ɭa
Alveolar	ഩ t̺a				ന na	റ ra	ര ɾa	സ sa		ല la
Dental	ത t̪a	ഥ t̪ʰa	ദ d̪a	ധ d̪ʰa	ന n̪a					
Labial	പ pa	ഫ pʰa	ബ ba	ഭ bʰa	മ ma					
Labiodental									വ ʋa	
Glottal								ഹ ha		

[a] Unaspirated and Unvoiced
[b] Aspirated and Unvoiced
[c] Unapirated and Voiced
[d] Aspirated and Voiced

approximants, 2 laterals and 1 each tap and trill. The place of articulation are indicated in the rows and manner of articulation in the columns of the Table I.2. Apart from the regular consonants, there are dead consonants (referred as *chillus*) in Malayalam, which do not have the inherent vowel associated with them. The *chillus* of Malayalam are listed in Table I.3.

Table I.3: *Chillus* in Malayalam and their IPA representations along with the base consonants from which *chillus* were derived.

Chillus	**Base consonants**
ക് k	ക ka
ൺ ɳ	ണ ɳa
ൻ n	ന na
ൽ l	ല la
മ് m	മ ma
യ് j	യ ja
ൾ ɭ	ള ɭa
ർ ɾ	ര r

I.1.3 Signs

The special signs *virama* (്), *dot reph* (ഁ), *anuswara* (ം) and *visarga* (ഃ) have their properties as tabulated in Table I.4. *Virama* removes the inherent vowel from the consonant preceding it. The virama that occurs at word ends, apart from removing the inherent vowel, adds the mid-central vowel schwa /ə/ to native Malayalam words. *Dot reph* is an alternate sign representation for the consonant clusters that begin with /r/ or /ɾ/. *Anuswara* is a sign common in Malayalam. Its phonemic representation is /m/ and always mark syllable endings. *Visarga* sign is popular in Sanskrit derived words and they introduce slight pronunciation changes similar to aspirated glottal stop.

Table I.4: Signs in Malayalam

Sign	**Properties**
Anuswara (ം)	Represents /m/ at syllable ends.
Dot reph (ഁ)	Represents /r/ or /ɾ/.
Visarga (ഃ)	Introduces aspirated glottal stop.
Virama (്)	Kills Inherent vowel.
	Inserts schwa at word ends.

I.1.4 Complex Graphemes in Malayalam

Apart from the basic characters, Malayalam script has hundreds of complex graphemes representing consonant clusters. A consonant cluster is a sequence of consonants with no intervening vowels. The removal of inherent vowel from the conjoining consonants

happens on the addition of a *virama* sign. A consonant cluster, often forms a complex grapheme with one or more of stacking, changing and merging the shapes of the constituent characters. Hundreds of possible complex graphemes in Malayalam are not individually encoded in Unicode, instead they are constituted from basic characters. Table I.5 lists certain examples of consonant clusters in Malayalam and their constituents.

Table I.5: Examples of consonant clusters in Malayalam and their constituents

Consonant cluster	Constituent character sequence
ക്ക kka	ക ka ◌് ക ka
ങ്ക ŋka	ങ ŋa ◌് ക ka
ക്ല kl̪a	ക ka ◌് ല la
ഘ്ന gʱna	ഘ gʱa ◌് ന n̪a
സ്ത sṯa	സ sa ◌് ത ṯa
ഗ്ര gɾa	ഗ ga ◌് ര ɾa
ഗ്യ gja	ഗ ga ◌് യ ja
ന്ത്ര n̪ṯra	ന n̪a ◌് ത ṯa ◌് ര ɾa

I.2 Phoneme Inventory of Malayalam

The regular vowel phonemes in Malayalam are listed in Table I.6, classified with their vowel height, length and backness. The graphemic origin of these vowel phonemes can be from independent vowels or dependent vowel signs. The mid-central vowel, schwa (/ə/) does not have a specific vowel grapheme. Its occurrence is limited to words that end in *virama*. Discussion on schwa addition at word ends is in Section I.3.4. Additionally there are two diphthongs (/ai/, /au/) and four vocalics (/rɨ/, /rɨː/, /lɨ/, /lɨː/) in Malayalam which belongs to the category of vowels. Vocalics are letters derived from Sanskrit that generally behave like vowels[1] [118].

Table I.6: Vowel phonemes of Malayalam

		Backness				
		Front		Central	Back	
		Short	Long	Short	Short	Long
Height	Close	i	iː		u	uː
	Close-mid	e	eː		o	oː
	Mid			ə		
	Open	a	aː			

There are 39 consonant phonemes in Malayalam. Their classification based on the manner and place of articulation is listed in Table I.7. The plosive phonemes in Malayalam

[1]Script notes on Malayalam by Richard Ishida: `https://r12a.github.io/scripts/malayalam/`

have the features of aspiration and voicing. Native Malayalam words do not have the phoneme labiodental fricative, /fa/. But a lot of foreign language words are written by overloading the labial aspirated plosive grapheme ഫ /pʰa/ with the /fa/ sound. This makes the consonant phoneme inventory larger than the consonant grapheme inventory by one. Disambiguating the pronunciations of ഫ, primarily involves identifying if it is a foreign language word from the grapheme context. The correspondence of the phonemes with the graphemes in Malayalam script and the rules of exceptions are discussed in Section I.3

