

## APPLIED RESEARCH

# Mlphon: A Multifunctional Grapheme-Phoneme Conversion Tool Using Finite State Transducers

KAVYA MANOHAR<sup>1,3</sup>, A. R. JAYAN<sup>2,3</sup>, AND RAJEEV RAJAN<sup>1,3</sup>, (Senior Member, IEEE)

<sup>1</sup>College of Engineering Trivandrum, Thiruvananthapuram, Kerala 695 586, India

<sup>2</sup>Government Engineering College, Thrissur, Kerala 680 009, India

<sup>3</sup>Department of Electronics and Communication Engineering, APJ Abdul Kalam Technological University, Thiruvananthapuram, Kerala 695 016, India


Corresponding author: Kavya Manohar (sakhi.kavya@gmail.com)

**ABSTRACT** In this article we present the design and the development of a knowledge based computational linguistic tool, Mlphon [em.el.fo:ɾŋ] for Malayalam language. Mlphon computationally models linguistic rules using finite state transducers and performs multiple functions including grapheme to phoneme (g2p) and phoneme to grapheme (p2g) conversions, syllabification, phonetic feature analysis and script grammar check. This open source software tool, released under MIT license, is developed as a one-stop solution to handle different speech related text processing tasks for automatic speech recognition, text to speech synthesis and non-speech natural language processing tasks including syllable subword based language modeling, phoneme diversity analysis and text sanity check. The tool is evaluated on a manually crafted gold standard lexicon. Mlphon performs orthographic syllabification with 99% accuracy with a syllable error rate of 0.62% on the gold standard lexicon. For grapheme to phoneme conversion task, overall phoneme recognition accuracy of 99% with a phoneme error rate of 0.55% is obtained on gold standard lexicon. Additionally an extrinsic evaluation of Mlphon is performed by employing the pronunciation lexicon created using Mlphon, in Malayalam automatic speech recognition (ASR) task. Performance analysis in terms of the computation time of lexicon creation process and the word error rate (WER) on ASR task are presented along with a comparison over other automated tools for lexicon creation. Pronunciation lexicons with more than 100k commonly used Malayalam words in phonemised and syllabified forms is created and they are published as open language resources along with this work. We also demonstrate the usage of Mlphon on different natural language processing applications - syllable subword ASR, assisted pronunciation learning, phoneme diversity analysis and text sanity check. Being a knowledge based solution with open source code, Mlphon can be adapted to other languages of similar script nature.

**INDEX TERMS** Computational Phonology, Low Resource Languages, Pronunciation Lexicon, Malayalam, Software Tool, Speech Recognition, Syllabification.

## I. INTRODUCTION

Precise text processing taking care of intricate linguistic details is a pre-requisite for many downstream natural language processing (NLP) tasks. This applied research work 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 and some general purpose NLP tasks. Mlphon is built on finite state transducers (FSTs) to

The associate editor coordinating the review of this manuscript and approving it for publication was Orazio Gambino .

perform multiple functions including grapheme to phoneme (g2p) and phoneme to grapheme (p2g) conversions, syllabification on graphemes as well as phonemes, phonetic feature analysis, and script grammar check for Malayalam. The features of Mlphon are accessible through a programmable python API, that can be integrated with the development process of automatic speech recognition (ASR) and text to speech synthesis (TTS).

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 [1], [2]. The correspondence between the two, largely depends on the nature

of the writing system. For grapheme-phoneme conversion tasks, most alphasyllabary languages have precise rule sets, unlike non-phonemic scripts like English [3], [4]. 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 [5]. 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.

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 [6] and [7]. This has made FST popular in many fundamental NLP applications [3], [8], [9], [10], [11]. The rule set of Mlphon is written in Stuttgart finite state toolkit (SFST) formalism and compiled to FSTs [12].

The rest of this article is organized as follows. Section II provides the motivation behind the proposed tool and section III explains how it relates to similar tools reported in literature. Section IV describes the nature of grapheme and phoneme inventories of Malayalam and section V explains the computational rules of Malayalam script syllabification. Section VI describes the design and development of Mlphon and explains how the linguistic rules are incorporated in Mlphon architecture. Section VII presents the evaluation of Mlphon against a gold standard reference. Performance analysis and comparison with other lexicon creation tools on ASR task is presented in section VIII. The usage of Mlphon to create the largest openly available pronunciation lexicon for Malayalam is described in section IX. Section X describes the applications of Mlphon in text sanity check, assisted pronunciation learning, phoneme diversity analysis and syllable subword based ASR. Section XI concludes the article with a summary of the work.

## II. MOTIVATION

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 [13]. Malayalam is a morphologically complex low resource language with very little transcribed audio datasets and no openly available pronunciation lexicons.

Morphologically complex languages with very large number of rare words are challenging for machine translation and ASR tasks due to huge out of vocabulary (OOV) rate. Malayalam language is known to demonstrate a high level of morphological complexity than many other Indian and European languages in terms of type-token ratio and type-token growth rate [14], [15]. For languages with very little transcribed audio datasets available for speech related tasks, a precise grapheme to phoneme conversion can ensure better acoustic modeling, even in end-to-end [16] ASR systems.

Segmenting words to syllables has got its 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 [17], [18]. In many languages, g2p correspondence depends on the relative position of grapheme within a word and a syllable, which makes syllable boundary identification further more important for phoneme level analysis. In the era of large language models being built on web crawled text corpora [19], it is necessary to ensure the sanity of text. Checking for the linguistic validity of character sequences can guarantee this to a large extent.

Availability of a ready to use pronunciation lexicon is an essential linguistic resource for ASR (DNN/HMM pipeline model) and TTS tasks. There are machine readable pronunciation dictionaries available for various world languages. CMUDict is an open source machine readable pronunciation lexicon for North American English that contains over 134k words and their pronunciations [20]. Similar efforts for creating pronunciation lexicons for different world languages are reported in literature, namely; Globalphone, providing pronunciation lexicon of 20 world languages [21], the LC-STAR Phonetic Lexica of 13 different languages [22], Arabic speech recognition pronunciation lexicon with two million pronunciation entries for 526k Modern Standard Arabic words [23], ASR oriented Indian English pronunciation lexicon [24], manually curated Bangla phonetic lexicon of 65k lexical entries prepared for TTS [25], 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 [26], [27]. Agricultural speech and text corpora for Malayalam with 4k manually transcribed phonetic lexicon entries has been reported by Lekshmi et al. [28]. 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 [14]. 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. These lexicons can serve as high quality annotated data sets for bootstrapping data driven g2p training.

The need to perform precise grapheme to phoneme 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, Mlphon offers certain unique features compared to these prior works which makes it a well suited tool for ASR related tasks in Malayalam.

### III. RELATED WORKS

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.

Data driven approaches perform g2p mapping by dictionary lookups [29], decision trees [29], conditional random fields [30], pronunciation by analogy [31] or joint sequence alignments [32]. Recently deep learning architectures for g2p developed based on recurrent neural networks [33], convolutional neural networks [34] and transformers [35]. Zero shot g2p learning 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 [36]. Phonetisaurus [37] 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 yet been reported.

For languages with regular grapheme to phoneme conversion patterns, knowledge-based g2p has been reported to produce good results [36], [38]. A set of sequential rewrite rules can be used to achieve this. Agglutinative languages like Turkish [8] and Amharic [9] have reported works on language specific knowledge based g2p conversion using FST technology. Epitran [3], 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 [38].

#### A. REQUIREMENT ANALYSIS: A COMPARISON WITH RELATED TOOLS IN MALAYALAM

This section focuses on related tools that works for Malayalam. Being a language with regular orthography, most of the g2p conversion tools in Malayalam follow knowledge based approaches [4], [39], [40], [41], [42], [43], [44] rather than data driven methods. The only data driven method is based on encoder-decoder architecture [45] and uses data prepared using an existing knowledge based solution. Table 1 compares the functionalities of different grapheme-phoneme

**TABLE 1. Comparing the functionalities and features of grapheme - phoneme 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 [4]			✓	✓	✓			✓	
Espeak [39]			✓	✓	✓			✓	✓
Festvox [40]			✓	✓	✓			✓	
Aksharamukha [41]			✓			✓		✓	✓
Indic NLP [42]		✓	✓					✓	✓
Code-switched [43]			✓	✓					
LTS [44]			✓	✓					
Encoder-Decoder [45]			✓	✓					
<b>Mlphon</b>	✓	✓	✓	✓	✓	✓	✓	✓	✓

conversion tools available for Malayalam. Here, we examine each of their features and drawbacks in detail:

- Unified parser [4] 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 syllabify graphemes.
- Espeak [39] 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 [40]. 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 [41] 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 [42], 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 [43], where English words are phoneme mapped using CMUDict<sup>1</sup> and contextual rule based FST was used for Malayalam. This

<sup>1</sup>CMUDict- The CMU Pronouncing Dictionary: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

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 [44]. 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 [45]. 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 ഹ (labiodental fricative and also as labial aspirated plosive) ങ (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.

## B. KEY CONTRIBUTIONS

The key contributions of this applied research work are:

- 1) The design and development of a knowledge based computational linguistic tool, Mlphon. It is the only tool with a programmable interface that can do every one of the following:
  - Syllabification on graphemes as well as phonemes.
  - Phonetic feature analysis.
  - g2p and p2g conversions.
- 2) Evaluation of Mlphon in comparison to a gold standard lexicon.
- 3) Comparison of Mlphon with other openly available tools for lexicon creation and their application on ASR task.
- 4) The publication of a large vocabulary pronunciation lexicon for Malayalam.
- 5) A description of the usage of Mlphon on various NLP applications:
  - Syllable level language modeling for open vocabulary ASR.
  - Web based tool for assisted pronunciation learning.
  - Script sanity check and correction.
- 6) Openly licensed resources and source codes (Appendix A).

## IV. GRAPHEME INVENTORY OF MALAYALAM

Malayalam belongs to the family of Brahmic writing systems that is alphasyllabary in nature [5]. 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.

### A. VOWELS

There are 16 vowels in Malayalam. It includes 5 short vowels, 5 long vowels, 2 diphthongs and 4 vocalics [46]. 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 ഐ /a/. See Table 2 for the list of all vowels in Malayalam and their IPA representations.

### B. CONSONANTS

There are 38 regular consonant graphemes in Malayalam [46]. This includes 21 plosives classified by their aspirational and voicing characteristics, 6 nasals, 4 fricatives, 3 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 3. 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 4.

### C. SIGNS

The special signs *virama* (◌̣), *dot reph* (◌̣̣), *anuswara* (◌̣̣̣) and *visarga* (◌̣̣̣̣) have their properties as tabulated in Table 5. *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 /r/. *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.

### D. 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 6 lists certain examples of consonant clusters in Malayalam and their constituents.

### V. THE SYLLABLE STRUCTURE OF MALAYALAM

A syllable in speech is typically composed of a mandatory vowel nucleus, along with optional consonants or consonant



TABLE 2. Short and long vowels in Malayalam and their IPA representations.

