EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH

Open Access

Improving speech recognition systems for the morphologically complex Malayalam language using subword tokens for language modeling

Kavya Manohar[1,2*], Jayan A R[1,3] and Rajeev Rajan[1,2]

Abstract

This article presents the research work on improving speech recognition systems for the morphologically complex Malayalam language using subword tokens for language modeling. The speech recognition system is built using a deep neural network–hidden Markov model (DNN-HMM)-based automatic speech recognition (ASR). We propose a novel method, syllable-byte pair encoding (S-BPE), that combines linguistically informed syllable tokenization with the data-driven tokenization method of byte pair encoding (BPE). The proposed method ensures words are always segmented at valid pronunciation boundaries. On a text corpus that has been divided into tokens using the proposed method, we construct statistical n-gram language models and assess the modeling effectiveness in terms of both information-theoretic and corpus linguistic metrics. A comparative study of the proposed method with other data-driven (BPE, Morfessor, and Unigram), linguistic (Syllable), and baseline (Word) tokenization algorithms is also presented. Pronunciation lexicons of subword tokenized units are built with pronunciation described as graphemes. We develop ASR systems employing the subword tokenized language models and pronunciation lexicons. The resulting ASR models are comprehensively evaluated to answer the research questions regarding the impact of subword tokenization algorithms on language modeling complexity and on ASR performance. Our study highlights the strong performance of the hybrid S-BPE tokens, achieving a notable 10.6% word error rate (WER), which represents a substantial 16.8% improvement over the baseline word-level ASR system. The ablation study has revealed that the performance of S-BPE segmentation, which initially underperformed compared to syllable tokens with lower amounts of textual data for language modeling, exhibited steady improvement with the increase in LM training data. The extensive ablation study indicates that there is a limited advantage in raising the n-gram order of the language model beyond $n = 3$. Such an increase results in considerable model size growth without significant improvements in WER. The implementation of the algorithm and all associated experiments are available under an open license, allowing for reproduction, adaptation, and reuse.

Keywords Subword tokens, Language modeling, Open vocabulary, Speech recognition, Morphological complexity, Malayalam language

*Correspondence:
Kavya Manohar
sakhi.kavya@gmail.com
Full list of author information is available at the end of the article

1 Introduction

Automatic speech recognition (ASR) is the process of converting speech acoustic signals into written text. This involves several stages, where the acoustic features are mapped to phonemes, which are the basic units of sound, and then reconstructed into meaningful words and sentences of the language under consideration.

Morphologically complex low-resource languages pose a significant challenge to speech recognition [1]. Morphologically complex languages exhibit productive word formation through agglutination, inflection, and compounding, resulting in very long words with phonetic and orthographic changes at morpheme boundaries [2]. The presence of such lengthy and morphologically complex words introduces difficulties in language modeling, which subsequently affects the word error rate (WER) in ASR tasks. This study aims to examine how different subword tokenization algorithms affect the performance of ASR models developed specifically for Malayalam.

1.1 Morphological complexity of Malayalam language

Malayalam is a morphologically complex language which has seven nominal case forms (nominative, accusative, dative, sociative, locative, instrumental, and genitive), two nominal number forms (singular and plural) and three gender forms (masculine, feminine, and neutral). These forms are indicated as suffixes to the nouns. Verbs in Malayalam get inflected based on tense (present, past, and future), mood (imperative, compulsive, promissive, optative, abilitative, purposive, permissive, precative, irrealis, monitory, quotative, conditional, and satisfactive), voice (active and passive), and aspect (habitual, iterative, perfect) [3].

The inflecting suffix forms vary depending on the final phonemes of the root words. Words agglutinate to form new words depending on the context [4]. Table 1 gives examples of a few complex word formations in Malayalam. It has been demonstrated in the literature that the Malayalam language exhibits a high level of morphological complexity than many other Indian and European languages in terms of type-token ratio and type-token growth rate [5, 6].

Figure 1 presents a comparison of the type-token growth rate of Malayalam with that of other Indian languages and English highlighting the notably higher rate at which new words (types) are encountered in the Malayalam corpus. This creates a large number of low-frequency words and it is practically impossible to build a pronunciation lexicon that covers all complex word forms. Additionally, it introduces the problem of data sparsity in language modeling [7].

1.2 Deep neural network–hidden Markov Model (DNN-HMM)-based ASR architecture

The classical ASR decoder shown in Fig. 2 is composed of an acoustic model (AM), a language model (LM), and a pronunciation lexicon (PL). The AM captures the relationship between acoustic features and phonemes in the language. The PL contains the phonemic representations of all words to be decoded by the ASR system. The LM establishes the statistical relationship between words in the language. Word-level statistical modeling of morphologically complex languages can not achieve the word sequence prediction capabilities of simple morphology languages [8, 9]. Additionally, the finite-sized word vocabulary of a pronunciation lexicon does not cover complex word forms and loan words that appear in a real-world setting. As a result, there is difficulty in recovering words that are not in the lexicon. [7].

The out-of-vocabulary (OOV) rate is the proportion of words in a given speech sample that are not present in the vocabulary of the ASR lexicon. OOV words can not be recognized by a word-based ASR decoder. A large number of OOV words and data sparsity are the natural consequences of word-based language models in ASR for morphologically complex languages [7]. Segmenting words to appropriate subword tokens before processing, and later reconstructing them to the whole words

Table 1 Complex morphological word formation in Malayalam

Malayalam word	English translation	Remark
പെട്ടിയിൽ/peṭṭijil/	In the box	Nominal locative suffix to the word പെട്ടി/peṭṭi/ (box).
കുട്ടിയോട്/kuṭṭijoːṭ/	To the child	Nominal sociative suffix to the word കുട്ടി/kuṭṭi/ (child).
ആനക്കുട്ടി/aːnakkuṭṭi/	Baby elephant	Compound word formed by agglutination of nouns ആന/aːna/ (elephant) and കുട്ടി/kuṭṭi/ (baby)
ആനക്കുട്ടികളോട്/aːn̪akkuṭṭikaḷoːṭ/	To the baby elephants	Nominal sociative suffix to the plural form of the compound word ആനക്കുട്ടി/aːnakkuṭṭi/ (baby elephant)
ഉണർന്നിരിക്കണ്ട/uɳarn̪n̪irikkaɳṭa/	Do not stay awake	Negative imperative mood of the verb ഉണരുക/uɳaruka/ (be awake)
പാടിക്കൊണ്ടിരിക്കും/paːṭikkoɳṭirikkum/	Will be singing	Future tense iterative aspect of the verb പാടുക/paːṭuka/ (to sing).

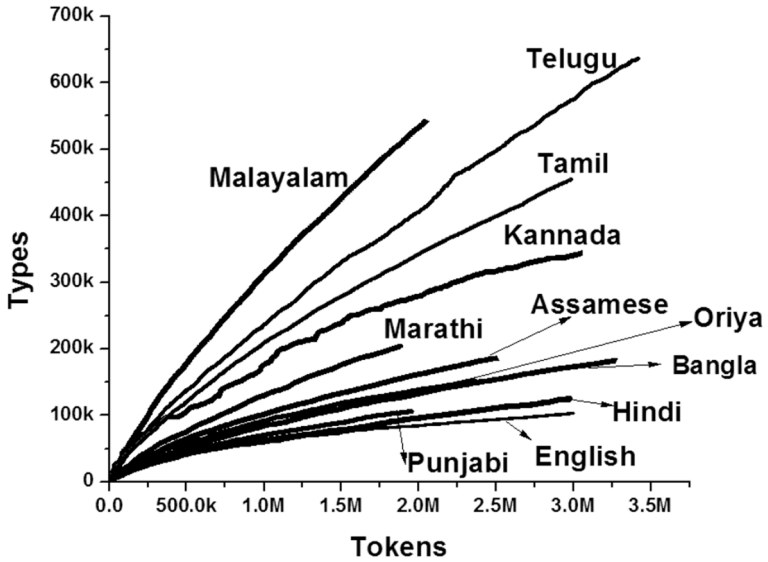

Fig. 1 Comparing the type-token growth rate of Indian languages in comparison to English. Reproduced from [5]