Table I.7: Consonant Phonemes of Malayalam

		Manner of Articulation									
		Plosive[1]	Plosive[2]	Plosive[3]	Plosive[4]	Nasal	Trill	Tap	Fricative	Approximant	Lateral
Place of Articulation	Velar	k	kʰ	g	gʰ	ŋ					
	Palatal	c	cʰ	ɟ	ɟʰ	ɲ			ɕ	j	
	Retroflex	ʈ	ʈʰ	ɖ	ɖʰ	ɳ			ʂ	ɻ	ɭ
	Alveolar	t̪				n	r	ɾ	s		l
	Dental	t̪	t̪ʰ	d̪	d̪ʰ	n̪					
	Labial	p	pʰ	b	bʰ	m					
	Labiodental								f	ʋ	
	Glottal								h		

[1]Unaspirated and Unvoiced
[2]Aspirated and Unvoiced
[3]Unaspirated and Voiced
[4]Aspirated and Voiced

I.3 Grapheme to Phoneme Correspondence

This section discusses the general rules of grapheme-phoneme conversions in Malayalam and the exceptional cases which are to be handled in an automatic tool for doing the same using FSTs. The correspondence between graphemes and phonemes in Malayalam is not strictly one-to-one. The conversions from graphemes to phonemes and vice versa is important in the context of ASR, TTS synthesis, phonemic transliterations etc. Mlphon, in its current form, is designed as a tool for phoneme level analysis and thus the allophonic variations (eg: voicing of word medial plosives) due to co-articulation effects are not considered in this work. A potential extension for Mlphon could involve incorporating allophone-level representations which would be a prospective direction for further development.

I.3.1 Vowels

Independent vowel graphemes and dependent vowel signs are always mapped to the corresponding IPAs listed as in Table I.1. The mid-central vowel schwa which does not have an explicit vowel grapheme is mapped to *virama* at word ends.

I.3.2 Base Consonants

Primarily all the consonant graphemes can be mapped to the corresponding IPAs as listed in Table I.2. The syllabic nature of alphabet makes all consonant graphemes to have the inherent vowel, /a/, associated with them, unless followed by a dependent vowel sign or a *virama*. Phonemes corresponding to the vowel signs replace the inherent vowel when vowel signs follow a consonant. *Virama* acts as inherent vowel killer, except when it occurs at word end positions.

Malayalam has graphemes for alveolar plosive, ഩ /t̪a/, and alveolar nasal, ന /na/, but are not in popular use. The dental nasal grapheme ന /n̪a/ of Malayalam is overloaded to represent the alveolar nasal sound. So the tool for grapheme to phoneme conversion must disambiguate whether the grapheme ന, represents the dental sound or the alveolar sound. This can be done by using contextual rules for native words. But the rules can not be generalized to foreign language words, complex morpheme boundaries etc. Alveolar plosive sound occur only in the context of two consonant clusters in Malayalam and it is discussed in Section I.3.3.

I.3.3 Consonant Clusters

Regular consonant clusters have base consonants separated by *virama* in between. The inherent vowel sound of the consonant preceding *virama* has to be removed for pronunciation modeling as a sequence of phonemes. For example the sequence ങ /ŋa/, ്(virama), ക /ka/, gives the consonant cluster ങ്ക /ŋka/, where *virama* removes the inherent vowel /a/ after /ŋ/. The consonant clusters whose pronunciation differ from the constituent consonants are discussed in Sections I.3.3 and I.3.3.

Exceptional Clusters of Alveolar Nasal and Alveolar Plosive (ണ്ട, ന്റ)

Alveolar plosive always occur in the language either as a geminate or in the consonant cluster ന്റ /nt̪a/. The geminate consonant /t̪t̪a/ is formed not from the consonant grapheme ഩ /t̪a/, but from alveolar trill grapheme ര /ra/. ie., the sequence ര /ra/, *virama* (്), ര /ra/ constitutes the cluster ട്ട /t̪t̪a/.

A very popular consonant cluster involving alveolar phonemes in Malayalam is ന്റ /n□a/. This consonant cluster is derived not from the sequence ന /na/, *virama* (്), ഩ /t̪a/, but

from ന /n̪a/, *virama* (്), റ /ra/. It is to be noted that, there is an alternative representation for ന്റ which involves chillu ൻ /n/ instead of ന /n̪a/, which is also supported by Mlphon.

Multiple Pronunciations of *Reph* Sign (ര്)

When the final consonant in a consonant cluster is the alveolar tap ര /ɾa/, it forms a consonant diacritic, called *reph* sign, usually pre based to the left of the rest of the consonant cluster or forms a new shape. The sequence ഗ /ga/, ് (*virama*), ര /ɾa/ forms the new shape ഗ്ര /gɾa/. The pronunciation of the alveolar tap ര /ɾa/, changes to alveolar trill റ /ra/, depending on the immediately preceding consonant. It is pronounced as /r/ in ക്രമം /kramam/ (*order*) and സ്ത്രീ /sṯriː/(*woman*) but as /ɾ/ in ഗ്രാമം /gɾaːmam/ (*village*). This pronunciation variation is supported by Mlphon.

I.3.4 Multiple Functions of *virama*

Virama acts as inherent vowel killer when used in between consonants in consonant clusters. It can occur at syllable final position, only at the word ends. In that position, apart from removing the inherent vowel /a/, *virama* adds the mid-central vowel schwa /ə/ at word ends as in പാല് /paːlə/. It is called *samvruthokaram*. An alternate graphemic representation for *samvruthokaram* is ു /u/ dependent vowel sign followed by *virama* at word ends. For example അവന് and അവൻ് are two alternate ways to write /ɐvanə/ (*to him*). Both of these representations are supported by Mlphon. It is to be noted that the schwa at the word end distinguishes it from the word അവൻ /ɐvan/ (*he*). Usage of *virama* at other contexts is considered to be linguistically invalid.