Vowels	അ a	ആ a:	ഇ i	ഈ i:	ഉ u	ഊ u:	ഋ ri	ൠ ri:	ഈ li	ഌ li:	എ e	ഈ e:	ഐ ai	ഓ o	ഔ o:	ഔ au
Vowel Signs		ാ	ി	ീ	ു	ൂ	ൃ	ൠ	ഌ	ഏ	െ	േ	ൈ	ൊ	ോ	ൗ

TABLE 3. Consonants in Malayalam and their IPA representations.

Place of Articulation	Manner of Articulation									
	Plosive <sup>a</sup>	Plosive <sup>b</sup>	Plosive <sup>c</sup>	Plosive <sup>d</sup>	Nasal	Trill	Tap	Fricative	Approximant	Lateral
Velar	ക ka	ഖ k <sup>h</sup> a	ഗ ga	ഘ g <sup>h</sup> a	ങ ṅa					
Palatal	ച ca	ഛ c <sup>h</sup> a	ജ ja	ഝ j <sup>h</sup> a	ഞ ña			ശ a	യ ja	
Retroflex	ട ṭa	ഢ ṭ <sup>h</sup> a	ഡ ḍa	ഢ ḍ <sup>h</sup> a	ണ ña			ഷ a	ഴ a	ള  a
Alveolar	ത ṭa				ന na	ര ra	റ ra	സ sa		ല la
Dental	ത ṭa	ഥ ṭ <sup>h</sup> a	ദ ḍa	ധ ḍ <sup>h</sup> a	ന na					
Labial	പ pa	ഫ p <sup>h</sup> a	ബ ba	ഭ b <sup>h</sup> a	മ ma					
Labiodental									വ va	
Glottal								ഹ ha		

<sup>a</sup> Unaspirated and Unvoiced    <sup>b</sup> Aspirated and Unvoiced    <sup>c</sup> Unaspirated and Voiced    <sup>d</sup> Aspirated and Voiced

TABLE 4. Chillus in Malayalam and their IPA representations. Top row lists the chillus and the corresponding bottom row shows the base consonants from which chillus were derived.

ക k	ങ ṅ	ൻ n	ൽ l	മ m	യ j	ൾ	ഴ ṣ	ര r
ക ka	ങ ṅa	ന na	ല la	മ ma	യ ja	ള  a	ഴ ṣa	ര ra

TABLE 5. Signs in Malayalam.

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

clusters in onset and coda positions as shown in Fig. 1. The sequence of characters and signs that constitute a valid syllable in Malayalam can be summarized as [47], [48]:

- 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.li/ (game),  
കി /ki/ (CV) in കിളി /ki.li/ (bird),  
സ്ത /st̪a/ (CCV) in പുസ്തകം /pu.st̪a.kam/ (book),  
ഷ്ടി /st̪i/ (CCV) in ഇഷ്ടിക /i.st̪i.ka/ (brick)
- 3) If there is a chillu, anuswara or visarga at the end of case 1 or 2 described above, it becomes the coda and joins to the previous syllable.  
eg: വൻ /van/ (CVC) in അവൻ /a.van/ (he),  
അം /am/ (VC) in അമ്പുഴം /am.bu.ʒam/ (lotus),  
സ്ത്രം /st̪ram/ (CCCVC) in അസ്ത്രം /a.st̪ram/ (arrow)
- 4) A consonant or consonant cluster followed by a virama preceded by optional u-vowel (ൠ) 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.

TABLE 6. Examples of consonant clusters in Malayalam and their constituents.

Consonant cluster	Constituent character sequence
ക kka	ക ka    ി    ക ka
ക ṅka	ങ ṅa    ി    ക ka
ക k a	ക ka    ി    ല la
ഘ ḍ <sup>h</sup> na	ഘ ḍ <sup>h</sup> a    ി    ന na
സ്ത st̪a	സ sa    ി    ത ṭa
ഗ ga	ഗ ga    ി    ര ra
ഗ ḍja	ഗ ga    ി    യ ja
ണ ñtra	ന na    ി    ത ṭa    ി    ര ra

eg: ന് /nə/ (CV) in അവൻ /a va nə/ (him),  
സ്ത /st̪ə/ (CCV) in പട്ട് /pa t̪ə/ (silk),  
സ്ത /st̪ə/ (CCV) in പട്ട് /pa t̪ə/ (silk),

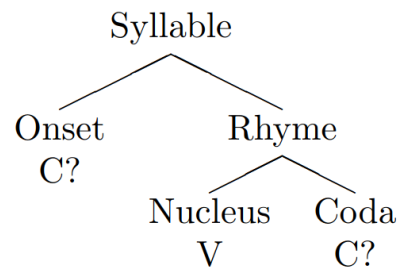


FIGURE 1. Structure of a syllable. C-consonant, V-vowel, ?- indicates optionality.

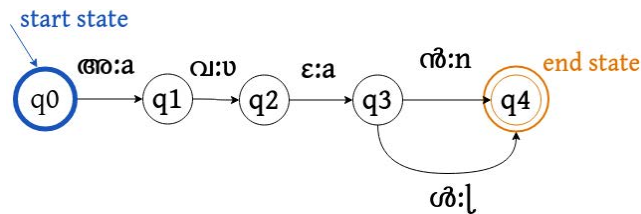
A sequence of characters that do not belong to any of the classes listed above, will not form a valid syllable and can not be accepted for pronunciation analysis. A vowel sign following an independent vowel (അി), a word beginning with a virama (ഃക്കം), an independent vowel after a consonant (കിഅ) etc.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

VI. THE DESIGN AND DEVELOPMENT OF MLPHON

Given that Malayalam linguistic literature contains well-established pronunciation modeling rules [46], [47], [48], we computationally model these rules in a deterministic

manner using finite state transducers, which are ideal for this task [6]. FSTs are apt for morphological as well as phonological parsing of natural languages [6]. An FST maps between two sets of symbols. Formally a finite state transducer  $T$  can be defined [49] as a set of seven parameters  $(Q, \Sigma, \Gamma, I, F, \delta, \sigma)$  where

- $Q$  is a finite set of states
- $\Sigma$  is a finite set of input symbols
- $\Gamma$  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 2.** An FST representing a simple pronunciation mapping that accepts two words അവാൻ and അവൾ. 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  $i : o$ .  $\epsilon$  is a special symbol that 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  $q_2$  to  $q_3$ .

**TABLE 7.** Parameters of FST illustrated in Fig. 2 is defined in this table.

Parameters	Definition
$Q$	{q0, q1, q2, q3, q4}
$\Sigma$	{അ, വ, ഞ, ള, എ}
$\Gamma$	{a, v, n, l}
$I$	{q0}
$F$	{q4}
$\delta$	$\delta(q_0, അ) = q_1; \delta(q_1, വ) = q_2; \delta(q_2, എ) = q_3;$ $\delta(q_3, ഞ) = q_4; \delta(q_3, ള) = q_4$
$\sigma$	$\sigma(q_0, അ) = a; \sigma(q_1, വ) = v; \sigma(q_2, എ) = a;$ $\sigma(q_3, ഞ) = n; \sigma(q_3, ള) = l$

In an FST, every state 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 [50]. For the cases where the number of symbols in input or output strings mismatch, a ‘null symbol’,  $\epsilon$  is introduced in the transition mapping. The FST described in Fig. 2, generates pronunciations for two words അവാൻ /avan/ and അവൾ /aval/. Its parameters are defined in Table 7.

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  $O_2$  [49]. 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.

Mlphon, the tool we introduce is developed using SFST. SFST is programming language for FSTs, written in C++ language [12]. It has a user-friendly python API,<sup>2</sup> 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 [11], German [51], Latin [52] 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.<sup>3</sup> In a code switching context, a language detector may be needed to separate the text and route it to language-specific g2p systems.

### A. ARCHITECTURAL DESCRIPTION

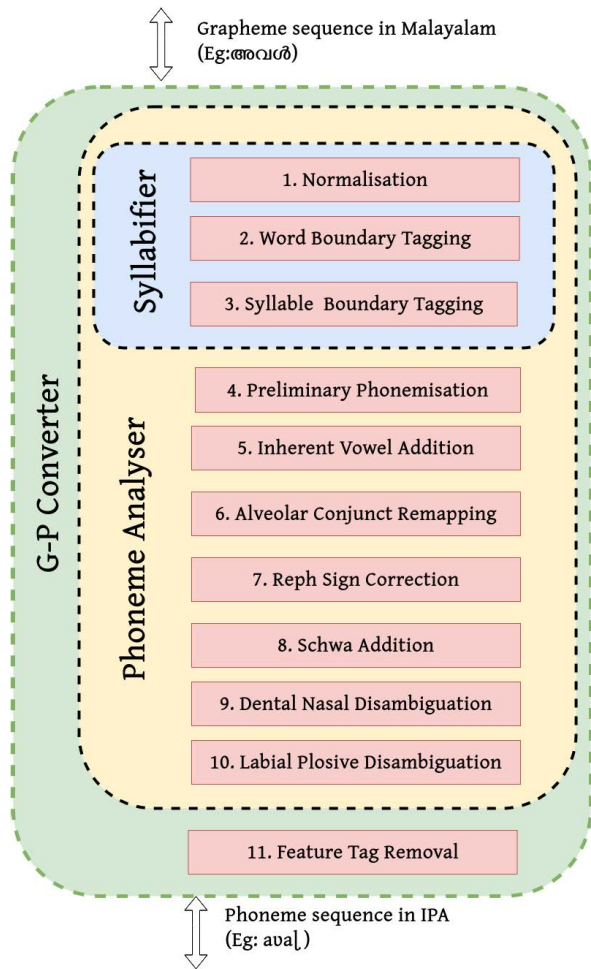
The system architecture of Mlphon is described in Fig. 3. We follow a modular approach in the design of Mlphon. The mapping from Malayalam script to IPA is carried out in eleven steps, where each step represents an FST. In Mlphon, FST parameters are not directly defined. They are instead compiled from 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. 3.

The SFST programs corresponding to the transducers are simplified and described in Algorithms 1 - 4. In the algorithmic description we use the SFST syntax, where ‘|’ indicates the union operation, ‘||’ indicates composition operation, ‘ $\leftarrow$ ’ indicates the mapping of the right hand side input symbol sequence to left hand side output symbol sequence. They are represented in the form:  $D \leftarrow [A] B [C]$ , 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 [53].

<sup>2</sup>SFST Python library: <https://pypi.org/project/sfst/>

<sup>3</sup><https://github.com/kavyamanohar/mlphon>



**FIGURE 3.** The system architecture of Mlphon. Each solid rectangular box 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.

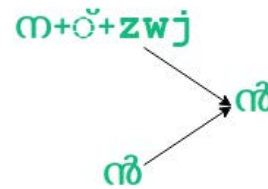
The SFST program is structured as a combination of (i) one-to-one and one-to-many mappings from input symbols to output symbols, (ii) contextual mappings of input symbol sequence to output symbol sequence, and (iii) self mappings where input symbols are passed as such to the output. Additional information 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 X. The functionalities of the individual FSTs are described in sections VI-A1 to VI-A11. Wherever it is essential to communicate the functionality, state transitions in each FST are diagrammatically represented.