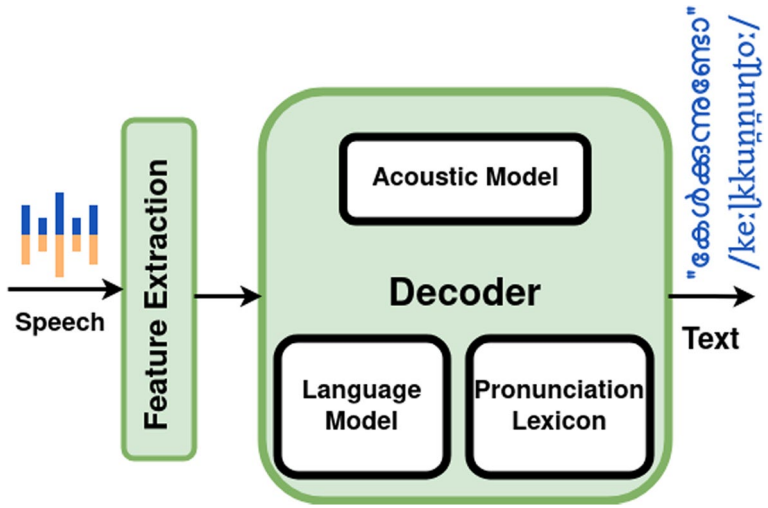

Fig. 2 Block schematic representation of a DNN-HMM ASR system

is a viable approach to solve the issues of data sparsity and OOV rates. When subword tokens are used for language modeling, a dummy symbol is added to identify the positions where the tokens can be glued together to form words [10]. When subwords replace words, the ASR vocabulary contains morphemes, syllables, or other character sequences that together can be used to create an unlimited number of words [11, 12].

An alternate approach to developing ASR models is the End to End (E2E) architecture, which is heavily data intensive. In the context of our research on the morphologically complex Malayalam language, we encountered a significant challenge related to the limited availability of annotated speech corpora under open licenses, making it a low-resource language. With less than 75 h of annotated data, training the E2E ASR system from scratch proved to be less practical due to the poor accuracy achieved with limited training data [13]. For improved speech recognition accuracy, DNN-HMM methods are best suited for languages with limited annotated speech [14]. Also, when much more text data is available than speech data, DNN-HMM models are the preferred choice [15, 16] than the modern E2E approaches. Additionally, DNN-HMM ASR models offer the advantage of easy integration into small hardware devices, enabling fast on-device speech recognition [14].

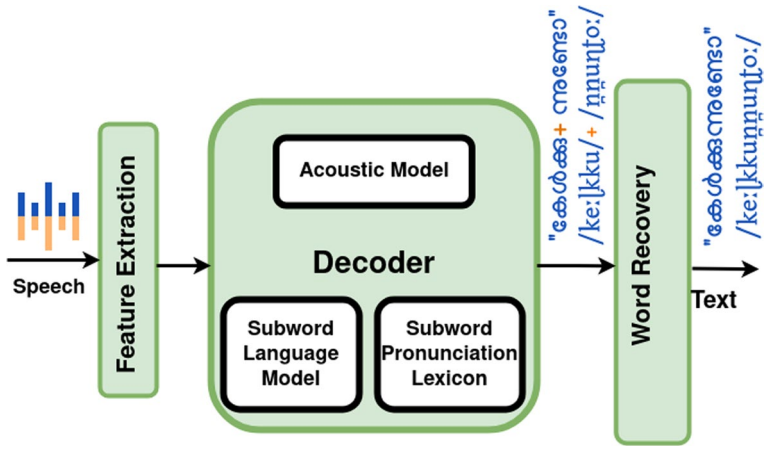

Fig. 3 Block schematic representation of DNN-HMM ASR system, with subword-based language model and pronunciation lexicon

Table 2 Subword tokenization illustrating the usage of continuity marker symbol '+'

Original text	അവൻ വള ഇടുകയില്ല /avan vaɭa iṭukajilla/
Subword tokenized text	അ+ വൻ വള ഇടുക+ യില്ല /a+ van va+ ɭa i+ ʈu+ ka+ ji+ lla/

Table 3 Phonemic Lexicon

Word	Pronunciation
SOPHIA	s o ʊ f i ə

Table 4 Graphemic Lexicon

Word/subword	Pronunciation
SOPHIA	S O P H I A
SOP+	S O P
HIA	H I A

1.3 DNN-HMM ASR with subword tokens

The ASR system illustrated in Fig. 3, which utilizes subword tokens, differs from the conventional ASR system depicted in Fig. 2 in terms of how the LM and the PL are constructed. In the subword token-based ASR, the LM is trained on a text corpus that is subword tokenized, and the PL consists of pronunciation descriptions for subwords instead of the conventional word-based entries. Subword ASR decoder requires an additional module to reconstruct words from the decoded subword units as shown in Fig. 3.

The reconstruction from subwords to words is facilitated by adding a dummy marker symbol [7]. In the experiments we perform in this work, we use the continuity marker "+" to the right side of the subwords, to indicate another subword has to follow it. In this approach, reconstruction is straightforward, as the marker indicates the positions for joining the following subword. Table 2 illustrates the usage of continuity markers in our experiments.

For languages where space and punctuation marks act as delimiters between words, segmenting the raw text of the language into word tokens is pretty straightforward. However, to segment text to subword units, there are data-driven as well as linguistically informed algorithms [1, 7]. Morfessor [11, 17, 18], byte pair encoding (BPE) [19, 20] and Unigram [21] are a few data driven algorithms in popular use. These algorithms do not ensure that subword tokenization happens at valid pronunciation boundaries. This makes precise representation of its pronunciation as a sequence of phonemes impossible.

For example, if the word SOPHIA/soʊfiə/ is segmented as SOP+ HIA, the pronunciation can not be segmented in a valid way. Then what is viable is to represent the subword tokens in the lexicon with their pronunciation described as a grapheme sequence. Tables 3 and 4 indicate how these entries would be represented in a phonemic and graphemic lexicon respectively. In this work, we use graphemic lexicons where graphemes would be mapped to acoustic features during acoustic model training. Subword tokens in lexicon entries have the continuity marker "+" indicating it will be followed by another subword segment to complete a word.

In this paper, we propose a novel hybrid subword tokenization algorithm, Syllable - byte pair encoding (S-BPE), combining linguistic syllabification rules with the data-driven tokenization method of BPE.

1.4 Research objectives and key contributions

The objectives of this paper are (i) to provide an overview of existing subword tokenization algorithms in Malayalam and to propose a hybrid algorithm of subword tokenization named S-BPE; (ii) to perform an analysis of the impact of different subword tokenization for Malayalam ASR in terms of WER, OOV-WER[1], and model memory requirement; and (iii) to contribute to the research on Malayalam ASR by publishing the source codes of our experiments[2] under permissive licenses to reuse and reproduce the results. We design our experiments to answer four related research questions (RQ).

- **RQ1:** Which subword tokenization method offers the most effective language modeling?
- **RQ2:** Does the subword tokenization method that exhibits the best language modeling complexity also result in the lowest WER for ASR tasks?
- **RQ3:** How does subword tokenization impact ASR performance concerning model memory requirements and OOV recovery rate in a morphologically complex language?
- **RQ4:** How do the amount of audio and textual training data and the n-gram order of the language model affect the effectiveness of subword token-based ASR in a morphologically complex language?

To answer our research questions, we create subword token-based n-gram language models using six tokenization algorithms: Word, Morfessor, BPE, Unigram, Syllable, and S-BPE. We analyze the complexity of language modeling using a modified perplexity metric (explained in detail in Section 2.2). These language models are then used to build DNN-HMM ASR models. We evaluate the performance of these ASR models based on metrics such as WER, OOV-WER, and the memory requirements of the models. The main contributions and practical implications of this study can be summarized as follows.