I.3.5 Syllable Final Consonants (Chillu, Anusvara, Visarga)

Chillus are special consonant graphemes that do not have inherent vowel associated with it. If a word final consonant sound has to be terminated without a samvruthokaram, a chillu is used as in പാൽ /pal/ (*milk*). Not all consonants in Malayalam has a chillu form. Even though there are 39 consonant phonemes in Malayalam, there are only 9 chillu graphemes as listed in the Table I.3. Chillus also appear in consonant clusters in word medial positions as ർ occurs in വർഗ്ഗം /ɐarggam/ (*class*).

Anusvara is a sign mark, that has the pronunciation of the consonant /m/, without an inherent vowel. മരം /maɾam/ (*tree*) is a word that ends in anusvara. Visarga is used in Sanskrit derived words. It indicates glottal aspiration as in ദുഃഖം /d̪uɦkʰam/ (*grief*). These three characters (Chillu, Anusvara, Visarga) indicate the end of an orthographic syllable. The orthographic syllable structure of Malayalam is presented in the following section.

I.4 The Syllable Structure of Malayalam

A syllable in speech is typically composed of a mandatory vowel nucleus, along with optional consonants or consonant clusters in onset and coda positions as shown in Fig. I.1. The sequence of characters and signs that constitute a valid syllable in Malayalam can be summarized as [117, 119]:

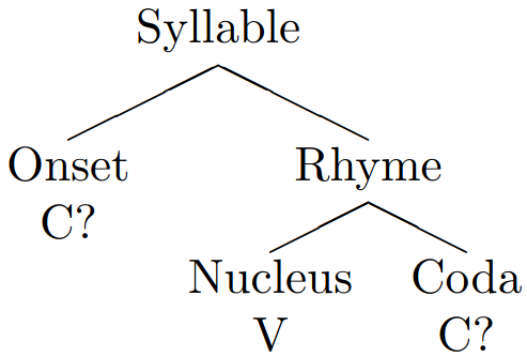


Figure I.1: Structure of a syllable. C-consonant, V-vowel, ?- indicates optionality

1. Every independent vowel occurring at word beginning is a syllable.

   eg: അ /a/ (V) in അമ്മ /a.mma/ (*mother*)

2. Every consonant or consonant cluster with or without vowel sign at end is a syllable.

   eg: ക /ka/ (CV) in കളി /ka.l̪i/ (*game*),

   കി /ki/ (CV) in കിളി /ki.l̪i/ (*bird*),

   സ്ത /st̪a/ (CCV) in പുസ്തകം /pu.st̪a.kam/ (*book*),

   ഷ്ടി /ʂʈi/ (CCV) in ഇഷ്ടിക /i.ʂʈi.ka/ (*brick*)

3. If there is a *chillu*, *anusvara* or *visarga* at the end of case 1 or 2 described above, it becomes the coda and joins to the previous syllable.

   eg: വൻ /ʋan/ (CVC) in അവൻ /a.ʋan/ (*he*),

   അം /am/ (VC) in അംബുജം /am.bu.ɟam/ (*lotus*),

   സ്ത്രം /st̪ram/ (CCCVC) in അസ്ത്രം /a.st̪ram/ (*arrow*)

4. A consonant or consonant cluster followed by a *virama* preceded by optional u-vowel (ു) sign, if and only if at word ends, is a syllable. In this scenario, the vowel sound is schwa /ə/, which does not have an explicit vowel grapheme in Malayalam.

   eg: ന് /nə/ (CV) in അവന് /a ʋa nə/ (*him*),

ട്ട /ʈʈə/ (CCV) in പട്ട /pa ʈʈə/ (*silk*),

ട്ടു /ʈʈə/ (CCV) in പട്ടു /pa ʈʈə/ (*silk*),

A sequence of characters that do not belong to any of the classes listed above, will not form a valid syllable and can not be accepted for pronunciation analysis. A vowel sign following an independent vowel (അി), a word beginning with a virama (്കം), an independent vowel after a consonant (കിഅ) etc. are examples of invalid sequences.

Bibliography

[1] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3(175):12, 2002.

[2] K-F Lee, H-W Hon, and Raj Reddy. An overview of the sphinx speech recognition system. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(1):35–45, 1990.

[3] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[4] Alexandru-Lucian Georgescu, Alessandro Pappalardo, Horia Cucu, and Michaela Blott. Performance vs. hardware requirements in state-of-the-art automatic speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2021(1):1–30, 2021.

[5] Sebastian P. Bayerl and Korbinian Riedhammer. A Comparison of Hybrid and End-to-End Models for Syllable Recognition. In Kamil Ekštein, editor, *Text, Speech, and Dialogue*, pages 352–360, Cham, 2019. Springer International Publishing.

[6] Aku Rouhe, Astrid Van Camp, Mittul Singh, Hugo Van Hamme, and Mikko Kurimo. An Equal Data Setting for Attention-Based Encoder-Decoder and HMM/DNN Models: A Case Study in Finnish ASR . In Alexey Karpov and Rodmonga Potapova, editors, *Speech and Computer*, pages 602–613, Cham, 2021. Springer International Publishing.

[7] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100, 2014.

[8] Kaushal Bhogale, Abhigyan Raman, Tahir Javed, Sumanth Doddapaneni, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[9] G Bharadwaja Kumar, Kavi Narayana Murthy, and BB Chaudhuri. Statistical analyses of telugu text corpora. *IJDL. International journal of Dravidian linguistics*, 36(2):71–99, 2007.

[10] Santhosh Thottingal. Finite State Transducer based Morphology analysis for Malayalam Language. In *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*, pages 1–5, Dublin, Ireland, August 2019. European Association for Machine Translation.