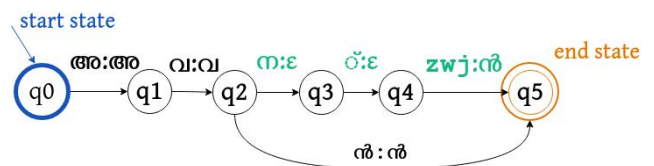
### 1) NORMALISATION

This FST accepts all Malayalam characters and invisible zero width characters.<sup>4</sup> 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.** Two alternate representations of 'അ' 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. It also converts 'അ' /nta/ represented as the sequence 'അ', virama (ഃ), o to 'അ', virama (ഃ), o. Fig. 5 provides an example indicating the state transitions happening in this FST. The procedural description is provided in Algorithm 1.



**FIGURE 5.** Normalisation of the word 'അവാ' indicating the two possible input sequences generating the normalised output. The word final chillu grapheme represented as 'om', zwj is normalised to a common form of single atomic character, 'അ', by passing through states from q2, q3, q4 and q5. If the word were already in normalized form, that character is self mapped as indicated in other transitions.

### 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 1.

### 3) SYLLABLE BOUNDARY TAGGING

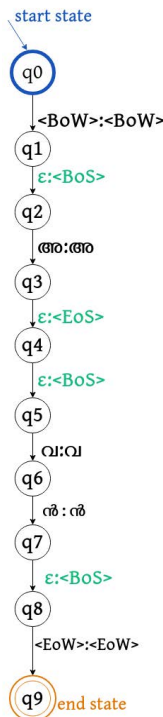
This FST accepts all Malayalam characters, along with word boundary tags. As discussed in section V, some character sequences are invalid according to Malayalam script grammar. The syllabifier FST checks for validity of character sequences to form syllables. An invalid sequence of Malayalam characters 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

<sup>4</sup>Zero Width Joiner: [https://en.wikipedia.org/wiki/Zero-width\\_joiner](https://en.wikipedia.org/wiki/Zero-width_joiner)

**Algorithm 1** Normalisation, Word Boundary Tagging, Syllable Boundary Tagging

```

1: procedure Normalisation
2:   chillunorm_fst: chillu ← base consonant+virama+zwj           ▷ Define a named FST for chillu normalisation
3:   ntanorm_fst: ഐ+ഃ+ഔ ← ഐ+ഃ+ഔ
4:   return chillunorm_fst | ntanorm_fst                           ▷ It is the union of two predefined FSTs
5: end procedure
6: procedure Word Boundary Tagging
7:   return <BoW>+token+<EoW> ← token                           ▷ Insert boundary tags to input word token
8: end procedure
9: procedure Syllable Boundary Tagging
10:  c_v ← consonant + virama
11:  syl_end = [anuswara, visarga, chillu]                          ▷ syl_end is a variable, that can take any value in the list
12:                                     ▷ Four types of character sequences that constitute a syllable is defined in the following lines
13:  Type 1 ← <BoW>+vowel+syl_end?                                  ▷ ? indicates optionaity
14:  Type 2 ← consonant+vowelsign?+syl_end?
15:  Type 3 ← c_v * + consonant                                     ▷ * indicates one or more occurrence
16:  Type 4 ← c_v ? + consonant + ഐ? + virama + <EoW>
17:  syllable ← Type 1 | Type 2 | Type 3 | Type 4                    ▷ A syllable is any of the 4 types
18:  return <BoS> + syllable + <EoS> ← syllable                    ▷ Insert boundary tags to syllables
19: end procedure
  
```



**FIGURE 6.** 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.

for pronunciation analysis. This procedure is explained in Algorithm 1. An example for the insertion of syllable tags is indicated in Fig. 6.

**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 as per tables 2-4 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 ഐ and ഐ to its phoneme with phonetic features is described in Algorithm 2.

**Algorithm 2** Preliminary Phonemisation

```

1: procedure Preliminary Phonemisation
2:   g2p_1: s+<fricative>+<alveolar> ← ഐ
3:   g2p_2: m+<labial>+<nasal> ← ഐ
4:   ...
5:   ...                                     ▷ Basic g2p mappings
6:   return g2p_1 || g2p_2 ...
7:   ...                                     ▷ Composition of basic g2p mappings
8: end procedure
  
```

**5) INHERENT VOWEL ADDITION**

Inherent vowel /a/ is added after consonant phonemes if it is at the end of a syllable position, or it is followed by the anuswara, visarga, or a chillu as described in Algorithm 3.

**6) ALVEOLAR CONJUNCTS REMAPPING**

The most common alveolar consonant clusters in Malayalam, ഐ /nta/ and ഐ /tta/ are constituted from consonants dental nasal ഐ /ṅa/ and alveolar trill ഐ /ra/, the pronunciations of which are strikingly different. Thus the grapheme sequence ഐ /ṅa/, virama, ഐ /ra/ can be mapped to /nta/ instead of /ṅra/. Also the grapheme sequence ഐ /ra/, virama, ഐ /ra/ can be mapped to /tta/ instead of /rra/. This unambiguous



**Algorithm 3** Inherent Vowel Addition, Alveolar Conjuncts Remapping and Reph Sign Correction

```

1: procedure Inherent Vowel Addition
2:   pre_context = consonant+<tags>                                ▷ Define a variable that is a sequence of consonants and tags
3:   post_context = [<chil>, <anuswara>, <visarga>, <EoS>]          ▷ Define a variable that takes any value in the list
4:   return [pre_context] a+<inherentvowel> [post_context] ← [pre_context] ε [post_context]
5: end procedure                                                ▷ <tags> - represent the sequence of phonetic feature tags
6: procedure Alveolar Conjuncts Remapping
7:   tta_fst: [<BoS>,<virama>]+t+<tags>+t+<tags> ← [<BoS>,<virama>]+r+<tags>+r+<tags>
8:   nta_fst: <BoS>+n+<tags>+t+<tags> ← <BoS>+n+<tags>+r+<tags>
9:   return tta_fst || nta_fst                                    ▷ Composition of two FSTs
10: end procedure
11: procedure Reph Sign Correction
12:   return [g,d]+<tags>+<virama>+r+<flapped>+<reph> ← [g,d]+<tags>+<virama>+r+<trill>+<reph>
13: end procedure
    
```

mapping is done by an FST that checks the context and remaps these phonemes as indicated in Algorithm 3.

7) REPH SIGN CORRECTION

If the final consonant in a cluster is the alveolar tap  $\text{ɹ}$  /ra/, its pronunciation gets modified to  $\text{ɹ}$  /ra/ depending on the preceding consonants. The  $\text{ɹ}$  /ra/ sound is retained only if the preceding consonant of the cluster is voiced velar or dental plosive ( $\text{g}$  /ga/ or  $\text{ɖ}$  /ɖa/) as described in Algorithm 3.

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 4. This FST basically disambiguates the function of *virama*. Loan words get adapted to native pronunciation by schwa addition at word ends. eg: ബാങ്ക് /ba:ŋkə/ (*bank*)

9) DENTAL NASAL DISAMBIGUATION

The dental nasal grapheme  $\text{᳚}$  /᳚a/, is pronounced as the alveolar nasal /na/ in the following contexts [48]:

- 1) When a morpheme medial syllable starts in  $\text{᳚}$  and is followed by a vowel sound.  
eg: ആന /a:na/ (*elephant*)  $\text{᳚}$ ാനം /ga:nam/ (*song*),  
അനുജൻ /anujan/ (*younger brother*)
- 2) When  $\text{᳚}$  is the starting character in a consonant cluster followed by  $\text{j}$  /ja/,  $\text{v}$  /va/ or  $\text{m}$  /ma/.  
eg: നന്മ /᳚nam/ (*virtue*), ന്യായം /nja:jam/ (*justice*),  
അന്വേഷണം /anve:᳚a᳚nam/ (*enquiry*)
- 3) When  $\text{᳚}$  is the second character in a consonant cluster, beginning with

$\text{k}$  /ka/  $\text{g}^h$  /g<sup>h</sup>a/  $\text{p}$  /pa/  $\text{m}$  /ma/  $\text{j}$  /ja/ and  $\text{s}$  /sa/.  
eg: വിഘ്നം /vig<sup>h</sup>nam/ (*blockage*), സ്വപ്നം /svapnam/  
(*dream*), പ്രശ്നം /pra᳚᳚nam/ (*issue*), സ്നേഹം /sne:᳚am/  
(*love*)

These three rules are implemented by identifying the context of appearance of  $\text{᳚}$  in terms of the surrounding

consonants and syllable boundaries etc. as described in the Algorithm 4.

10) LABIAL PLOSIVE DISAMBIGUATION

The unvoiced aspirated labial plosive grapheme  $\text{ɸ}$  /p<sup>h</sup>a/ is used to represent the labiodental fricative /f/ in non-native words. On analysing a corpus of 100k most frequent Malayalam words [19], only 6% of words that contained the letter  $\text{ɸ}$  were native. All those native words had the letter  $\text{ɸ}$ , either preceded by the letter  $\text{᳚}$  or followed by  $\text{᳚}$ . This graphemic context is used as the parameter to determine the word origin and remap fricative to plosive as described in Algorithm 4.

11) FEATURE TAG REMOVAL

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

**B. SYLLABIFIER FST**

The composition of the series of FSTs from VI-A1 to VI-A3 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 modeling for ASR as described in section X. An illustration of this module accepting Malayalam text as input and generating output with syllable boundary tags is shown in Fig. 7.

If the token passed to the syllabifier is  $\text{അവൾ}$ , it returns the syllabified string  $\text{<BoS>അ<EoS><BoS>വൾ<EoS>}$ . The python interface to the FST for syllabification, parses the boundary tags and returns the sequence of syllables.