1 We introduce a novel subword tokenization algorithm, called S-BPE[3], specifically designed for the Malayalam language.
2 We investigate the impact of different subword tokenization algorithms on the complexity of language modeling. Our analysis reveals that tokenization algorithms resulting in a higher average number of tokens per word tend to receive a significant penalty in terms of the complexity evaluation metric known as surprisal per sentence (SPS). As a result, language models based on syllable-level tokenization exhibit the highest complexity, while word-level tokenization leads to the lowest complexity.
3 We also found that the reduction in language modeling complexity as measured by SPS does not necessarily imply a reduction in WER for ASR in morphologically complex languages.
4 With the proposed S-BPE tokenization algorithm, we could achieve state-of-the-art (SOTA) results for Malayalam ASR. The S-BPE model results in the best WER (10.6%) and best OOV-WER (24.8%), which is significantly better than the best baseline WER (27.4%) and OOV-WER (100%).
5 Additionally, the S-BPE model demonstrates model memory requirements comparable to other subword tokens, but considerably lower than the baseline word model. The significance of a smaller memory requirement happens in small hardware scenarios. It enables the efficient deployment of ASR models on resource-constrained devices, facilitating seamless on-device speech recognition.
6 Through a rigorous ablation study by varying the n-gram order and the amount of textual data used for language modeling and the amount of speech data used for acoustic modeling, we present the influence of these factors on ASR performance, providing valuable insights for optimizing ASR systems in the context of subword tokenization.
7 We provide open licenses for all source codes and models related to subword tokenization[4] and the resulting ASR models. This allows for public evaluation and facilitates further research advancements in the field.

In the following sections, we describe the related works, present the proposed algorithm, describe the experimental setup, and analyze the results which lead to answers to the research questions posed in the earlier section.

2 Related works

In this section, we review various studies related to subword tokenization in morphologically complex languages. We begin with different subword tokenization algorithms and explore the possibility of employing them for language modeling in the Malayalam language. We then explore the use of subword token-based language modeling on the DNN-HMM ASR task and discuss the SOTA status of Malayalam ASR.

[1] OOV-WER: The WER exclusively for OOV words
[2] ASR Training script: https://gitlab.com/kavyamanohar/ml-subword-asr
[3] S-BPE Code Repo: https://github.com/kavyamanohar/subword-syl-bpe-ml
[4] Segmentation Models: https://gitlab.com/kavyamanohar/ml-subword-segmentation

2.1 Subword tokenization algorithms

Subword token-based language modeling has been proposed for applications in speech recognition [7, 11, 12, 22, 23], statistical machine translation [24], neural machine translations [20, 21] and handwriting recognition [25]. The choice of subword tokens used in language modeling impacts the performance of the model on many downstream tasks [26] including speech recognition [7].

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

To perform tokenization that aligns with the Malayalam script, it is important to analyze the nature of the grapheme inventory of Malayalam. The graphemes in Malayalam script are classified as: (i) vowels; (ii) vowel signs; (iii) regular consonants; (iv) special consonants: *anuswara*, *visarga*, and *chili*; and (v) multi-functional character: *virama* [4]. Vowel graphemes occur only at word beginnings. Regular consonants inherently have the vowel /a/ present in them. Vowel sounds at positions other than word beginnings are represented by vowel signs. Vowel signs modify the inherent vowel sound of the consonants. A consonant cluster, also known as a conjunct, in Malayalam is a sequence of consonants separated by *virama* in between, where *virama* kills the inherent vowel from the preceding consonant [4]. *Chillus* are special consonants that do not have inherent vowels associated with them. The characteristics of other special consonants and *virama* are marked in Table 5. The four types of syllable structures possible in Malayalam are listed in Table 6 [27]. The syllable tokenization will make use of these linguistic rules.

Several approaches for tokenizing Malayalam text to meaningful morpheme units incorporating linguistic knowledge are reported in the literature. But for the reasons listed below, none of these could be used for the language modeling task required for ASR. For Malayalam morphological tokenization, earlier studies have used probabilistic, rule-based suffix-stripping, machine learning, and dictionary-based approaches [28–31]. The most recent deep learning technique uses Romanised Malayalam text and requires annotated data for training [32]. However, none of these research offers an application program interface (API) that can be programmed to perform morphological tokenization for use in downstream applications. The only tool with a programmable interface that works with Malayalam script performs morphological analysis[5] and not morphological tokenization [3]. For example, we need the compound word ആനകൾ/a:nakaɭ/ to be tokenized as ആന/a:na/+ , while its morphological analysis returns ആന/a:na/ <noun><plural>. Morphological analysis is not appropriate for an ASR task, as we expect to piece together the original word from the morpheme tokens by concatenation. For the ASR task, we, therefore, do not rely on any knowledge-based morpheme tokenization in Malayalam.

The data-driven tokenization algorithms are designed only based on word spellings and do not have access to pronunciation information. It is therefore possible for these algorithms to break a word sequence into units that do not imply well-formed correspondence to phonetic units. Pronunciation-assisted subword modeling (PASM) is an algorithm proposed to solve this issue by using a pronunciation dictionary as an aligner to determine the positions for tokenization [23]. To perform this task, PASM needs a pronunciation dictionary. PASM

Table 5 Special consonants and Virama sign in Malayalam

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

Table 6 Syllable structure in Malayalam with examples

Type	Syllable structure	Example
1	<BoW>[a] + vowel	അ /a/
	<BoW> + vowel + special consonant	അം /am/
2	Consonant	ക /ka/
	Consonant + special consonant	കൾ /kaɭ/
	Consonant + vowel sign + special consonant	കിം /kim/
3	Conjunct	സ്ത്ര /stra/
	Conjunct + special consonant	സ്ത്രം /st̪ra/
	Conjunct + vowel sign + special consonant	സ്ക്രീം /skri:m/
4	Consonant+ *virama* + <EoW>[b]	ക് /kə/
	Conjunct +ു+ *virama* +<EoW>	ട് /ʈʈə/

[a] Beginning of word

[b] End of word

[5] Mlmorph: https://pypi.org/project/mlmorph/

implemented in the English language was reported in [23] using the CMUDict pronunciation dictionary. But many low-resource languages do not have a ready-to-use pronunciation dictionary. In the comparative analysis performed in this work, we use several tokenization algorithms that break the pronunciation flow. So to be fair in the comparison, we use only graphemic lexicons and not phonemic ones. In this scenario, PASM for tokenization is not used.

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

2.1.1 Word tokens

In Malayalam, the technique of word tokenization is simple. After removing punctuation, the raw text corpus is divided up by spaces. Given a word of length M, it requires no more splitting, and hence time complexity of further tokenization of a word is a constant, denoted by $\mathcal{O}(1)$.

2.1.2 Morpheme tokens

Morfessor is a language-independent, data-driven method of subword tokenization. The Morfessor baseline algorithm is based on the minimum description length principle [17]. It is an unsupervised technique in which frequently occurring sub-strings in several different word forms from the training text corpus are proposed as morphs (or morpheme-like units) and the words are then represented as a concatenation of morphs [11]. Its current version, `Morfessor2.0`, has a Python interface that may be customized and it supports annotated training data as well [18]. The Morfessor algorithm does not guarantee either tokenization at appropriate pronunciation boundaries or tokenization into meaningful units.

2.1.3 BPE tokens

BPE is a data-driven algorithm that determines the optimal set of subword tokens through an iterative process. It was originally proposed as a data compression algorithm [19]. The BPE algorithm splits the training data into characters and creates an initial vocabulary. During further iterations, the most frequent character bigrams are determined, merged into a single token, and added to the vocabulary. The process is continued until a desired number of merge operations are performed. The final vocabulary size is the sum of the initial vocabulary and the number of merge operations, which is a hyperparameter [20]. The time complexity in training the BPE model is $\mathcal{O}(Nm)$, where N is the corpus length and m the number of merge operations [33].