[11] Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages . In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 380–387, 2007.

[12] Ken H Davis, R Biddulph, and Stephen Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[13] Bruce T Lowerre. *The harpy speech recognition system.* Carnegie Mellon University, 1976.

[14] Josh Meyer. *Multi-task and transfer learning in low-resource speech recognition*. PhD thesis, The University of Arizona, 2019.

[15] Mark Gales and Steve Young. The application of hidden Markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304, 2008.

[16] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[17] A Madhavaraj. *Strategies for Handling Large Vocabulary and Data Sparsity Problems for Tamil Speech Recognition*. PhD thesis, 2020.

[18] Jacob Benesty, M Mohan Sondhi, Yiteng Huang, et al. *Springer handbook of speech processing*, volume 1. Springer, 2008.

[19] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. IEEE, 2013.

[20] Steve J Young, Julian J Odell, and Phil C Woodland. Tree-based state tying for high accuracy modelling. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[21] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups . *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[22] Dong Yu, Li Deng, Dong Yu, and Li Deng. Deep neural network-hidden markov model hybrid systems. *Automatic Speech Recognition: A Deep Learning Approach*, pages 99–116, 2015.

[23] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth annual conference of the international speech communication association*, 2015.

[24] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Proc. Interspeech 2018*, pages 3743–3747, 2018.

[25] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *Interspeech*, volume 2013, pages 2345–2349, 2013.

[26] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755, 2016.

[27] R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, 1995.

[28] Mehryar Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2):269–311, 1997.

[29] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlíček, Yanmin Qian, et al. Generating exact lattices in the wfst framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4213–4216. IEEE, 2012.

[30] Dan Jurafsky and James H Martin. *Speech and language processing*. Pearson London, 2nd edition, 2014.

[31] Rohit Prabhavalkar, Takaaki Hori, Tara N Sainath, Ralf Schlüter, and Shinji Watanabe. End-to-end speech recognition: A survey. *arXiv preprint arXiv:2303.03329*, 2023.

[32] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[33] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[34] Hasim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *Interspeech*, volume 8, pages 1298–1302, 2017.

[35] C S Anoop and A G Ramakrishnan. Exploring a unified asr for multiple south indian languages leveraging multilingual acoustic and language models. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 830–837, 2023.

[36] Peter Smit, Sami Virpioja, and Mikko Kurimo. Advances in subword-based HMM-DNN speech recognition across languages. *Computer Speech & Language*, 66:101158, 2021.

[37] Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. Unsupervised speech recognition. *Advances in Neural Information Processing Systems*, 34, 2021.

[38] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[39] Piotr Kłosowski. A Rule-Based Grapheme-to-Phoneme Conversion System. *Applied Sciences*, 12(5), 2022.

[40] Anoop Kunchukuttan and Pushpak Bhattacharyya. Orthographic Syllable as basic unit for SMT between Related Languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1912–1917, Austin, Texas, November 2016. Association for Computational Linguistics.

[41] Alan W Black, Kevin Lenzo, and Vincent Pagel. Issues in building general letter to sound rules. In *The third ESCA/COCOSDA workshop (ETRW) on speech synthesis*, 1998.

[42] Eric Fosler-Lussier, Yanzhang He, Preethi Jyothi, and Rohit Prabhavalkar. Conditional random fields in speech, audio, and language processing. *Proceedings of the IEEE*, 101(5):1054–1075, 2013.

[43] Francois Yvon. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. *arXiv preprint cmp-lg/9608006*, 1996.

[44] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451, 2008.

[45] Kaisheng Yao and Geoffrey Zweig. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. 2015.

[46] Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. Grapheme-to-phoneme conversion with convolutional neural networks. *Applied Sciences*, 9(6):1143, 2019.

[47] Yolchuyeva Sevinj, Németh Géza, and Gyires-Tóth Bálint. Transformer based grapheme-to-phoneme conversion. *Proceedings of Interspeech 2019*, pages 2095–2099, 2019.

[48] Xinjian Li, Florian Metze, David Mortensen, Shinji Watanabe, and Alan Black. Zero-shot Learning for Grapheme to Phoneme Conversion with Language Ensemble. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2106–2115, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[49] Josef Robert Novak, Nobuaki Minematsu, and Keikichi Hirose. Phonetisaurus: Exploring Grapheme-to-phoneme Conversion with joint n-gram models in the WFST framework . *Natural Language Engineering*, 22(6):907–938, 2016.

[50] Vathnak Sar and Tien-Ping Tan. Applying Linguistic G2P Knowledge on a Statistical Grapheme-to-phoneme Conversion in Khmer. *Procedia Computer Science*, 161:415–423, 2019.

[51] Kemal Oflazer and Sharon Inkelas. The Architecture and the Implementation of a Finite State Pronunciation Lexicon for Turkish. *Comput. Speech Lang.*, 20(1):80–106, January 2006.

[52] Tadesse Anberbir, Michael Gasser, Tomio Takara, and Kim Dong Yoon. Grapheme-to-phoneme conversion for Amharic text-to-speech system. In *Proceedings of conference on human language technology for development, Bibliotheca Alexandrina, Alexandria*, 2011.

[53] David R. Mortensen, Siddharth Dalmia, and Patrick Littell. Epitran: Precision G2P for Many Languages. In *Proceedings of the Eleventh International Conference on*

Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[54] Jonathan Duddington and R Dunn. eSpeak text to speech. *Web publication: http://espeak.sourceforge.net*, 2012.

[55] Arun Baby, NL Nishanthi, Anju Leela Thomas, and Hema A Murthy. A Unified Parser for developing Indian language text to speech synthesizers. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *International Conference on Text, Speech, and Dialogue*, pages 514–521, Cham, 2016. Springer, Springer International Publishing.