**C. PHONEME ANALYSER FST**

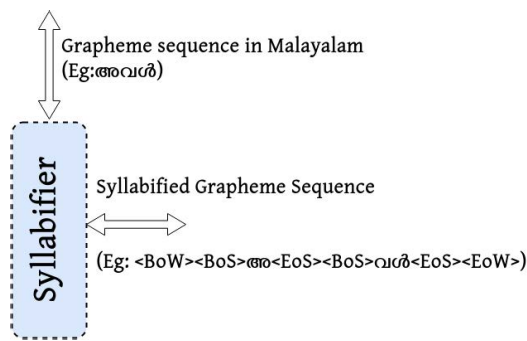
Phoneme analyser FST is compiled as a composition of 10 FSTs described in sections VI-A1 to VI-A10 and indicated in Fig. 3 of Mlphon architecture. This FST accepts a grapheme sequence as input and returns phoneme sequence, tagged with their phonetic features. If the token  $\text{അവൾ}$

**Algorithm 4** Schwa Addition, Dental Nasal Disambiguation, Labial Plosive Disambiguation

```

1: procedure Schwa Addition
2:   schwa_1: ə+<schwa>+<EoS> ← u+<v_sign>+<virama>+<EoS>           ▷ Define named FSTs for schwa addition
3:   schwa_2: ə+<schwa>+<EoS> ← <virama>+<EoS>
4:   return schwa_1 || schwa_2                                       ▷ Return the composition of two FSTs
5: end procedure
6: procedure Dental Nasal Disambiguation
7:   nasalrule_1: <BoS>+n+<alveolar>+<virama>+[j,v,m] ← <BoS>+ŋ+<dental>+<virama>+[j,v,m]   ▷ Define named
   FSTs
8:   nasalrule_2: <EoS>+<BoS>+n+<alveolar>+[vowel] ← <EoS>+<BoS>+ŋ+<dental>+[vowel]
9:   nasalrule_3: [k,gh,p,m,ŋ,s]+<tags>+<virama>n+<alveolar> ← [k,gh,p,m,ŋ,s]+<tags>+<virama>ŋ+<dental>
10:  return nasalrule_1 || nasalrule_2 || nasalrule_3                 ▷ Return the composition of three FSTs
11: end procedure
12: procedure Labial Plosive Disambiguation
13:  fa_1: <BoW>+<BoS>ph+<plosive>+a+<EoS>+<EoW> ← <BoW>+<BoS>+f+<fricative>+a+<EoS>+<EoW>
14:  fa_2: s+<fricative>+<alveolar>+<virama>ph+<plosive> ← s+<fricative>+<alveolar>+<virama>+f+<fricative>
15:  fa_3: ph+<plosive>+a+<EoS>+<BoS>+l ← f+<fricative>+a+<EoS>+<BoS>+l
16:  return fa_1 || fa_2 || fa_3                                     ▷ Return the composition of three FSTs
17: end procedure

```



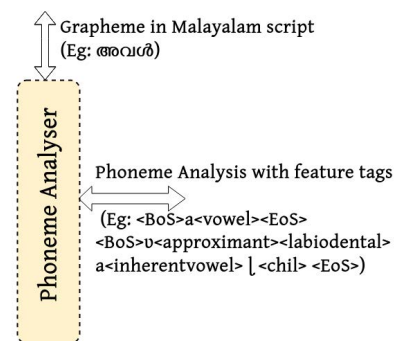
**FIGURE 7.** Syllabifier performing syllabification on the word അവൾ. Boundary tags for words and syllables are demonstrated.

is passed to the phoneme analyser FST, it returns the string <BoS>a<vowel><EoS><BoS>v<approximant><labiodental>a<inherentvowel>|<chil><EoS> as illustrated in Fig. 8. This module can play crucial role in the context of linguistic learning providing pronunciation information regarding the graphemes in a word.

Fig. 9 illustrates the state transitions and the insertion of tags in the phoneme analyser FST when input tokens passed are: അവൾ and അവൾ. The python interface of Mlphon utilizes this FST to analyse the Malayalam word and return the sequence of phonemes and phonetic feature tags like place and manner of articulation.

**D. 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. 3. All the FSTs previously discussed are bidirectional. However the bidirectionality property is particularly useful when there



**FIGURE 8.** 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.

is need to convert graphemes to phonemes and vice-versa. Fig. 10 demonstrates an input and output symbol sequence of G-P Converter FST.

This FST, parses the words അവൾ and അവൾ in analysis mode as shown in the Fig. 11 (i). When operated in generate mode, it converts a valid phoneme sequence into graphemes. For example, in generate mode, it can parse the inputs av| and avan as shown in Fig. 11 (ii).

**E. 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 minimized FSTs which are very much memory optimized [54]. 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

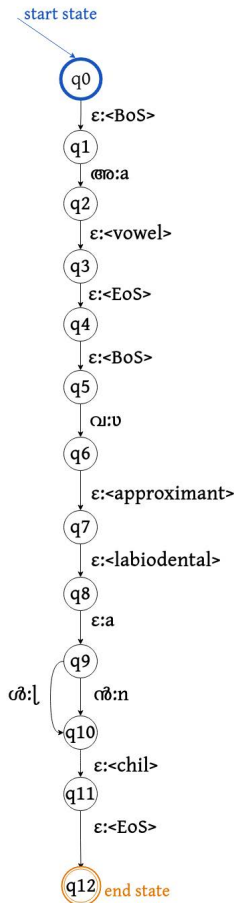


FIGURE 9. Phoneme analyser FST, showcasing grapheme to phoneme conversion on the word അവാൾ.

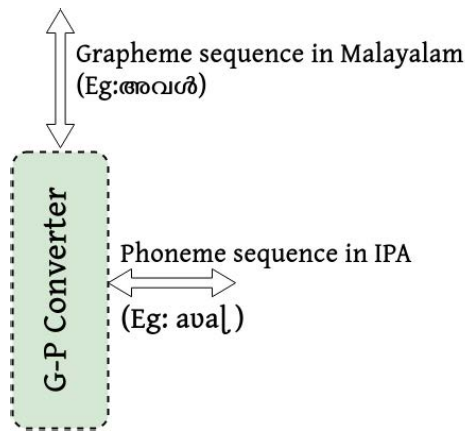


FIGURE 10. G-P Converter FST, performing phoneme analysis on the word അവാൾ.

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

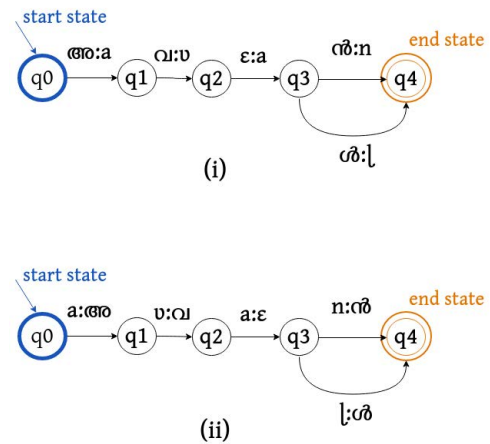


FIGURE 11. G-P converter FST performing (i) grapheme to phoneme conversion on the words അവാൾ and അവൻ in analysis mode and (ii) phoneme to grapheme conversion on `ava|` and `avan` in generate mode.

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 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.