The subword tokens in the learned vocabulary are later used to segment any text. BPE ensures that the most common words are represented in the pronunciation dictionary as a single token while the rare words are broken down into two or more subword tokens [20]. BPE tokenization algorithm available in `subword-nmt` Python library is used in the experiments described in this work[6]. The time complexity in tokenizing a word of length M using BPE implementation in [20] is $\mathcal{O}(M^2)$[7] [33].

2.1.4 Unigram tokens

Subword regularization with Unigram language model, henceforth refrerred to as Unigram tokenization [21] is a language-independent tokenization algorithm. It makes the assumption that the probability of the occurrence of subword tokens in the sentence is independent of one another. The vocabulary of the desired size is built from a heuristically large vocabulary by retaining only η% (say $\eta = 80$) of the subwords in each iteration and discarding the rest. The top 80% of subwords are obtained by ranking all subwords according to the likelihood reduction of removing them from the vocabulary. The most probable tokenization of an input sentence is determined by the Viterbi algorithm. The Unigram tokenization algorithm is available in the open source Python library, `sentencepiece`[8], which is used in the experiments performed in this work.

2.1.5 Syllable tokens

Orthographic syllable-based tokenization of text was proposed by Kunchukuttan et al. for statistical machine translation applications [24]. Splitting the tokens based on vowels and adjacent consonants, named vowel segmentation, was proposed by Adiga et al. and employed in the context of Sanskrit speech recognition [22]. These two methods segment text into syllable-like units at valid pronunciation boundaries.

A syllabification algorithm tailored for Malayalam script using finite state transducers has been proposed in [34]. The linguistic rules for syllable tokenization described in Table 6 have been computationally implemented as in Algorithm 1 and made available in the `Mlphon` Python library[9]. The algorithm analyzes the input text sequence and determines whether it falls into one of the four allowable categories of syllable structures in Malayalam. If it falls into any of these categories, it inserts tags (<BoS>

[6] subword-nmt: https://pypi.org/project/subword-nmt/

[7] Byte Pair Encoding and Data Structures https://guillaume-be.github.io/2021-09-16/byte_pair_encoding

[8] sentencepiece: https://pypi.org/project/sentencepiece/

[9] https://pypi.org/project/mlphon/

and <EoS>) at the appropriate positions to indicate the beginning and end of syllables. For a more comprehensive understanding of this algorithm, please refer to [34]. This results in variable-length subword tokens where each segment is a syllable with valid pronunciation. Since this algorithm is implemented as a deterministic and minimized finite state automaton, the time complexity for the syllable tokenization process is $\mathcal{O}(M)$, where M is the number of characters in a word [35].

The perplexity, $PPL(S)$, of a sentence, is the inverse probability normalized by the number of tokens, N. Normalization ensures, longer sentences are not heavily penalised [8].

$$PPL(S) = \left(\frac{1}{P(S)}\right)^{\frac{1}{N}} \tag{3}$$

1: **procedure** SYLLABLE BOUNDARY TAGGING
2: c_v ← consonant + virama
3: Type 1 ← <BoW> + vowel+[anuswara, visarga, chillu] ? ▷ ? indicates optionality
4: Type 2 ← consonant + vowelsign ? + [anuswara, visarga, chillu]?
5: Type 3 ← c_v * + C ▷ * indicates one or more occurence
6: Type 4 ← c_v ? + consonant + ◌्? + virama + <EoW>
7: syllable ← [Type 1, Type 2, Type 3, Type 4] ▷ Defines a syllable
8: `SyllableBoundaryTagger`: <BoS>+ syllable + <EoS> ← syllable
9: **end procedure**

Algorithm 1 FST-based Syllabification Algorithm

2.2 Language modeling analysis

Tokenization can mitigate the effects of rich morphology by breaking down highly inflected words into smaller components [9]. Statistical n-grams serve as a simple and powerful tool to capture language modeling information. The order of n-gram needed to capture this information depends largely on the properties of the tokens used. The tokenization algorithm determines the properties (the total number of tokens in the text, the number of characters within each segment, the number of tokens in a word, and the frequency of tokens) of the subword tokenized language modeling corpus.

Perplexity can be interpreted as the weighted average branching factor of a language. The branching factor of a language is the number of possible next words that can follow any word. Higher perplexity is positively correlated with difficulty in language modeling [36]. For a sentence, S, formed by a sequence of N tokens $S = s_1, s_2 ... s_N$, the probability $P(S)$ of the sentence is given by the following formula applying the chain rule of probability [36].

$$\begin{aligned} P(S) &= P(s_1, s_2, ...s_N) \\ &= P(s_1)P(s_2|s_1)...P(s_N|s_{N-1}, s_{N-2}...s_1) \end{aligned} \tag{1}$$

Based on the Markovian assumption of n-gram language modeling, the probability of each word depends only on the previous $n-1$ words [36]. This makes the sentence probability to be computed as

$$P(S) = \prod_{i=1}^{N} P(s_i|s_{i-1}, s_{i-2}..s_{i-(n-1)}) \tag{2}$$

Applying logarithm base 2 on the Eq. (3), we get

$$log_2\, PPL(S) = -\frac{1}{N} log_2\, (P(S)) \tag{4}$$

Since the number of tokens, N, in a sentence is largely determined by the tokenization method, the perplexity measure that is dependent on this parameter can not be used to compare modeling complexity across different tokenization algorithms [8]. The negative log-likelihood, $NLL(S)$, of the sentence probability distribution, effectively removes this dependency as described in the following equation and is called the *surprisal* of that sentence [9].

$$NLL(S) = -log_2\, (P(S)) \tag{5}$$

Based on Eq. (4), this can be rewritten as

$$NLL(S) = N log_2\, (PPL(S)) \tag{6}$$

By scaling down the $NLL(S)$ by the number of characters, M in a sentence, we obtain the character level surprisal, $NLL_c(S)$ as in Eq. (7) [37]. The number of characters in a sentence is a parameter independent of the tokenization used. This metric can also be used to compare language modeling complexity across tokenization algorithms.

$$NLL_c(S) = \frac{N}{M} log_2(PPL(S)) \tag{7}$$

Alternatively, word level surprisal, $NLL_w(S)$, where the scale factor is the number of words, W in a sentence, can also be used for comparison across tokenization algorithms [7, 12].

$$NLL_w(S) = \frac{N}{W} log_2(PPL(S)) \tag{8}$$

The metric surprisal per sentence (SPS) is a measure that quantifies the average surprisal of a corpus by dividing the total surprisal of the corpus by the number of sentences it contains. To calculate SPS, we follow these steps:

1 Compute surprisal: First, calculate the surprisal for each sentence in the corpus. This involves using a language model to estimate the probability of each word or subword unit in the sentence given the preceding context.
2 Sum surprisal: Add up the surprisal values for all sentences in the corpus to obtain the total surprisal.
3 Calculate SPS: Divide the total surprisal by the number of sentences k, to compute the SPS.

$$SPS = \frac{1}{k} \sum_{i=1}^{k} NLL(S_i) \tag{9}$$

For language modeling complexity comparisons across tokenization algorithms, SPS computation using $NLL(S)$ is employed in [9], NLL_c is used in [37], and $NLL_w(S)$ is used in [7].

2.3 Subword-based ASR

Subword-level modeling has proved to be instrumental in addressing the challenges posed by morphologically complex languages. Language modeling on subword tokens in ASR for morphologically complex languages has been extensively studied [11, 38, 39] in the framework of DNN-HMM ASR systems. Representation using subword units allowed for a more granular representation of words. Promising results have been achieved by applying Morfessor-based tokenization to DNN-HMM ASR systems, leading to reduced WER and improved OOV word recovery in languages such as Finnish, Arabic, Swedish, and English [7].

The introduction of subword units, such as morphemes or BPE tokens, has shown improved performance in capturing the morphological richness of Indian languages. An analysis of data-driven subword tokenization algorithms in Tamil and Kannada revealed significant reductions in WER compared to baseline word-based ASR systems. Specifically, the study conducted by Pilar et al. [40] reported an absolute WER reduction of 6.24% for Tamil and 6.63% for Kannada. Alternately, by manually identifying word classes and creating lists of prefixes, infixes, and suffixes for subwords, a subword grammar model was developed for Tamil and Kannada. This approach achieved even greater improvements in ASR performance, with a maximum absolute WER reduction of 12.39% for Tamil and 13.56% for Kannada [41]. These findings demonstrate the effectiveness of subword-level modeling techniques in Indian languages, highlighting their potential for enhancing ASR systems by effectively capturing the morphological complexity present in the languages.