[56] Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson, and Alan W Black. The festvox indic frontend for grapheme to phoneme conversion. In *WILDRE: Workshop on Indian Language Data-Resources and Evaluation*, 2016.

[57] Vinodh Rajan. Aksharamukha script converter web application., 2018. https://aksharamukha.appspot.com/about.

[58] Anoop Kunchukuttan. The IndicNLP Library. `https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf`, 2020.

[59] Sreeja Manghat, Sreeram Manghat, and Tanja Schultz. Malayalam-English Code-Switched: Grapheme to Phoneme System. In *Proc. Interspeech 2020*, pages 4133–4137, 2020.

[60] Aswathy P V, Arun Gopi, Sajini T, and Bhadran V K. Improving the accuracy of pronunciation lexicon using Naive Bayes classifier with character n-gram as feature: for language classified pronunciation lexicon generation . In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 113–118, Goa, India, December 2014. NLP Association of India.

[61] R. Priyamvada, D. Govind, Vijay Krishna Menon, B. Premjith, and K. P. Soman. Grapheme to Phoneme Conversion for Malayalam Speech Using Encoder-Decoder Architecture. In Suresh Chandra Satapathy, Peter Peer, Jinshan Tang, Vikrant Bhateja, and Anumoy Ghosh, editors, *Intelligent Data Engineering and Analytics*, pages 41–49, Singapore, 2022. Springer Nature Singapore.

[62] Kevin Lenzo. The CMU pronouncing dictionary, 2007. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[63] Tanja Schultz and Tim Schlippe. GlobalPhone: Pronunciation Dictionaries in 20 Languages. In *Proceedings of the Ninth International Conference on Language*

Resources and Evaluation (LREC'14), pages 337–341, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[64] F de Vriend, N Castell, J Giménez, and G Maltese. LC-STAR: XML-coded Phonetic Lexica and Bilingual corpora for Speech-to-Speech Translation. 2004.

[65] Ahmed Ali. Arabic Speech Recognition Pronunciation Dictionary LDC2017L01. Web Download. Philadelphia: Linguistic Data Consortium, 2017. , March 2017. https://catalog.ldc.upenn.edu/LDC2017L01.

[66] Xian Huang, Xin Jin, Qike Li, and Keliang Zhang. On Construction of the ASR-oriented Indian English Pronunciation Dictionary. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6593–6598, Marseille, France, May 2020. European Language Resources Association.

[67] Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat. TTS for Low Resource Languages: A Bangla Synthesizer. In *10th edition of the Language Resources and Evaluation Conference, 23-28 May 2016*, pages 2005–2010, Portorož, Slovenia, 2016.

[68] Cini Kurian. Speech database and text corpora for Malayalam language automatic speech recognition technology . In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 7–11, 2016.

[69] G Deekshitha, K R Sreelakshmi, Ben P Babu, and Leena Mary. Development of Spoken Story Database in Malayalam Language. In *2018 4th International Conference on Electrical Energy Systems (ICEES)*, pages 530–533, 2018.

[70] Lekshmi K R, Jithesh V S, and Elizabeth Sherly. Malayalam Speech Corpus: Design and Development for Dravidian Language. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*, pages 25–28, Marseille, France, May 2020. European Language Resources Association (ELRA).

[71] Kavya Manohar, A. R. Jayan, and Rajeev Rajan. Quantitative Analysis of the Morphological Complexity of Malayalam Language. In Petr Sojka, Ivan Kopeček, Karel Pala, and Aleš Horák, editors, *Text, Speech, and Dialogue*, pages 71–78, Cham, 2020. Springer, Springer International Publishing.

[72] Max Bane. Quantifying and Measuring Morphological Complexity. In *Proceedings of the 26th west coast conference on formal linguistics*, pages 69–76. Cascadilla Proceedings Project Somerville, MA, 2008.

[73] Ximena Gutierrez-Vasques and Victor Mijangos. Comparing morphological complexity of Spanish, Otomi and Nahuatl. *arXiv preprint arXiv:1808.04314*, pages 30–37, August 2018.

[74] Christian Bentz, Tatjana Soldatova, Alexander Koplenig, and Tanja Samardžić. A comparison between morphological complexity measures: typological data vs. language corpora. pages 142–153, 2016.

[75] Kimmo Kettunen. Can type-token ratio be used to show morphological complexity of languages? *Journal of Quantitative Linguistics*, 21(3):223–245, 2014.

[76] Michael A Covington and Joe D McFall. Cutting the Gordian knot: The moving-average type–token ratio (MATTR). *Journal of quantitative linguistics*, 17(2):94–100, 2010.

[77] Matthew Baerman, Dunstan Brown, and Greville G Corbett. *Understanding and measuring morphological complexity*. Oxford University Press, USA, USA, 2015.

[78] Peter Smit, Sami Virpioja, and Mikko Kurimo. Improved Subword Modeling for WFST-Based Speech Recognition. In *Proc. Interspeech 2017*, pages 2551–2555, 2017.

[79] Aditya Yadavalli, Shelly Jain, Ganesh Mirishkar, and Anil Kumar Vuppala. Investigation of subword-based bilingual automatic speech recognition for indian languages. In *Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing*, IC3-2022, page 234–241, New York, NY, USA, 2022. Association for Computing Machinery.

[80] Bharathi Pilar et al. Subword Dictionary Learning and Segmentation Techniques for Automatic Speech Recognition in Tamil and Kannada . *arXiv preprint arXiv:2207.13331*, 2022.