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

optional arguments:
-h, --help
-s, --syllabify
-a, --analyse
-p, --tophoneme
-g, --tographeme
-i INFILE, --input INFILE
-o OUTFILE, --output OUTFILE
-v, --verbose
```

Listing 1. Command line utility for Mlphon library.

### VII. INTRINSIC EVALUATIONS AND DISCUSSIONS

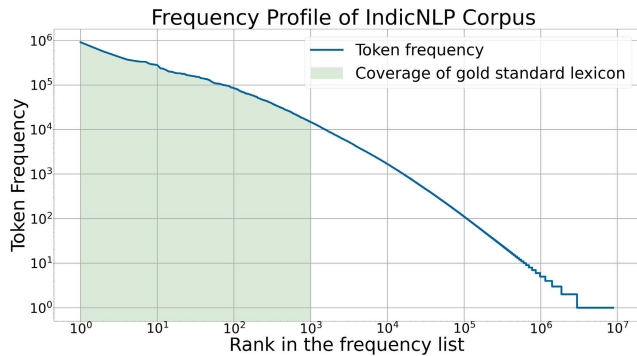
Evaluating a script analysis toolkit like Mlphon is not straight forward due the absence of any baseline ground truth linguistic resource. A gold standard with manually annotated data, which can serve as a reference is an important part of any quantifiable evaluation [11]. A gold standard for g2p conversion 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 [11], used for creating a gold standard annotations for Turkish morphological analyser.

**A. DESIGN OF GOLD STANDARD LEXICON**

The lexical entries in gold standard lexicon are chosen from the IndicNLP Corpus<sup>5</sup> [19] 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 [46], [48]. 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.



**FIGURE 12.** The lexical entries of gold standard lexicon is chosen from the most frequent thousand words from the IndicNLP corpus [19] such that these words cover 26% of the 167.4 million tokens present in this corpus.

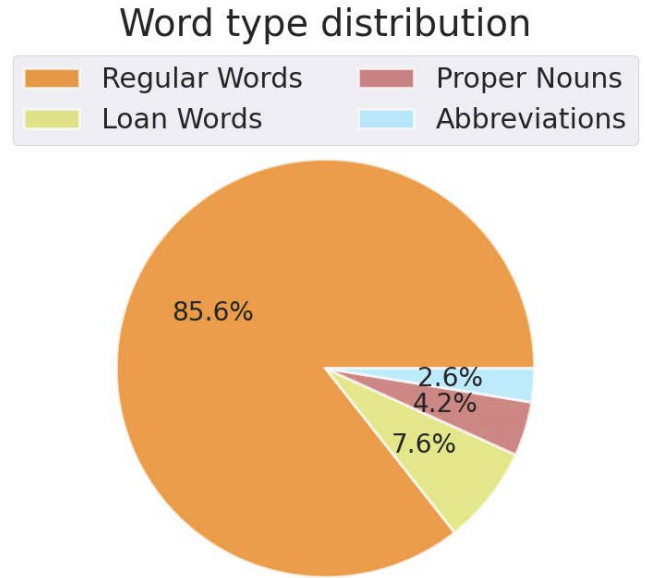
The lexical entries in the gold standard lexicon constitutes 26% of the total number of tokens in the said corpus according to the computation shown in (1). The frequency profile in Fig. 12 illustrates this. The coverage of tokens in the gold standard lexicon with respect to the corpus is computed as:

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

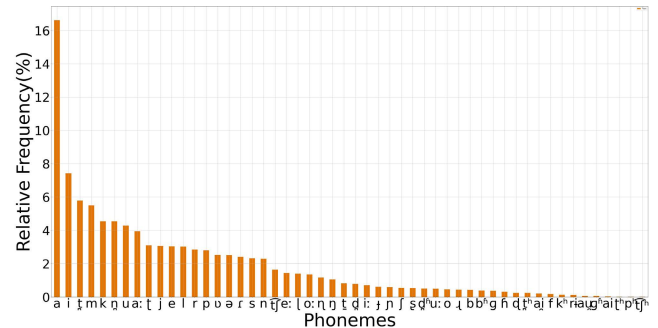
<sup>5</sup>[https://github.com/AI4Bharat/indicnlp\\_corpus](https://github.com/AI4Bharat/indicnlp_corpus)

$$\begin{aligned} &\approx \frac{44321494 \times 100}{167.4\ million} \\ &\approx 26\% \end{aligned} \tag{1}$$

The gold standard lexicon covers many regular words, loan words, proper nouns and abbreviations as per the distribution illustrated in Fig. 13.



**FIGURE 13.** Distribution of word types in gold standard lexicon.



**FIGURE 14.** Phoneme diversity analysis in gold standard lexicon.

A phoneme diversity analysis of the gold standard lexicon was performed and plotted in Fig. 14. The relative frequency of phonemes in gold standard lexicon follows the same pattern as previously reported values of phoneme diversity in Malayalam speech corpora [55].

**B. 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 8 and analysed in detail in section VII-B3.



**TABLE 8. 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.	-
സി.ഐ.ഡി	സി.ഐ.ഡി.		Deletion

1) 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 [56]. 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%

2) 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%

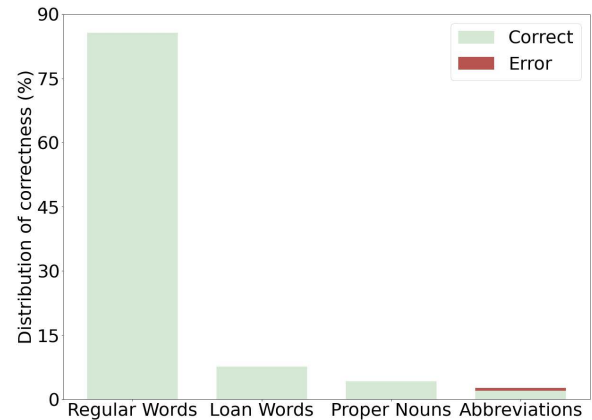
3) ERROR ANALYSIS

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.

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. 15.

**C. GRAPHEME TO PHONEME CONVERSION**

The grapheme to phoneme conversion of Mlphon is evaluated, comparing its output with gold standard phoneme transcriptions. Evaluation involves phoneme level alignment



**FIGURE 15. Distribution of syllabification errors in different word types in gold standard lexicon.**

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<sup>6</sup> to perform the same. A sample of gold standard transcriptions with the phoneme sequence output provided by Mlphon is shown in Table 9.

**TABLE 9. Comparing the g2p transcription provided by Mlphon with gold standard reference.**

Word	Reference	Mlphon	Error
ഈ	i:	i:	-
ഉന്നത	u n n a t̪ a	u n n a t̪ a	-
എന്ന	e n̪ n̪ a	e n̪ n̪ a	-
ഒരു	o r u	o r u	-
കഫേ	k a f e:	k a f e:	-
തോമസ്	t̪ o: m a s	t̪ o: 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 o: t̪ t̪ o:	f o: t̪ t̪ o:	-
സി.ഐ.ഡി	s i a j̪ d̪ i	s i a j̪ d̪ i	-

1) ACCURACY OF g2p CONVERSION

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 in Fig. 16. For all phonemes other than those listed in Table 10, the accuracy, precision, recall, and F1 scores were computed to be 100%.

Except for the റ 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.16 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

<sup>6</sup>Kaldialign library <https://pypi.org/project/kaldialign/>

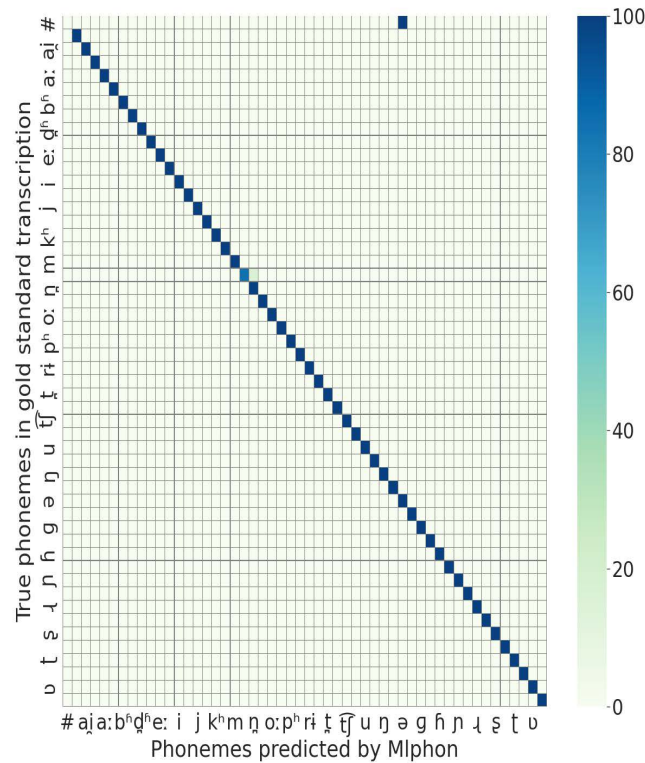
**TABLE 10. Precision, Recall and F1 Scores of phoneme transcription by Mlphon. For all other phonemes, these metrics are evaluated to be 100%.**

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

100k common Malayalam words. According to the confusion matrix in Fig. 17, the transcription accuracy of  $\text{aɳ}$  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%

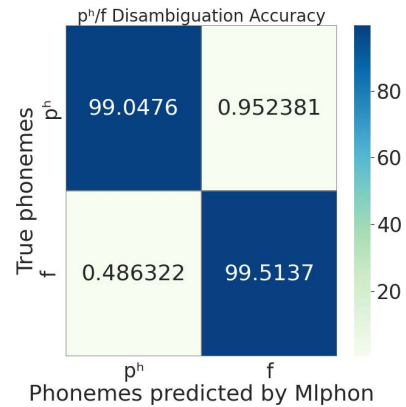


**FIGURE 16. Confusion matrix comparing Mlphon transcription with gold standard transcription. The values are normalized and represented as percentage.**

2) 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

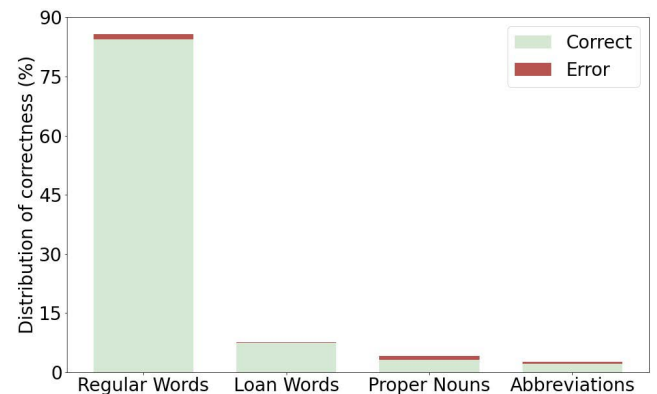


**FIGURE 17. In an evaluation space of 100k tokens, we computed the accuracy of transcribing  $\text{aɳ}$ . The two possible pronunciations /f/ and /p^h/ were accurately identified in more than 99% of the cases as shown in this confusion matrix.**

- Phonemes Inserted: 12
- Phonemes Substituted: 25
- Phoneme Error Rate = 0.55%

3) ERROR ANALYSIS

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. 18.



**FIGURE 18. Distribution of g2p errors in different word types in gold standard lexicon.**

The correction of substitution and insertion errors involve morphologically analysing the 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%

**VIII. EVALUATIONS AND COMPARISON WITH OTHER TOOLS**

The development of a pronunciation lexicon for use in speech tasks like ASR is one of the crucial applications of a g2p

**TABLE 11. A qualitative linguistic comparison between the lexicons produced by Mlphon and freely accessible automated tools.**

No.	Word	Unified Parser	Espeak	Mlphon	Remarks
1	കല്പന	kalpana	kɛlbənɐ	kalpana	The third phoneme produced by Unified Parser is different in rows 1 and 2, which should have been same as produced by Espeak and Mlphon
2	കൽപന	kalwpana	kɛlbənɐ	kalpana	
3	അവൻ	awan	ɛvənɪ	avanə	Espeak and Mlphon ensures word end vowel ( <i>Samvruthokaram</i> ) is rightly inserted, corresponding to <i>virama</i> at word ends. But Unified Parser does not handle this.
4	നിനക്ക്	ninakk	ninək:i	ninakkə	Only Mlphon disambiguates the dental (ɳ) and alveolar nasal (n) pronunciations of റ.
5	കരന്റ്	karxanrx	kərəndi	karantə	Unified Parser fails to contextually change the pronunciation of റ, while Espeak and Mlphon handles this correctly

conversion tool. Of 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.

All of these tools use different phoneme alphabets. Additionally, the g2p mapping criteria vary. Unified Parser and Mlphon both aim to turn graphemes into phonemes. Espeak aims to create allophones of phonemes, which means that a particular phoneme may be represented by multiple phonetic symbols in Espeak, depending on where it appears in 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 11. On analysing these lexicons, following observations can be made:

- 1) Unified Parser ignores all other contextual rules discussed in section VI, except inherent vowel deletion in the context of *virama* and other signs.
- 2) Espeak implements most of the contextual rules discussed in section VI, 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 below.

#### A. ASR EXPERIMENTAL SETUP

Kaldi toolkit [57] is used for our experiments on ASR. We split the openly available transcribed Malayalam speech

corpora from various sources [58], [59], [60], into training and test datasets, ensuring non-overlapping speakers and speech transcripts as listed in Table 12. This amounts to 19 hours of speech data for training and 2 hours of speech data for testing. Apart from the transcripts of speech which amount to 7924 unique sentences, we have utilized the curated collection of text corpus published by SMC [61] amounting to 205k unique sentences for language modeling. After combining these, we explicitly removed all sentences that are present in our test audio transcripts. Bigram language model is prepared on this language modeling corpus using SRILM toolkit [62]. The vocabulary of our ASR is 69k words and the lexicons are prepared using Unified Parser, Espeak and Mlphon.

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 standard Mel frequency cepstral coefficients (MFCCs) with delta and double delta coefficients computed over a window (Hamming) size of 25 ms with an overlap of 10 ms for GMM-HMM monophone and triphone models. The acoustic modeling begins with flat start monophone model followed by context dependent triphone acoustic modeling. Then speaker independent linear discriminant analysis (LDA) to reduce the feature space dimensionality and maximum likelihood linear transform (MLLT) are performed. It is followed by triphone speaker adaptive training (SAT).