Open vocabulary speech recognition in the Malayalam language has received limited exploration so far. To the best of our knowledge, the only prior work in this area was reported by Manghat et al. [12]. They proposed a tokenization technique that ensures the inclusion of rare subwords in the vocabulary, specifically focusing on Malayalam-English code-switched ASR. Their study marked the first attempt to employ subword-based language modeling for this particular language pair. Unfortunately, as the algorithmic implementation is not publicly available, we were unable to incorporate it into our experiments for this study.

The current work presented in this paper builds upon the initial findings presented in [42], which primarily focused on investigating the impact of using syllables as subword tokens in Malayalam ASR. While the previous study served as an exploration of a single algorithm, the present work represents a significant advancement. In the current study, we have expanded the acoustic modeling dataset size by five fold and conducted a thorough analysis of five different subword tokenization algorithms, including Morfessor, BPE, Unigram, Syllable, and S-BPE along with detailed ablation studies. These enhancements provide a more comprehensive understanding of subword-based approaches for ASR in the context of the Malayalam language.

2.3.1 Comparison with other reported works

When comparing ASR models, it is reasonable when a common benchmark dataset is used. Most of the works on Malayalam ASR are evaluated either on private datasets or the exact test data split is not published. Hence, we attempt to compare our results with the previous work [42] which was tested on the same test dataset as in the current work.

Notably, our SOTA results demonstrate remarkable improvements in ASR performance. In the previous work, the best WER achieved on a medium OOV test set was 26%. However, through the advancements made in the current study, we have achieved a significant reduction in WER. Specifically, by utilizing the proposed S-BPE method, we have achieved a remarkable WER of 10.6%. These results underscore the substantial improvements in ASR performance that can be achieved through the utilization of subword-based modeling techniques for the Malayalam language.

3 Proposed subword tokenization algorithm

In this section, we present the details of our proposed subword tokenization algorithm. It is built on top of the syllabification algorithm using finite state transducers

(FST) explained in Section 2.1.5. This hybrid algorithm combines the data-driven approach of BPE with the linguistic information about syllables.

3.1 S-BPE algorithm

S-BPE is a hybrid algorithm that takes into account syllables as irreducible units in the same way that BPE takes into account characters. The traditional BPE algorithm operates at the character level, where it progressively merges the most frequent pairs of characters into a single subword unit [20]. However, in the context of S-BPE, the algorithm operates on syllables instead of individual characters. The pseudocode for the S-BPE training algorithm is described in Algorithm 2. The S-BPE training begins by initializing the vocabulary, V, with all individual syllables present in the training data. It then iteratively applies the following steps:

1. Frequency calculation: The algorithm counts the frequencies of all pairs of adjacent syllables (S_L: The left syllable and S_R: The right syllable) in the training corpus, C.
2. Pair merging: It identifies the most frequent pair of syllables and merges them into a new subword unit, S_{new}. This merged subword unit is then added to the vocabulary.
3. Updating the corpus: The algorithm updates the corpus by replacing occurrences of the merged pair with the newly created subword unit, S_{new}.
4. Repeat: The algorithm continues to iterate, recalculating frequencies of pairs in the updated training corpus, merging the most frequent pairs, and updating the vocabulary until a predefined number of merges (k) is reached. The number of merge operations is set to $k = 10,000$, in our experiments.

Require: Training corpus (C), Number of merges (k)
procedure S-BPE(C,k)
  $V \leftarrow$ All unique syllables in C
  $merge\_counter = 0$
  while $merge\_counter < k$ **do** ▷ Merge frequent pair
    S_L, $S_R \leftarrow$ Most frequent pair in C
    $S_{new} \leftarrow S_L + S_R$ ▷ Merge most frequent pair
    $V \leftarrow V + S_{new}$
    Replace each occurrence of S_L, S_R with S_{new}
    $merge\_counter = merge\_counter + 1$
  end while
end procedure

Algorithm 2 S-BPE Training Algorithm

The algorithmic implementation[10] has been adapted from the original BPE algorithm in `subword-nmt` Python library, and made available under MIT License [20].

The syllabification operation, being implemented as an FST based regular expression [34], has a linear time complexity, $\mathcal{O}(N)$, where N is the number of characters in the training data. The time complexity of BPE taining is documented as $\mathcal{O}(sm)$ in [33], where s represents the length of the input string (measured in terms of the number of syllables), and m denotes the number of merge operations. The maximum possible number of syllables s is equal to N, the corpus length in number of characters. Thus the BPE portion of the algorithm may potentially have a complexity of $\mathcal{O}(Nm)$. To summarize, S-BPE algorithm has an overall time complexity determined by the dominant factor in $\mathcal{O}(N)$ and $\mathcal{O}(Nm)$, which is $\mathcal{O}(Nm)$.

Once the training part is completed, the S-BPE model is created with a model vocabulary. To segment words using S-BPE, the algorithm compares the input text with the learned vocabulary. First, the text is syllabified using a specific syllabification algorithm tailored for the Malayalam script (Algorithm 1). It has a linear time complexity, ie., syllabifying a word of M characters is $\mathcal{O}(M)$. Then, for every instance of the syllable sequence S_L,S_R in the text, the algorithm replaces it with a newly created subword symbol S_{new}. The replacements are performed in the order in which the symbols were learned and added to the vocabulary, as in the original BPE implementation in [20] and this process has a time complexity of $\mathcal{O}(M^2)$[11]. The overall time complexity of S-BPE-based syllabification is determined by the dominant factor which is $\mathcal{O}(M^2)$. On a comparative scale, the S-BPE algorithm has the same time complexity as that of BPE, both during training and during tokenization. However, tokenization is only a one-time process in the training of ASR models discussed in this work.

The S-BPE algorithm ensures that the most common words in the corpus are represented by a single symbol in the vocabulary. On the other hand, rare words are broken down into two or more subword tokens, while maintaining valid pronunciation for each segment. This combined process of knowledge-based syllabification and data-driven BPE allows for effective subword tokenization. While the syllabification algorithm is specifically designed for the Malayalam script, the S-BPE algorithm can be extended to other languages that can be syllabified.

In summary, the S-BPE algorithm leverages both knowledge-based syllabification and data-driven BPE techniques. It creates a vocabulary of frequent syllable sequences during training and uses this vocabulary to segment words into subwords during tokenization, ensuring effective representation of both common and rare words in the language.

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

[11] https://guillaume-be.github.io/2021-09-16/byte_pair_encoding

Table 7 Details of speech datasets used in our experiments

Corpus	#Speakers	#Utterances	Duration (hours)	Environment	Usage
Indic TTS, IITM [43]	2	8601	14	Studio	Training
Open SLR 63 - Train [44]	37	3346	5	Studio	Training
IMaSC [45]	8	34,473	49	Studio	Training
MSC [46]	75	1541	1	Natural	Training
IIITH [47]	1	1000	1	Studio	Development
Open SLR 63 - Test [44]	7	679	1	Studio	Testing

4 Experimental setup

This section presents the details of our experiments[12]. We begin with a description of the datasets followed by the structure of the DNN-HMM ASR system. We will describe in detail about the tokenization of the text corpus for language modeling, the creation of subword-tokenized pronunciation lexicons, and the process of acoustic modeling.