[81] Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. Automatic Speech Recognition in Sanskrit: A New Speech Corpus and Modelling Insights. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5039–5050, Online, August 2021. Association for Computational Linguistics.

[82] Sreeja Manghat, Sreeram Manghat, and Tanja Schultz. Hybrid sub-word segmentation for handling long tail in morphologically rich low resource languages . In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6122–6126, 2022.

[83] Kishore Prahallad, E Naresh Kumar, Venkatesh Keri, S Rajendran, and Alan W Black. The IIIT-H Indic speech databases. In *Thirteenth annual conference of the international speech communication association*, page 4, 2012.

[84] Arun Baby, Anju Leela Thomas, NL Nishanthi, TTS Consortium, et al. Resources for Indian languages. In *Proceedings of Text, Speech and Dialogue*. CBBLR Workshop, 2016.

[85] Fei He, Shan-Hui Cathy Chu, Oddur Kjartansson, Clara Rivera, Anna Katanova, Alexander Gutkin, Isin Demirsahin, Cibu Johny, Martin Jansche, Supheakmungkol Sarin, and Knot Pipatsrisawat. Open-source Multi-speaker Speech Corpora for Building Gujarati, Kannada, Malayalam, Marathi, Tamil and Telugu Speech Synthesis Systems . In *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, pages 6494–6503, Marseille, France, May 2020. European Language Resources Association (ELRA).

[86] Kavya Manohar. Releasing Malayalam Speech Corpus., 2020. https://blog.smc.org.in/malayalam-speech-corpus/.

[87] Deepa P Gopinath, Thennal D K, Vrinda V Nair, Swaraj K S, and Sachin G. Imasc – icfoss malayalam speech corpus, 2022.

[88] V. R. Vimal Krishnan, Athulya Jayakumar, and Anto P. Babu. Speech Recognition of Isolated Malayalam Words Using Wavelet Features and Artificial Neural Network . In *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, pages 240–243, Hong Kong, China, January 2008. IEEE.

[89] Cini Kurian, A Firoz Shah, and Kannan Balakrishnan. Isolated malayalam digit recogntion using support vector machines. In *2010 international conference on communication control and computing technologies*, pages 692–695. IEEE, 2010.

[90] Sonia Sunny, S David Peter, and K Poulose Jacob. Development of a speech recognition system for speaker independent isolated malayalam words. *International Journal of Computer Science & Engineering Technology*, 3(4):69–75, 2012.

[91] Cini Kurian and Kannan Balakrishnan. Speech recognition of Malayalam numbers. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 1475–1479, Coimbatore, India, 2009. IEEE.

[92] Cini Kurian and Kannan Balakrishnan. Malayalam isolated digit recognition using hmm and plp cepstral coefficient. *International journal of advanced information technology*, 1(5):31, 2011.

[93] Cini Kurian and Kannan Balakriahnan. Continuous Speech Recognition System for Malayalam Language Using PLP Cepstral Coefficient. *International Journal of Computing and Business Research*, 3(1):23, 2012.

[94] P. Shobana Devi, Jose Stephen, G. Sulochana Kurambath, and R. Ravindra Kumar. Implementation of dictation system for malayalam office document. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, page 763–767, New York, NY, USA, 2012. Association for Computing Machinery.

[95] Maya Moneykumar, Elizabeth Sherly, and Win Sam Varghese. Isolated word recognition system for Malayalam using machine learning. In *Proceedings of the 12th International Conference on Natural Language Processing*, pages 158–165, Trivandrum, India, December 2015. NLP Association of India.

[96] G Deekshitha, KR Sreelakshmi, Ben P Babu, and Leena Mary. Development of spoken story database in Malayalam language. In *2018 4th International Conference on Electrical Energy Systems (ICEES)*, pages 530–533. IEEE, 2018.

[97] Lavanya B. Babu, Anu George, K R Sreelakshmi, and Leena Mary. Continuous Speech Recognition System for Malayalam Language Using Kaldi. In *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, pages 1–4, 2018.

[98] Anuj Mohamed and K.N. Ramachandran Nair. HMM/ANN hybrid model for continuous Malayalam speech recognition. *Procedia Engineering*, 30:616–622, 2012.

[99] Ashana Mariam Moncy, Athira M., Hanna Jasmin, and Rajeev Rajan. Automatic Speech Recognition in Malayalam Using DNN-based Acoustic Modelling. In *2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pages 170–174, 2020.

[100] Ravi Sankar S Nair. : A grammar of malayalam. *Language in India*, 12:1–135, 2012.

[101] Ronald E Asher and TC Kumari. *Malayalam*. Psychology Press, 1997.

[102] Georgi Georgiev, Valentin Zhikov, Kiril Simov, Petya Osenova, and Preslav Nakov. Feature-rich part-of-speech tagging for morphologically complex languages: Application to Bulgarian. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 492–502, Avignon, France, April 2012. Association for Computational Linguistics.

[103] Irina Kipyatkova and Alexey Karpov. Study of Morphological factors of factored language models for Russian ASR. In *International Conference on Speech and Computer*, pages 451–458. Springer, 2014.

[104] Edvin Pakoci, Branislav Popović, and Darko Pekar. Using Morphological Data in Language Modeling for Serbian Large Vocabulary Speech Recognition. *Computational Intelligence and Neuroscience*, 2019:1–8, March 2019.

[105] Tommi Pirinen. Weighted Finite-State Methods for Spell-Checking and Correction. *Helsinki: University of Helsinki*, 2014.

[106] Swathanthra Malayalam Computing. Malayalam Text Corpora, March 2020. https://gitlab.com/smc/corpus.

[107] Mark Davis and Martin Dürst. Unicode normalization forms, 2001.