Phone alignments from final triphone model are used for Kaldi chain acoustic modeling. It is implemented using time delay neural networks (TDNNs) [63]. 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 [64] 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. Neural acoustic model is trained on a single NVIDIA Tesla T4 GPU.

#### 1) 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 (I), deleted (D) and substituted (S) in the

**TABLE 12.** Details of Speech data sets used in our experiments.

Name	Corpus	#Speakers	#Utterances	Duration (minutes)	Type	Environment
1	Indic TTS, IITM [59]- Train	2	8601	838	Read, Formal	Studio
2	Open SLR Malayalam [60] - Train	37	3346	287	Read, Formal	Studio
T1	Open SLR Malayalam [60] - Test	7	679	48	Read, Formal	Studio
T2	Festvox IIITH [58] - Test	1	1000	98	Read, Formal	Studio

predicted speech transcript when compared to the ground truth transcript.

The 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 modeling. We evaluate our ASR models on two different test datasets namely, T1 (14% OOV) and T2 (1% OOV) derived respectively from OpenSLR [60] and Festvox IIITH [58] 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 13. The best WERs on T1 and T2 are 34.6% and 9.6% respectively, and they are both given by Mlphon lexicon.

**TABLE 13.** Comparing WER (%) obtained in Malayalam ASR experiments with lexicons created using the proposed tool, Mlphon, and other openly available tools.

Acoustic Models	T1 (14% OOV)			T2 (1% OOV)		
	Unified Parser	Espeak	Mlphon	Unified Parser	Espeak	Mlphon
Monophone	60.9	<b>58.4</b>	58.7	25.0	<b>20.9</b>	21.8
Triphone	49.9	48.3	<b>47.4</b>	21.0	17.5	<b>17.1</b>
Triphone (LDA)	43.8	<b>41.2</b>	43.7	18.4	14.3	<b>13.9</b>
Triphone(SAT)	43.6	41.2	<b>41.0</b>	14.3	11.1	<b>10.6</b>
TDNN	35.7	34.9	<b>34.6</b>	10.7	9.7	<b>9.6</b>

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 section VI.

This analysis is an indicator of the importance of precise g2p conversion required for speech tasks. Also as we demonstrate in the following section, Mlphon has the additional advantage of high word processing speed, while creating pronunciation lexicons.

Although there have been previously published works on ASR for continuous Malayalam speech [27], [65], [66], 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 14.

**TABLE 14.** 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. [27]	29k	8	34.2
Lavanya et al. [65]	-	-	34.4
Lekshmi et al. [66]	-	-	10.0
Proposed tool	69k	14	<b>34.6</b>
Proposed tool	69k	1	<b>9.6</b>

## IX. CREATING LARGE VOCABULARY PRONUNCIATION LEXICONS

Apart from small pronunciation lexicons created manually or semi-automatically for some specific experiments as discussed in section II, there is exists no openly available pronunciation lexicons for Malayalam. To bridge this gap we publish a large vocabulary pronunciation lexicon for Malayalam, automatically created using Mlphon.

These lexicons consist of different categories of words as described in Table 15. The tokens in common words pronunciation lexicon are extracted from a general domain text corpus of 167 million types covering the fields of business, entertainment, sports, technology etc. as described in Indic NLP dataset [19]. 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.



**TABLE 15. 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

Multiple pronunciations of the same word are provided whenever applicable.

Table 16 gives an excerpt from different categories of pronunciation lexicons. These lexicons are publicly available for download and usage under CC-BY-SA License.

**TABLE 16. An excerpt from pronunciation lexicons.**

Lexicon	Word	Phonemised transcription	Syllabified transcription
Common words	ഒരു	o r u:	o ru
	ഈ	i:	i:
	എന്ന	e n n a	e nna
	തന്നെ	t a n n e	ta nne
	പഠഞ്ഞു	p a r a n n u	pa ra nnu
	എന്നാൽ	e n n a:l	e nna:l
Verbs	അകലുക	a k a l u k a:	a ka lu ka
	അഞ്ചുക	a n n c h u k a	a n n c h u ka
	അടക്കുക	a t a k k u k a	a ta kku ka
Nouns	അടക്ക്	a t a k k e	a ta kka
	അടങ്കൽ	a t a n n k a l	a ta n n kal
	അടുത്തു	a t a t t u t t u r a	a ta t t t u tu ra
	അടുവാറ്റി	a t a t t u v a:t t t i	a ta t t t u va: t t t i
Proper Nouns	അകബർ	a k b a r	a kbar
	അക്ഷയ	a k s a j a	a ksa ja
	അക്ഷര	a k s a r a	a ksa ra
	അഖില	a k h i l a	a k h i la
	അഖിലൻ	a k h i l a n	a k h i lan
Loan words	അക്കാദമി	a k k a: d a m i	a kka: da mi
	അക്കൗണ്ട്	a k k a u n t t e	a kka u n t e
	അക്വേറിയം	a k v e: r i j a m	a kve: ri jam
	അങ്കിൾ	a n n k i l	a n n ki l

**A. COMPARISON OF PROCESSING SPEED**

Word processing speed (WPS) is one indicator of a g2p algorithm’s effectiveness [13]. The WPS for the applications, Unified Parser [4], Espeak<sup>7</sup> 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 [19] as per (2). 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 17. This faster processing speed of Mlphon makes it particularly suitable for integration with other real time NLP applications.

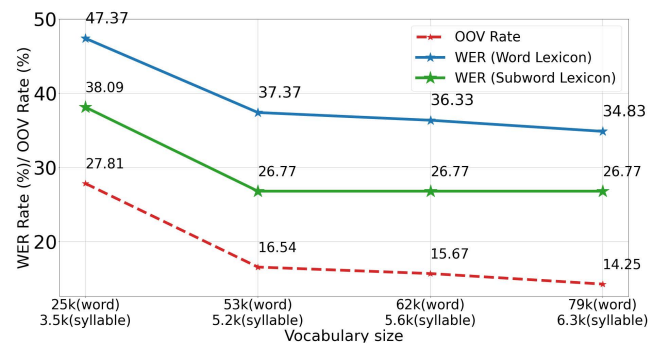
$$WPS = \frac{100k [words]}{Processing\ time [minutes]} \quad (2)$$

The reason for the time efficiency while using Mlphon can be attributed to the computationally fast determined and

<sup>7</sup>Using Phonemizer library <https://pypi.org/project/phonemizer/>

**TABLE 17. Comparison of the WPS of the proposed tool Mlphon with other openly available tools.**

Tool	WPS (Words per minute)
Unified Parser	42
Espeak	6722
<b>Mlphon</b>	<b>69142</b>



**FIGURE 19. Plot showing WER obtained in Malayalam ASR experiments with word based and syllable subword based language models and lexicons. Word vocabulary sizes are indicated on x-axis.**

minimised FSTs [54], upon which Mlphon is built. Unified Parser is prohibitively slow due to the additional memory management requirement.<sup>8</sup> The measurement of grapheme-to-phoneme conversion speed was performed on a PC workstation with 2 × AMD CPU @ 2.250 GHz and 4 GB of RAM.

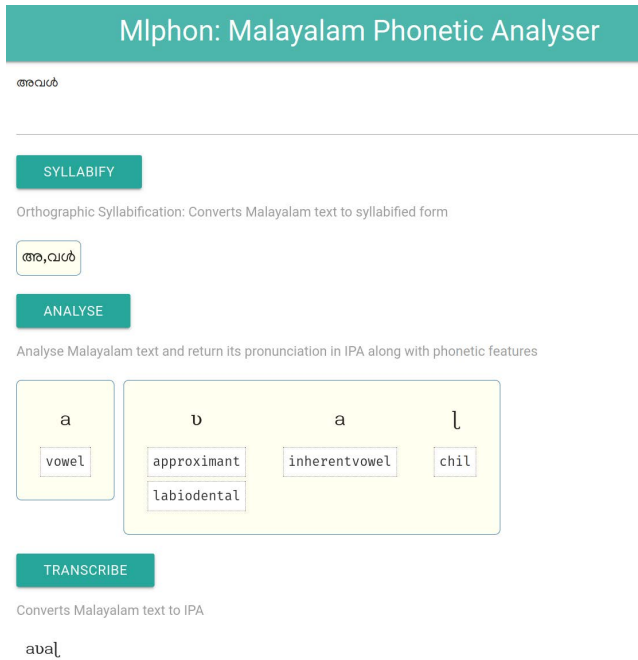
**X. APPLICATIONS**

In this section we describe some potential application of Mlphon.

**A. SYLLABLE BASED LANGUAGE MODELING**

The syllabification module in Mlphon is a standalone unit that splits words to syllables. It has been demonstrated in literature that subword based models are better in capturing language features for morphologically complex languages [18]. Syllables serve as a good choice of subwords for practical applications including automatic speech recognition [67] that takes care of OOV scenarios. Orthographic syllable units have proven to be more effective in statistical machine translation, than other basic units (word, morpheme and character) when trained over small parallel corpora [17]. Mlphon can be employed in various applications that require syllable level language modeling.

<sup>8</sup>Solution for segmentation fault error suggested in the discussion forum <https://groups.google.com/g/indicnlp/c/YUuHfr3YSug/m/xcflHJTkaQAJ>



**FIGURE 20.** Web interface for Mlphon. Features of syllabification, phonetic analysis and IPA transcription are shown.

As an example, we demonstrate the usage of syllable based lexicons and language models on ASR task. We use the same experimental setup as described in section VIII. Evaluation is done on OpenSLR test set where OOV is higher. To evaluate syllable based language models and lexicons, we use word based lexicons and language models as baseline. The Fig. 19,

shows how the WER of syllable based lexicons and language models are consistently better than word based ones, while incrementally increasing the vocabulary size. Each subword lexicon is built by including all the syllables present in corresponding word lexicon. For example the first subword lexicon has 3.5k syllables as entries, obtained by syllabifying every entry in corresponding word lexicon with 25k entries. It is observed that syllable based ASR performed much better than word based ones, as it recovered many OOV words by reconstructing words by concatenating syllables.

**B. ASSISTED PRONUNCIATION LEARNING**

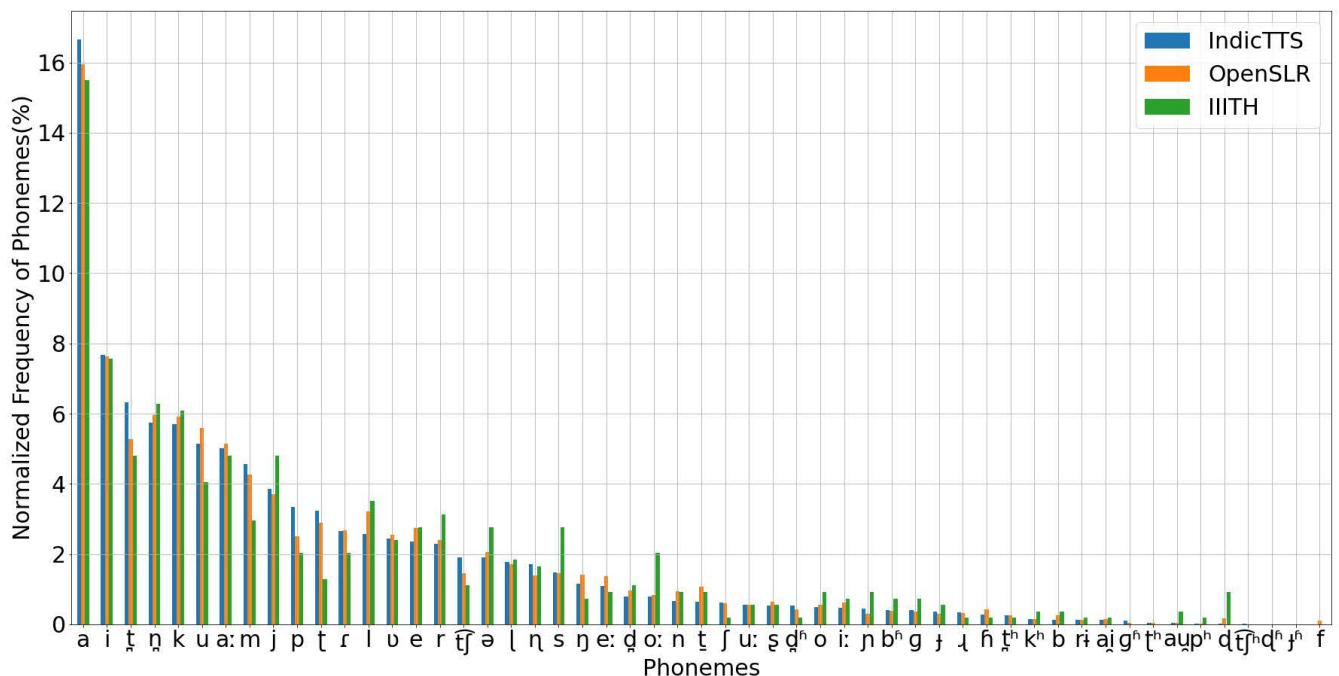
It is important for a new script learner to understand the pronunciations correctly and get a comprehensive idea of phonetic features of the text. Mlphon provides phonetic feature tags corresponding to every phoneme. A web interface<sup>9</sup> has been developed for user friendly access to Mlphon features. As demonstrated in Fig. 20, 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.

**C. 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 modeling [68]. We transcribed the speech corpus

<sup>9</sup>Mlphon Web Interface <https://phon.smc.org.in/>



**FIGURE 21.** Phonemic diversity analysis of various speech corpora used in ASR Experiments. It indicates relative frequency of each phoneme.

transcript to phonemised text using Mlphon. The phoneme diversity of the resulting text is then analysed. The graph in Fig. 21 illustrates the phonemic richness of the corpora used in the ASR experiments described in Section VIII. 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 [59] corpus while it is /ŋ/ in OpenSLR [60] and Festvox IIITH [58] corpora. The statistical analysis of phonemes could potentially be used to design corpora with phonemically balanced content [69].