4.1 Datasets

Obtaining an adequate training corpus for Malayalam ASR is challenging due to the limited availability of comprehensive speech datasets. Creating such datasets involves resource-intensive tasks, including recruiting diverse speakers, establishing recording environments, and ensuring accurate transcription. While there is a vast amount of multimedia content available online, we have not attempted to use it for ASR training tasks mainly because of the difficulty in obtaining high-quality, aligned speech and text segments from the online content. To address this limitation, we leveraged existing publicly available open licensed speech datasets in Malayalam, such as Indic TTS [43], Open SLR 63 [44], IMaSC [45], MSC [46], and IIITH [47]. Among these, MSC has the highest number of speakers but exhibits an imbalance in the number of utterances per speaker. In contrast, Open SLR 63 offers a more balanced distribution of utterances among its 44 speakers, allowing for multi-speaker testing. Consequently, we partitioned the Open SLR 63 dataset to facilitate multi-speaker testing.

Each audio recording in the dataset is paired with a corresponding textual transcript written in the Malayalam script. The recordings are provided as `wav` files, with a sampling rate of either 16 or 48 kHz and 16-bit precision for each sample. For consistency during acoustic model training, the higher sampling rate of 48 kHz is downsampled to 16 kHz.

The speech dataset content is predominantly non-conversational in nature, with one dataset [46] recorded in natural environments. By including diverse speech samples from natural settings, we aim to enhance the robustness and generalizability of our findings. We divide the available speech into train and test datasets, ensuring zero speaker overlap. The train datasets described in Table 7 are combined to get approximately 69 h of audio for acoustic modeling. The ASR models are tested on a subset of the multi-speaker Open SLR 63 [44] dataset.

To create the language model, we use the sentences from the speech transcripts and combine it with the curated collection of text corpus published by SMC [48]. The resulting text corpus contains 227,686 sentences, 1,425,504 word types, and 364,170 unique word tokens.

4.2 DNN-HMM ASR system

The DNN-HMM ASR decoder consists of three modules as described in Fig. 2. The functions of these modules are listed below:

1. Acoustic model: It predicts the posterior likelihood $p(\mathbf{P}|\mathbf{X})$ of phone states $\mathbf{P} = p_0, p_1, ...p_K$ given the acoustic feature frames $\mathbf{X} = x_0, x_1, ..., x_N$ trained with deep neural networks based on the frame level alignment of audio and phoneme labels obtained from a previously trained GMM-HMM acoustic model [49].
2. Pronunciation lexicon: It maps words into a sequence of phonemes. The acoustic model training module would need to look up the pronunciation lexicon to convert the word-level transcripts into phoneme sequences.
3. Language model: It predicts the conditional likelihood $p(w_{i+1}|w_0, w_1, ...w_i)$ of the next word w_{i+1} given the previous words.

In the ASR decoder, all these components are composed into a weighted finite-state transducer framework [50] and the most likely word sequence is retrieved using graph search methods. This word-based system would serve as the baseline for our experiments. In a subword ASR system described in Fig. 3, the pronunciation lexicon and the language model are subword based.

[12] The Kaldi Experimental Setup: https://gitlab.com/kavyamanohar/ml-subword-asr

The creation of an acoustic model, a subword-tokenized text corpus for subword language model training, and the creation of subword-tokenized pronunciation lexicons are explained in the following subsections.

4.3 Acoustic modeling

Acoustic modeling in speech recognition begins with extracting relevant features from the raw audio signal. This process involves dividing the audio into frames of a fixed size using overlapping windows. To ensure smooth transitions at the frame borders and to avoid frequency artifacts, a Hamming window [51] is applied. In our experiments, we use a window size of 25 ms with a 10 ms overlap with the previous frame.

Once the speech signal is windowed, a fast Fourier transform (FFT) is applied to convert the signal from the time domain to the frequency domain. The resulting spectrum is then transformed logarithmically to obtain the log-magnitude representation. To capture the spectral characteristics relevant to speech recognition, the energy within specific Mel frequency ranges is computed. These energy values are typically represented as Mel frequency cepstral coefficients (MFCC), which provide a compact representation of the speech signal's spectral content [52]. MFCCs are commonly used as features in acoustic modeling due to their effectiveness in capturing the phonetic information necessary for speech recognition tasks [53].

In addition to MFCCs, the inclusion of i-vectors as features in the acoustic model training process is crucial [54]. These i-vectors play a vital role in effectively modeling and addressing speaker variability, resulting in improved recognition accuracy, especially in scenarios involving multiple speakers or unknown speakers. By capturing and incorporating speaker-specific information, i-vectors enable the system to adapt and account for individual speaker characteristics, ultimately enhancing the robustness of the acoustic model.

The training of the DNN-HMM model begins with the creation of a traditional HMM acoustic model, followed by utilizing the HMM state labels for each frame to train the time delay neural network (TDNN) acoustic model [55]. This two-step process facilitates the incorporation of both the conventional HMM framework and the powerful representation learning capabilities of the TDNN, resulting in an enhanced acoustic model for improved speech recognition performance.

Acoustic features used in TDNN training are (i) 40-dimensional MFCCs extracted from frames of 25 ms length and 10 ms shift and (ii) 100-dimensional i-vectors [56] computed from chunks of 150 consecutive frames. Three consecutive MFCC vectors (3×40 dimension) and the i-vector corresponding to a chunk (100 dimension) are concatenated, obtaining a 220-dimensional feature vector for a frame [14].

There are 16 layers of TDNNs, each working with different temporal contexts. Each layer is a succession of typical DNN operations, such as affine transforms, ReLU activations, and batch normalizations. Layers 2 to 13 use factored form of TDNN with the subsampled connection between layers. No subsampling is used in the remaining layers. All other hidden layers of the TDNN are trained in parallel. A declining learning rate was used, with an initial $\alpha_{initial} = 0.0015$ and a final $\alpha_{final} = 0.00015$. This acoustic model is trained simultaneously with two discriminative training criteria, one based on cross-entropy loss and the other based on maximum mutual information [57]. The dimension of the output layer is determined automatically, based on the number of tied phoneme states. The model is trained for 5 epochs where every layer uses L2 regularization to avoid overfitting. To achieve optimal WER, we made parameter adjustments motivated by improvements in WER on a development speech corpus, accounting for the interplay between the acoustic model, pronunciation lexicon, and language model. The model is trained on a single Nvidia Tesla T4 GPU.

4.4 Creating subword tokenized text corpora

Subword-based ASR, as shown in Fig. 3, is very much like a word-based ASR system, except that (i) the language model represents the conditional probability of subword sequences, instead of words and (ii) the pronunciation lexicon is composed of subword tokens. The word boundary marker is chosen so that the predicted subword tokens can be easily concatenated to form words. We use the tokenization algorithms described in Sections 2.1 and 3.1, and compare them with the baseline word-based ASR to answer the research questions.

Data-driven and hybrid tokenization algorithms require a training corpus for learning the model parameters, which is then applied to the target text to obtain a subword tokenized text corpus. Morfessor, BPE, and Unigram are data-driven tokenization algorithms while S-BPE is a hybrid one that additionally relies on linguistic knowledge. As the training corpus, we set aside a subset of the entire text corpus (7.5k sentences).

1. Words are separated by spaces in the text corpus and are thus already segmented.
2. Morfessor model is trained using the `morfessor` python library [18]. The training stops when the decrease in the model cost of the last iteration is smaller than `finish_threshold` value of 0.005. The trained model is applied to create the morpheme-tokenized text corpus.

3 BPE [20] learns the vocabulary from the training dataset. The initial vocabulary is formed by the Malayalam characters in the training dataset. The number of merge operations is set to 10,000. This results in a BPE model which is used to obtain the BPE tokenized text corpus.
4 Unigram [21] model is trained by the `sentence piece` library using the training dataset with a vocabulary of 15,000. The trained Unigram model is used to get the Unigram tokenized text corpus.
5 Being a rule-based algorithm, the syllabifier requires no training. Algorithm 1 is directly applied to the text corpus to obtain syllable tokenized corpus.
6 The S-BPE model is trained using the Algorithm 2 so that the vocabulary is learned. The initial vocabulary is formed by the Malayalam syllables present in the training dataset. The number of merge operations is set to 10,000. This results in a model which is used to obtain S-BPE tokenized text corpus.

Samples of text tokenized using these methods are presented in Table 8. These examples indicate how the number of tokens per sentence varies with the method of tokenization.