[108] Hla Hla Htay, G Bharadwaja Kumar, and Kavi Narayana Murthy. Statistical Analyses of Myanmar Corpora. *Department of Computer and Information Sciences, University of Hyderabad*, pages 1–15, 2007.

[109] Masako Fidler and Václav Cvrček. *Taming the Corpus: From Inflection and Lexis to Interpretation*. Springer, 1 edition, 2018.

[110] Florian Coulmas. *The Blackwell Encyclopedia of Writing Systems*. John Wiley & Sons, Ltd, 1999.

[111] David Crystal. *A Dictionary of Linguistics and Phonetics*, volume 30. John Wiley & Sons, 2011.

[112] William Bright. A Matter of Typology: Alphasyllabaries and Abugidas. *Written Language & Literacy*, 2(1):45–55, 1999.

[113] Ronald M. Kaplan and Martin Kay. Regular Models of Phonological Rule Systems. *Computational Linguistics*, 20(3):331–378, 1994.

[114] Lauri Karttunen and Kenneth R Beesley. *Two-level rule compiler*. Xerox Corporation, Palo Alto Research Center, 1992.

[115] Ayla Kayabaş, Helmut Schmid, Ahmet Ercan Topcu, and Özkan Kiliç. TRMOR: a finite-state-based morphological analyzer for Turkish. *Turkish Journal of Electrical Engineering and Computer Sciences*, 27(5):3837–3851, 2019.

[116] Helmut Schmid. A programming language for finite state transducers. In *FSMNLP*, volume 4002, pages 308–309. Citeseer, 2005.

[117] Tara Mohanan. Syllable structure in Malayalam. *Linguistic inquiry*, 20(4):589–626, 1989.

[118] Ronald E Asher and T C Kumari. *Malayalam (Descriptive grammars)*. Routledge, London and New York, NY, 1997.

[119] V. R. Prabodhachandran Nair. ഭാഷാശാസ്ത്രപരിചയം */bʰaːʂaːʃaːsʈraparitʃajam/ (Introduction to Linguistics)*. MaluBen Publications, Thiruvananthapuram, Kerala, 2016.

[120] Žiga Golob, Jerneja Žganec Gros, Mario Žganec, Boštjan Vesnicer, and Simon Dobrišek. FST-based pronunciation lexicon compression for speech engines. *International Journal of advanced robotic systems*, 9(5):211, 2012.

[121] Helmut Schmid, Arne Fitschen, and Ulrich Heid. SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[122] Uwe Springmann, Helmut Schmid, and Dietmar Najock. LatMor: A Latin finite-state morphology encoding vowel quantity. *Open Linguistics*, 2(1), 2016.

[123] Helmut Schmid. *SFST manual*, 2005. `https://www.cis.uni-muenchen.de/~schmid/tools/SFST/data/SFST-Manual.pdf`.

[124] Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Avik Bhattacharyya, Mitesh M Khapra, Pratyush Kumar, et al. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for Indic languages. 2020.

[125] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2020.

[126] Shrikant Malviya, Rohit Mishra, and Uma Shanker Tiwary. Structural analysis of Hindi phonetics and a method for extraction of phonetically rich sentences from a very large Hindi text corpus . In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 188–193. IEEE, 2016.

[127] Humberto M Torres, Jorge A Gurlekian, Diego A Evin, and Christian G Cossio Mercado. Emilia: a speech corpus for Argentine Spanish text to speech synthesis. *Language Resources and Evaluation*, 53(3):419–447, 2019.

[128] Nimisha Srivastava, Rudrabha Mukhopadhyay, Prajwal K R, and C V Jawahar. IndicSpeech: Text-to-Speech Corpus for Indian Languages. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6417–6422, Marseille, France, May 2020. European Language Resources Association.

[129] Andreas Stolcke. SRILM-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[130] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

[131] Reinhold Haeb-Umbach and Hermann Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *icassp*, volume 92, pages 13–16. Citeseer, 1992.

[132] Mark JF Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE transactions on speech and audio processing*, 7(3):272–281, 1999.

[133] Mark JF Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[134] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth annual conference of the international speech communication association*, 2015.

[135] K R Lekshmi and Elizabeth Sherly. An ASR System for Malayalam Short Stories using Deep Neural Network in KALDI. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pages 972–979, 2021.

[136] Bao Thai, Robert Jimerson, Raymond Ptucha, and Emily Prud'hommeaux. Fully Convolutional ASR for Less-Resourced Endangered Languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 126–130, Marseille, France, May 2020. European Language Resources association.

[137] Sabrina J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. What Kind of Language Is Hard to Language-Model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy, July 2019. Association for Computational Linguistics.

[138] Hyunji Hayley Park, Katherine J. Zhang, Coleman Haley, Kenneth Steimel, Han Liu, and Lane Schwartz. Morphology Matters: A Multilingual Language Modeling Analysis. *Transactions of the Association for Computational Linguistics*, 9:261–276, 03 2021.

[139] Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary

words across languages. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(1):1–29, dec 2007.

[140] Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited Vocabulary Speech Recognition with Morph Language Models Applied to Finnish. *Computer Speech & Language*, 20(4):515–541, 2006.

[141] G. Choueiter, D. Povey, S.F. Chen, and G. Zweig. Morpheme-Based Language Modeling for Arabic Lvcsr. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I, 2006.

[142] Weiran Wang, Guangsen Wang, Aadyot Bhatnagar, Yingbo Zhou, Caiming Xiong, and Richard Socher. An Investigation of Phone-Based Subword Units for End-to-End Speech Recognition. In *Proc. Interspeech 2020*, pages 1778–1782, 2020.