#### D. 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 18 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 section VIII. However, errors which do not violate the script grammar rules can not be detected by Mlphon.

**TABLE 18.** 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 [59]	1013	1.2%
OpenSLR [60]	83	0.3%
Festvox IIITH [58]	22	0.3%
Indic Speech [70]	4337	4.3%

## XI. CONCLUSION

In this article we presented the requirement analysis, design and the development of a knowledge based computational linguistic tool, Mlphon, using FSTs. The syllabification of graphemes as well as phonemes, phonetic feature analysis and bidirectional grapheme-phoneme conversions that can be performed by Mlphon has applications in speech and linguistic research. The syllabification and grapheme-phoneme conversion capability of Mlphon is evaluated against a gold standard lexicon. Mlphon performs syllabification with an accuracy of 99% and syllable error rate of 0.62% on gold standard lexicon. On grapheme to phoneme conversion task, a phoneme recognition accuracy of 99% with a phoneme error of 0.55% is observed. The pronunciation lexicons generated with Mlphon has improved performance in terms of WER, than other automated tools when employed in ASR task as described in this article. Mlphon has a high word processing speed when compared to other tools for g2p mapping.

The pronunciation lexicon with 100k common words, verbs, nouns and foreign language words in phonemised and syllabified forms published along this work is the first of its kind in Malayalam. Mlphon that takes care of the script specific contextual rules for phonemic analysis serve as a useful resource for various NLP tasks including ASR, TTS, syllabification for language modeling, phonemic diversity analysis, assisted pronunciation learning and text sanity check as demonstrated in this article.

## APPENDIX A OPEN RESOURCES PUBLISHED WITH THIS WORK

- 1) Companion Website for this paper
- 2) Source code of Mlphon
- 3) Mlphon Python library
- 4) Malayalam Pronunciation Lexicons
- 5) Malayalam ASR experiments using Kaldi

## REFERENCES

- [1] F. Coulmas, *The Blackwell Encyclopedia of Writing Systems*. Hoboken, NJ, USA: Wiley, 1999. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118932667.ch7>
- [2] D. Crystal, *A Dictionary of Linguistics and Phonetics*, vol. 30. Hoboken, NJ, USA: Wiley, 2011.
- [3] D. R. Mortensen, S. Dalmia, and P. Littell, "Epitrans: Precision G2P for many languages," in *Proc. 11th Int. Conf. Lang. Resour. Eval. (LREC)*. Miyazaki, Japan, May 2018, pp. 1–5. [Online]. Available: <https://aclanthology.org/L18-1429>
- [4] A. N. N. L. Baby and A. L. H. A. Thomas Murthy, "A unified parser for developing Indian language text to speech synthesizers," in *Text, Speech, and Dialogue*, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds. Cham, Switzerland: Springer, 2016, pp. 514–521.
- [5] W. Bright, "A matter of typology: Alphasyllabaries and abugidas," *Written Lang. Literacy*, vol. 2, no. 1, pp. 45–55, 1999. [Online]. Available: <https://www.jbe-platform.com/content/journals/10.1075/wll.2.1.03bri>
- [6] R. M. Kaplan and M. Kay, "Regular models of phonological rule systems," *Comput. Linguistics*, vol. 20, no. 3, pp. 331–378, 1994. [Online]. Available: <https://www.aclweb.org/anthology/J94-3001>
- [7] L. Karttunen and K. R. Beesley, *Two-Level Rule Compiler*. Norwalk, CT, USA: Xerox Corporation, Palo Alto Research Center, 1992. [Online]. Available: <https://web.stanford.edu/~laurikl/book2software/twolc.pdf>
- [8] K. Oflazer and S. Inkelas, "The architecture and the implementation of a finite state pronunciation lexicon for Turkish," *Comput. Speech Lang.*, vol. 20, no. 1, pp. 80–106, Jan. 2006, doi: [10.1016/j.csl.2005.01.002](https://doi.org/10.1016/j.csl.2005.01.002).
- [9] T. Anberbir, M. Gasser, T. Takara, and K. D. Yoon, "Grapheme-to-phoneme conversion for Amharic text-to-speech system," in *Proc. Conf. Hum. Lang. Technol. Develop.*, Bibliotheca, Alexandrina, 2011, pp. 1–6.
- [10] S. Thottingal, "Finite state transducer based morphology analysis for Malayalam language," in *Proc. 2nd Workshop Technol. MT Low Resource Lang.*, Dublin, Ireland, Aug. 2019, pp. 1–5. [Online]. Available: <https://aclanthology.org/W19-6801>
- [11] A. Kayabaş, H. Schmid, A. E. Topcu, and O. Kiliç, "TRMOR: A finite-state-based morphological analyzer for Turkish," *Turkish J. Elect. Engineering Comput. Sci.*, vol. 27, no. 5, pp. 3837–3851, 2019.
- [12] H. Schmid, "A programming language for finite state transducers," in *Proc. FSMNL*, vol. 4002, 2005, pp. 308–309.
- [13] P. Klosowski, "A rule-based grapheme-to-phoneme conversion system," *Appl. Sci.*, vol. 12, no. 5, p. 2758, 2022. [Online]. Available: <https://www.mdpi.com/2076-3417/12/5/2758>
- [14] K. Manohar, A. Jayan, and R. Rajan, "Quantitative analysis of the morphological complexity of Malayalam language," in *Proc. Int. Conf. Text, Speech, Dialogue*. Cham, Switzerland: Springer, 2020, pp. 71–78, doi: [10.1007/978-3-030-58323-1\\_7](https://doi.org/10.1007/978-3-030-58323-1_7).
- [15] G. B. Kumar, K. N. Murthy, and B. Chaudhuri, "Statistical analysis of Telugu text corpora," *Int. J. Dravidian Linguistics*, vol. 36, no. 2, pp. 71–99, 2007.
- [16] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.

- [17] A. Kunchukuttan and P. Bhattacharyya, "Orthographic syllable as basic unit for SMT between related languages," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, Nov. 2016, pp. 1912–1917. [Online]. Available: <https://www.aclweb.org/anthology/D16-1196>
- [18] P. Smit, S. Virpioja, and M. Kurimo, "Advances in subword-based HMM-DNN speech recognition across languages," *Comput. Speech Lang.*, vol. 66, Mar. 2021, Art. no. 101158. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0885230820300917>
- [19] A. Kunchukuttan, "Ai4Bharat-indicNLP corpus: Monolingual corpora and word embeddings for indic languages," 2020, *arXiv:2005.00085*.
- [20] K. Lenzo. (2007). *The CMU Pronouncing Dictionary*. [Online]. Available: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>
- [21] T. Schultz and T. Schlippe, "GlobalPhone: Pronunciation dictionaries in 20 languages," in *Proc. 9th Int. Conf. Lang. Resour. Eval. (LREC)*. Reykjavik, Iceland, May 2014, pp. 337–341. [Online]. Available: [http://www.lrec-conf.org/proceedings/lrec2014/pdf/1212\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2014/pdf/1212_Paper.pdf)
- [22] F. D. Vriend, N. Castell, J. Giménez, and G. Maltese, "LC-STAR: XML-coded phonetic lexica and bilingual corpora for speech-to-speech translation," in *Proc. Workshop Multilingual Lexical Databases*, 2004.
- [23] A. Ali. (Mar. 2017). *Arabic Speech Recognition Pronunciation Dictionary LDC2017L01. Web Download*. Philadelphia: Linguistic Data Consortium, 2017. [Online]. Available: <https://catalog.ldc.upenn.edu/LDC2017L01>
- [24] X. Huang, X. Jin, Q. Li, and K. Zhang, "On construction of the ASR-oriented Indian English pronunciation dictionary," in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, May 2020, pp. 6593–6598. [Online]. Available: <https://aclanthology.org/2020.lrec-1.812>
- [25] A. Gutkin, L. Ha, M. Jansche, K. Pipatsrisawat, and R. Sproat, "TTS for low resource languages: A Bangla synthesizer," in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, Portoroz, Slovenia, 2016, pp. 2005–2010. [Online]. Available: [http://www.lrec-conf.org/proceedings/lrec2016/pdf/286\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2016/pdf/286_Paper.pdf)
- [26] C. Kurian, "Speech database and text corpora for Malayalam language automatic speech recognition technology," in *Proc. Conf. Oriental Chapter Int. Committee Coordination Standardization Speech Databases Assessment Techn. (O-COCOSDA)*, Oct. 2016, pp. 7–11, doi: [10.1109/ICSODA.2016.7918975](https://doi.org/10.1109/ICSODA.2016.7918975).
- [27] G. Deekshitha, K. R. Sreelakshmi, B. P. Babu, and L. Mary, "Development of spoken story database in Malayalam language," in *Proc. 4th Int. Conf. Elect. Energy Syst. (ICEES)*, Feb. 2018, pp. 530–533.
- [28] K. R. Lekshmi, V. S. Jithesh, and E. Sherly, "Malayalam speech corpus: Design and development for dravidian language," in *Proc. 5th Workshop Indian Lang. Data: Resour. Eval. (WILDRE)*, Marseille, France, May 2020, pp. 25–28. [Online]. Available: <https://www.aclweb.org/anthology/2020.wildre-1.5>
- [29] A. W. Black, K. Lenzo, and V. Págel, "Issues in building general letter to sound rules," in *Proc. 3rd ESCA/COCOSDA Workshop (ETRW) Speech Synth.*, 1998, pp. 1–4.
- [30] E. Fosler-Lussier, Y. He, P. Jyothi, and R. Prabhavalkar, "Conditional random fields in speech, audio, and language processing," *Proc. IEEE*, vol. 101, no. 5, pp. 1054–1075, May 2013.
- [31] F. Yvon, "Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks," 1996, *arXiv:cmp-1g/9608006*.
- [32] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, no. 5, pp. 434–451, 2008.
- [33] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," 2015, *arXiv:1506.00196*.