4.5 Language modeling

In the experiments performed in this work, we report SPS computed on $NLL(S)$ for measuring language modeling complexity. Statistical n-gram language modeling is performed on the subword tokenized text corpus. SRILM toolkit is used for the training and evaluation of language models [58]. To avoid zero probability assignment to unseen word sequences, the probability weights are redistributed by a process known as smoothing. We use the modified Kneser-Ney smoothing algorithm [59] to create n-gram language models of orders 2 to 6 for every tokenization algorithm. The models are trained to predict the next segment based on the previous n-gram context. The SRILM toolkit can evaluate the test dataset and return the log-likelihood values with respect to base 10 logarithms and the perplexity. Surprisal values are computed by converting these values to base 2 logarithms.

Table 8 Examples for different tokenization algorithms. Space is used as delimiter between tokens. Number of tokens per sentence is also tabulated

Method	Example	Segment count
Word	അവൻ വള ഇടുകയില്ല /avan vaḷa iṭukajilla/	3
Morfessor	അവ+ ൻ വള ഇട+ ുക+ യില്ല /ava+ n vaḷa iṭa+ uka+ jilla/	6
BPE	അവൻ വ+ ള ഇട്ട+ കയ+ ില്ല /avan va+ ḷa iṭu+ kaja + illa/	6
Unigram	അവ+ ൻ വള ഇട+ ുകയില്ല /ava+ n vaḷa iṭa+ ukajilla/	5
Syllable	അ+ വൻ വ+ ള ഇ+ ട്ട+ ക+ യി+ ല്ല /a+ van va+ ḷa i+ ṭu+ ka+ ji+ lla/	9
S-BPE	അവൻ വള ഇട്ട+ കയില്ല /avan vaḷa iṭu+ kajilla/	4

4.6 Creating subword tokenized lexicons

The graphemic lexicon describes the pronunciation using the language's native alphabets, or graphemes. Since BPE, Unigram, and Morfessor tokenization algorithms in our experiments do not have access to pronunciation information, the tokenization can happen at locations that break the pronunciation flow. So, it was decided to use a graphemic pronunciation lexicon, instead of a phonemic one [7] for all the tokenization algorithms to ensure fair comparison.

For the baseline ASR, the word pronunciation lexicon is prepared by using all the words in the text corpus with at least three occurrences. It is then expanded to include all the words in the training speech transcript. This word lexicon is referred to as PL_{word} and has 79,947 entries. Subword lexicons are obtained by segmenting every word entry in PL_{word} as per the tokenization algorithm under consideration and choosing the list of unique tokens as described in Algorithm 3. This involves the following steps.

1 Initialize an empty list to store the subword tokens.
2 Iterate over each word in the input word lexicon.
3 Tokenize the current word into subword units.
4 Add these to the list of subword tokens.
5 Repeat this process for all words in the input lexicon.
6 Make the list of subword tokens unique by removing duplicates.
7 Generate pronunciations for each unique subword token.

Require: PL_{word}
procedure CREATE-SUBWORD-LEXICON(PL_{word})
  $Tokens = \{\}$ ▷ Define an empty list
  for each $word$ in PL_{word} **do**
    $Tokens +=$ **Tokenize**($word$)
    ▷ Add subword tokens to the list
  end for
  $uniqueTokens \leftarrow$ **Unique**($Tokens$)
  ▷ List of unique subword token
  $PL_{subword} \leftarrow$ **Generate-Pronunciation**($uniqueTokens$)
  return $PL_{subword}$
end procedure

Algorithm 3 Subword Lexicon from Word Lexicon

The number of entries in these lexicons is described in Table 9.

4.7 Summary of experimental investigations

The acoustic models are built and combined with language models and pronunciation lexicons using the Kaldi toolkit [60]. From six ways (word, morfessor, BPE, unigram, syllable, S-BPE) tokenized text corpus, we construct language models with n-gram orders of 2 to 6. The language modeling effectiveness is then measured using corpus level and information theoretic metrics. Keeping the Kaldi-based TDNN acoustic model fixed across tokenization algorithms, we use subword tokenized lexicons and corresponding language models to create 30 (1 acoustic model × 6 subword tokenized lexicons × 5 n-gram orders) different ASR decoders. These decoders are then tested on a multispeaker test dataset described in Table 7.

5 Results

In this section, we present the findings from our experiments. We first perform a corpus linguistic analysis on the subword tokenized corpora. After that, we analyze the language model. This is done in terms of the metric SPS, which roughly indicate the complexity, and the overall difficulty that the model has in predicting sentences [8]. Finally, we analyze the ASR results. The WER, OOV-WER, lexicon size, and overall model size are used to measure this.

5.1 Corpus linguistic analysis of the LM

Words can be broken down into smaller pieces that are likely to convey similar meanings in different contexts by segmenting them into subwords, which can lessen the impact of rich morphology. We analyze the linguistic properties of these tokens in this section.

5.1.1 Linguistic validity of tokens

The tokens given by different methods, as exemplified in Table 8, do not necessarily comply with linguistic correctness. The word tokens are orthographically and phonetically valid linguistic units. The tokenization given by the Morfessor tool is not true morpheme tokens. The Morfessor tokens break the orthographic flow as in ഇടുക /iṭuka/ being subword tokenized as ഇട+/iṭa+/ ുക/uka/. In the second segment, the vowel sign ു, occurs without a consonant preceding it, which is an invalid orthographic usage. Similar invalid orthographic usages can be observed in BPE and Unigram tokenization algorithms too.

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

Table 9 Lexicon Sizes of different tokenization algorithms

Segmentation	Lexicon size
Word	79,947
Morfessor	10,545
BPE	9986
Unigram	19,564
Syllable	6279
S-BPE	15,926

5.1.2 Mean length of tokens

The mean length of tokens is the average number of characters in a token and it depends on the tokenization algorithm. The distribution of token lengths, in the form of box plots is shown in Fig. 4. It is the highest for words (8.3) as expected and the smallest is for syllables (2.2). The mean token length for Morfessor, Unigram, BPE, and S-BPE tokenization algorithms are 3.9, 4.3, 4.5, and 4.8, respectively.

A comparatively smaller box for syllables indicates the length is distributed closely about the median value, with very few outliers. However, for word tokenization, the length of the box plot is larger, indicating the segment lengths vary widely.

5.1.3 Token count per word and per sentence

The distribution of the number of tokens per word in the test dataset is illustrated in Fig. 5. Word tokenization does not break down the words, resulting in a single bar graph. In BPE, Unigram, and S-BPE tokenization algorithms, more than 50% of the words remain unsegmented, followed by words being tokenized into two subwords. In Morfessor tokenization, the distribution shows more than half the words are tokenized into two, followed by words remaining unsegmented. The percentage of words that get segmented into more than two tokens is rare in all these methods. However, in syllable tokenization, about 28% and 24% of words get segmented into two and three subwords respectively. The token length per word is more broadly distributed in syllable tokenization.

On analyzing the tokenization statistics over sentences, we get the values reported in Table 10. It

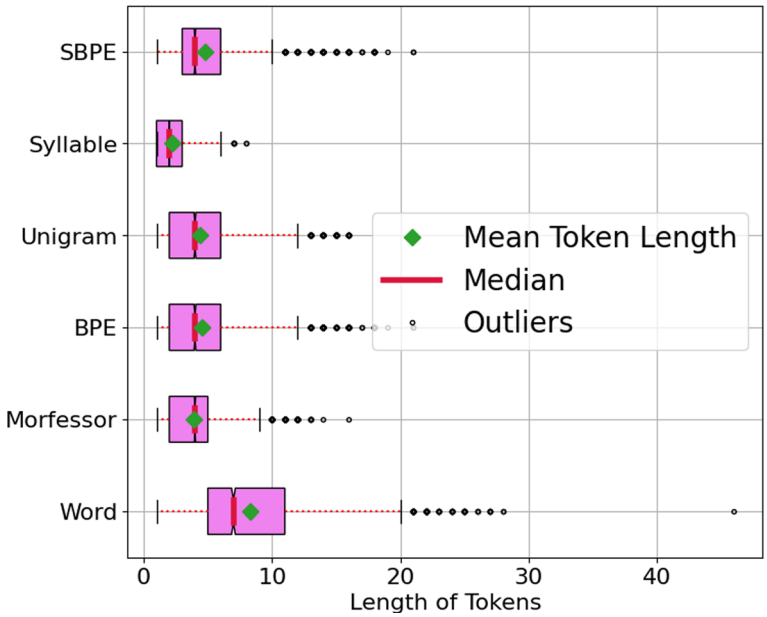

Fig. 4 Distribution of token length

describes the minimum, maximum, and mean number of tokens per sentence. Syllable tokenized sentences contain on average 19.9 subword tokens, which is the highest count of all. Sentences that contain a large number of tokens would need a longer n-gram language model context to guide the decoding [7]. We will analyze its impact on ASR later in this section.