[143] Wei Zhou, Mohammad Zeineldeen, Zuoyun Zheng, Ralf Schlüter, and Hermann Ney. Acoustic Data-Driven Subword Modeling for End-to-End Speech Recognition. In *Proc. Interspeech 2021*, pages 2886–2890, 2021.

[144] Hainan Xu, Shuoyang Ding, and Shinji Watanabe. Improving End-to-end Speech Recognition with Pronunciation-assisted Sub-word Modeling. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7110–7114, 2019.

[145] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[146] Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates . In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[147] Harjeet Singh, Rajendra Kumar Sharma, and VP Singh. Online Handwriting Recognition Systems for Indic and non-Indic scripts: A Review. *Artificial Intelligence Review*, 54(2):1525–1579, 2021.

[148] Mathias Creutz and Krista Lagus. Unsupervised Discovery of Morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30, 2002.

[149] Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. 2013.

[150] Philip Gage. A New Algorithm for Data Compression. *C Users Journal*, 12(2):23–38, 1994.

[151] RR Rajeev and Elizabeth Sherly. A suffix Stripping based Morph Analyser for Malayalam Language. In *Proceedings of 20th Kerala Science Congress*, pages 482–484, 2007.

[152] OR Rinju, RR Rajeev, PC Reghu Raj, and Elizabeth Sherly. Morphological Analyzer for Malayalam: Probabilistic Method vs Rule based Method. *International Journal of Computational Linguistics and Natural Language Processing*, 2(10):502–507, 2013.

[153] PJ Antony and KP Soman. Computational Morphology and Natural Language Parsing for Indian Languages: A Literature Survey . *International Journal of Scientific and Engineering Research*, 3, 2012.

[154] VP Abeera, S Aparna, RU Rekha, M Anand Kumar, V Dhanalakshmi, KP Soman, and S Rajendran. Morphological analyzer for Malayalam using Machine Learning. In *International Conference on Data Engineering and Management*, pages 252–254. Springer, 2010.

[155] Premjith B, Soman K.P., and M Anand Kumar. A Deep Learning Approach for Malayalam Morphological Analysis at Character Level. *Procedia Computer Science*, 132:47–54, 2018. International Conference on Computational Intelligence and Data Science.

[156] Piotr Żelasko, Siyuan Feng, Laureano Moro Velázquez, Ali Abavisani, Saurabhchand Bhati, Odette Scharenborg, Mark Hasegawa-Johnson, and Najim Dehak. Discovering Phonetic Inventories with Crosslingual Automatic Speech Recognition. *Comput. Speech Lang.*, 74(C), jul 2022.

[157] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted Finite-state Transducers in Speech Recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[158] Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* . Pearson, India, 2009.

[159] Amr Ibrahim El-Desoky Mousa. *Sub-Word Based Language Modeling of Morphologically Rich Languages for LVCSR*. PhD thesis, RWTH Aachen University, 2014.

[160] Sabrina J. Mielke. Can you compare perplexity across different segmentations?, Apr 2019.

[161] Piotr Kłosowski. Statistical analysis of orthographic and phonemic language corpus for word-based and phoneme-based Polish language modelling . *EURASIP Journal on Audio, Speech, and Music Processing*, 2017(1):1–16, 2017.

[162] Maximilian Bisani and Hermann Ney. Open vocabulary speech recognition with flat hybrid models. 2005.

[163] Rudolf A Braun, Srikanth Madikeri, and Petr Motlicek. A Comparison of Methods for OOV-Word Recognition on a New Public Dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5979–5983. IEEE, 2021.

[164] Kavya Manohar, A. R. Jayan, and Rajeev Rajan. Mlphon: A Multifunctional Grapheme-Phoneme Conversion Tool Using Finite State Transducers. *IEEE Access*, 10:97555–97575, 2022.

List of Publications

Journals (SCIE Indexed)

[1] Manohar, K., Jayan, A.R. and Rajan, R., "Mlphon: A Multifunctional Grapheme-Phoneme Conversion Tool Using Finite State Transducers", *IEEE Access* (2022), vol. 10, pp. 97555-97575, https://doi.org/10.1109/ACCESS.2022.3204403.

[2] Manohar, K., Jayan, A. R. and Rajan, R., "Improving speech recognition systems for the morphologically complex Malayalam language using subword tokens for language modeling.", *EURASIP Journal on Audio, Speech, and Music Processing* 2023, 47 (2023). https://doi.org/10.1186/s13636-023-00313-7

Conferences:

[1] Manohar, K., Jayan, A. R. and Rajan, R., "Quantitative analysis of the morphological complexity of Malayalam language", In *Text, Speech, and Dialogue: 23rd International Conference, TSD 2020, Brno, Czech Republic* (2020), Springer International Publishing. https://doi.org/10.1007/978-3-030-58323-1_7.

[2] Manohar, K., Jayan, A. R. and Rajan, R., 'Syllable Subword tokens for Open Vocabulary Speech Recognition in Malayalam", In *Third Workshop on NLP solutions for Under Resourced Languages, Trento, Italy (NSURL 2022)* (2022), .

[3] Manohar K., Menon G., Abraham A., Rajan R., Jayan A. R. "Automatic Recognition of Continuous Malayalam Speech using Pretrained Multilingual Transformers," In *International Conference on Intelligent Systems for Communication, IoT and Security, Coimbatore, India. (ICISCoIS 2023)*

[4] Manohar, K., Jayan, A. R. and Rajan, R., 'An Open framework for the development of automatic speech recognition in Malayalam language", In *Thirty Fifth Kerala Science Congress, Kerala, India* (2023). (**Recieved Best Paper Award in the category of Scientific Social Responsibility**)