- [34] S. Yolchuyeva, G. Németh, and B. Gyires-Tóth, "Grapheme-to-phoneme conversion with convolutional neural networks," *Appl. Sci.*, vol. 9, no. 6, p. 1143, 2019.
- [35] Y. Sevinj, N. Géza, and G.-T. Bálint, "Transformer based grapheme-to-phoneme conversion," *Proc. Interspeech*, 2019, pp. 2095–2099.
- [36] X. Li, F. Metzger, D. Mortensen, S. Watanabe, and A. Black, "Zero-shot learning for grapheme to phoneme conversion with language ensemble," in *Proc. Findings Assoc. Comput. Linguistics (ACL)*, Dublin, Ireland, May 2022, pp. 2106–2115. [Online]. Available: <https://aclanthology.org/2022.findings-acl.166>
- [37] J. R. Novak, N. Minematsu, and K. Hirose, "Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework," *Natural Lang. Eng.*, vol. 22, no. 6, pp. 907–938, 2016.
- [38] V. Sar and T.-P. Tan, "Applying linguistic G2P knowledge on a statistical grapheme-to-phoneme conversion in Khmer," *Proc. Comput. Sci.*, vol. 161, pp. 415–423, Jan. 2019.
- [39] J. Duddington and R. Dunn. (2012). *ESpeak Text to Speech*. [Online]. Available: <http://espeak.sourceforge.net>
- [40] A. Parlikar, S. Sitaram, A. Wilkinson, and A. W. Black, "The Festvox indic frontend for grapheme to phoneme conversion," in *Proc. WILDRE: Workshop Indian Lang. Data-Resour. Eval.*, 2016.
- [41] V. Rajan. (2018). *Aksharamukha Script Converter Web Application*. [Online]. Available: <https://aksharamukha.appspot.com/about>
- [42] A. Kunchukuttan. (2020). *The IndicNLP Library*. [Online]. Available: [https://github.com/anoopkunchukuttan/indic\\_nlp\\_library/blob/master/docs/indicnlp.pdf](https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf)
- [43] S. Manghat, S. Manghat, and T. Schultz, "Malayalam-English code-switched: Grapheme to phoneme system," in *Proc. Interspeech*, 2020, pp. 4133–4137, doi: [10.21437/Interspeech.2020-1936](https://doi.org/10.21437/Interspeech.2020-1936).
- [44] P. V. Aswathy, A. Gopi, and T. Sajini, "Improving the accuracy of pronunciation lexicon using Naive Bayes classifier with character n-gram as feature: For language classified pronunciation lexicon generation," in *Proc. 11th Int. Conf. Natural Lang. Process.* Goa, India: NLP Association of India, Dec. 2014, pp. 113–118. [Online]. Available: <https://www.aclweb.org/anthology/W14-5118>
- [45] R. Priyamvada, D. Govind, V. K. Menon, B. Premjith, and K. P. Soman, "Grapheme to phoneme conversion for Malayalam speech using encoder-decoder architecture," in *Intelligent Data Engineering and Analytics*, S. C. Satapathy, P. Peer, J. Tang, V. Bhateja, and A. Ghosh, Eds. Singapore: Springer, 2022, pp. 41–49.
- [46] R. E. Asher and T. C. Kumari, *Malayalam (Descriptive Grammars)*. New York, NY, USA: Routledge, 1997.
- [47] T. Mohanan, "Syllable structure in Malayalam," *Linguistic Inquiry*, vol. 20, no. 4, pp. 589–626, 1989.
- [48] V. R. P. Nair, *Introduction to Linguistics*. Thiruvananthapuram, Kerala: MaluBen Publications, 2016.
- [49] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. London, U.K.: Pearson, 2014.
- [50] V. Z. Golob, J. V. Z. Gros, M. Žganec, B. Vesnicer, and S. Dobrišek, "FST-based pronunciation lexicon compression for speech engines," *Int. J. Adv. Robotic Syst.*, vol. 9, no. 5, p. 211, 2012.
- [51] H. Schmid, A. Fitschen, and U. Heid, "SMOR: A German computational morphology covering derivation, composition and inflection," in *Proc. 4th Int. Conf. Lang. Resour. Eval. (LREC)*, Lisbon, Portugal, May 2004, pp. 4–. [Online]. Available: <http://www.lrec-conf.org/proceedings/lrec2004/pdf/468.pdf>
- [52] U. Springmann, H. Schmid, and D. Najock, "LatMor: A Latin finite-state morphology encoding vowel quantity," *Open Linguistics*, vol. 2, no. 1, pp. 386–392, 2016.
- [53] H. Schmid, "SFST manual," Part of the SFST Software Package, 2005. [Online]. Available: <https://www.cis.uni-muenchen.de/~schmid/tools/SFST/data/SFST-Manual.pdf>
- [54] M. Mohri, "Finite-state transducers in language and speech processing," *Comput. Linguistics*, vol. 23, no. 2, pp. 269–311, 1997. <https://aclanthology.org/J97-2003>
- [55] K. Manohar. (2020). *Releasing Malayalam Speech Corpus*. [Online]. Available: <https://blog.smc.org.in/malayalam-speech-corpus/>
- [56] A. Tharwat, "Classification assessment methods," *Appl. Comput. Inform.*, vol. 17, no. 1, pp. 168–192, 2021, doi: [10.1016/j.aci.2018.08.003](https://doi.org/10.1016/j.aci.2018.08.003)
- [57] D. Povey, A. Ghoshal, and G. Boulianne, "The Kaldi speech recognition toolkit," in *IEEE workshop Autom. speech Recognit. Understand.*, Nov. 2011, pp. 1–4.
- [58] K. Prabhallad, E. N. Kumar, V. Keri, S. Rajendran, and A. W. Black, "The IIT-H Indic speech databases," in *Proc. 13th Annu. Conf. Int. speech Commun. Assoc.*, 2012, pp. 1–4.
- [59] A. Baby, "Resources for Indian languages," in *Proc. Text, Speech Dialogue*, 2016, pp. 37–43.
- [60] F. He, S.-H. C. Chu, O. Kjartansson, C. Rivera, A. Katanova, A. Gutkin, I. Demirsahin, C. Johnny, M. Jansche, S. Sarin, and K. Pipatsrisawat, "Open-source multi-speaker speech corpora for building Gujarati, Kannada, Malayalam, Marathi, Tamil and Telugu speech synthesis systems," in *Proc. 12th Lang. Resour. Eval. Conf. (LREC)*, Marseille, France, May 2020, pp. 6494–6503. [Online]. Available: <https://www.aclweb.org/anthology/2020.lrec-1.800>
- [61] Swathantra Malayalam Computing. (Mar. 2020). *Malayalam Text Corpora*. [Online]. Available: <https://gitlab.com/smc/corpus>
- [62] A. Stolcke, "SRILM—an extensible language modeling toolkit," in *Proc. 7th Int. Conf. Spoken Lang. Process.*, 2002, pp. 1–4.
- [63] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1–5.
- [64] G. Saon, H. Soltau, D. Nahamou, and M. Picheny, "Speaker adaptation of neural network acoustic models using I-vectors," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2013, pp. 55–59.



- [65] L. B. Babu, A. George, K. R. Sreelakshmi, and L. Mary, "Continuous speech recognition system for Malayalam language using Kaldi," in *Proc. Int. Conf. Emerg. Trends Innov. Eng. Technol. Res. (ICETIETR)*, 2018, pp. 1–4.
- [66] L. K. R. and E. Sherly, "An ASR system for Malayalam short stories using deep neural network in KALDI," in *Proc. Int. Conf. Artif. Intell. Smart Syst. (ICAIS)*, Mar. 2021, pp. 972–979.
- [67] D. Adiga, R. Kumar, A. Krishna, P. Jyothi, G. Ramakrishnan, and P. Goyal, "Automatic speech recognition in Sanskrit: A new speech corpus and modelling insights," in *Proc. Findings Assoc. Comput. Linguistics (ACL-IJCNLP)*, Aug. 2021, pp. 5039–5050. [Online]. Available: <https://aclanthology.org/2021.findings-acl.447>
- [68] S. Malviya, R. Mishra, and U. S. Tiwary, "Structural analysis of Hindi phonetics and a method for extraction of phonetically rich sentences from a very large Hindi text corpus," in *Proc. Conf. Oriental Chapter Int. Committee Coordination Standardization Speech Databases Assessment Techn. (O-COCOSDA)*, Oct. 2016, pp. 188–193.
- [69] H. M. Torres, J. A. Gurlekian, D. A. Evin, and C. G. C. Mercado, "Emilia: A speech corpus for Argentine Spanish text to speech synthesis," *Lang. Resour. Eval.*, vol. 53, no. 3, pp. 419–447, 2019.
- [70] N. Srivastava, R. Mukhopadhyay, K. R. Prajwal, and C. V. Jawahar, "IndicSpeech: Text-to-speech corpus for Indian languages," in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, May 2020, pp. 6417–6422. [Online]. Available: <https://www.aclweb.org/anthology/2020.lrec-1.789>



Her research interests include the computational linguistics, speech processing, and automatic speech recognition.

**KAVYA MANOHAR** received the B.Tech. degree from the University of Kerala, India, in 2010, and the M.Tech. degree from the University of Calicut, Kerala, India, in 2012. She is currently pursuing the Ph.D. degree with the College of Engineering Trivandrum, India. Since 2013, she has been serving as an Assistant Professor in electronics and communication engineering in engineering colleges with the University of Calicut.



He holds two patents for his work on consonant-vowel-ratio modification for improving speech perception. His research interest includes speech signal processing.

**A. R. JAYAN** received the B.Tech. degree in electronics and communication from the Government Engineering College, Thrissur, India, in 1992, the M.Tech. degree in electronics from the Cochin University of Science and Technology, India, in 1994, and the Ph.D. degree from the Indian Institute of Technology, Bombay, in 2014. He currently serves as a Professor with the Department of Electronics and Communication Engineering, Government Engineering College. He holds two



He currently serves as an Associate Professor with the Department of Electronics and Communication Engineering, College of Engineering Trivandrum. His research interest includes speech and music signal processing.

**RAJEEV RAJAN** (Senior Member, IEEE) received the B.Tech. degree in electronics and communication from the College of Engineering, Adoor (Cochin University of Science and Technology), in 2000, the M.Tech. degree in applied electronics and instrumentation from the College of Engineering Trivandrum, India, in 2004, and the Ph.D. degree from the Department of Computer Science and Engineering, Indian Institute of Technology, Madras, Chennai, India, in 2017.

...