5.2 Information theoretic analysis of the LM

The complexity of a language model is related to its difficulty in determining the next segment from the previous n-gram context. The higher order n-grams extract more context for the occurrence of a segment and generally reduce language modeling complexity and hence perplexity and surprisal. However, raising the n-gram order beyond a limit reintroduces the data sparsity problem, resulting in unimproved perplexity and surprisal values [37]. Subword language models require higher-order n-grams to capture the context than word-based ones [40]. In our experiments, we create language models of orders n=2 to 6 and analyze their complexity in terms of SPS.

The SPS values obtained in our experiments are shown in Table 11. For every tokenization method, with the increase in n-gram order, the SPS reduces initially and then stabilises. The best set of SPS values are obtained for the word segment-based language model. Our investigation demonstrates that tokenization algorithms yielding a greater average number of tokens per sentence are associated with a notable increase in the complexity evaluation metric SPS. Consequently, language models utilizing syllable-level tokenization demonstrate the highest complexity, whereas word-level tokenization yields the lowest complexity. Syllable tokens of lower n-gram orders show higher SPS values than all other tokenization algorithms.

The impact of subword token-based language modeling on the ASR decoder needs to be evaluated in terms of its ability to recover OOV words and a corresponding reduction in WER, which is attempted in the following section. However, lowering the language model complexity does not always ensure an improvement in automatic speech recognition accuracy [7, 61].

5.3 WER for each tokenization algorithm

To begin with, we present the ASR error rate which is computed as WER. It is based on the number of words inserted (I), deleted (D), and substituted (S) in the predicted speech transcript when compared to the ground truth transcript according to Eq. (10), where N represents the total number of words in ground truth transcript [62].

$$WER = \frac{(I + D + S) \times 100}{(N)} \tag{10}$$

The evaluation is performed on a multi-speaker studio recorded dataset. About 14% of words in this test dataset are OOV words, which can not be recovered by word-based ASR. According to [63], it has been shown that the presence of an OOV word in the test set can result in approximately two errors during ASR decoding.

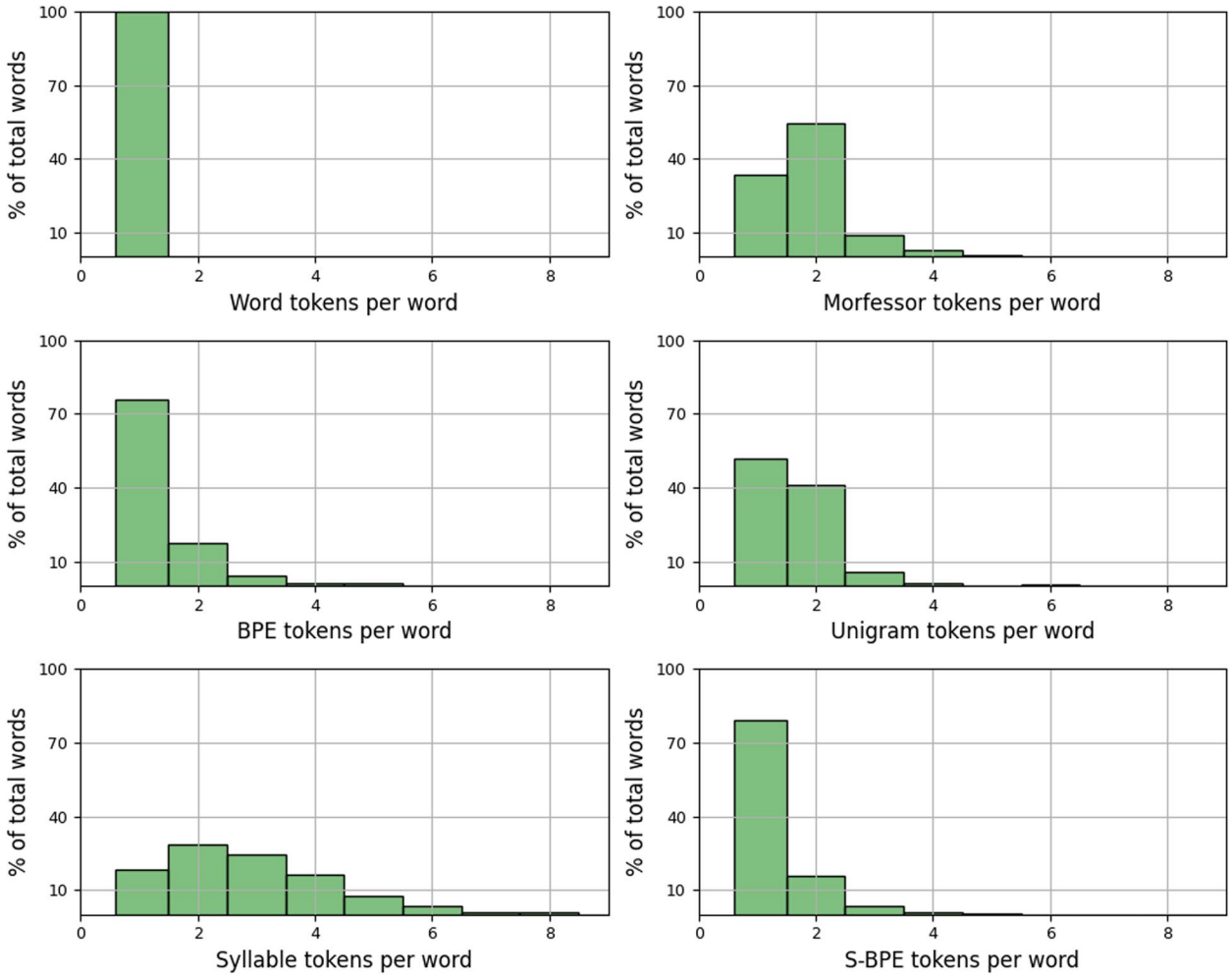

Fig. 5 Distribution of the number of tokens per word in the text corpus

Table 10 Sentence length statistics in terms of the number of tokens per sentence

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

Figure 6 presents the best set of WER obtained for different tokenization algorithms. To study the performance of subword-based ASR compared to the baseline word model on OOV recovery, we compute the WER, specifically for OOV words. The OOV words in the test set are determined with respect to the word-level lexicon. To analyze the extent of OOV-WER in subword token-based ASR, we use the `texterrors` Python library[64]. Providing the list of OOV words in the test set along with the true speech transcripts, this library computes the OOV-WER of the subword ASR model.

The baseline method, using words as the tokenization units, achieves a WER of 27.4% but suffers from a high OOV-WER of 100.0%, indicating that it struggles with words not present in the pronunciation lexicon. Among the alternative tokenization algorithms, Morfessor achieves a WER of 12.8% and significantly reduces the OOV-WER to 26.6%. BPE and Unigram tokenization also show competitive performance with WERs of 11.0% and 11.9% respectively, but their OOV-WERs remain close to that of Morfessor.

Syllable tokenization, while having a relatively higher WER of 13.5%, manages to achieve a lower OOV-WER of 24.8% compared to other methods. This is because syllables being the most granular of all tokenization algorithms, provide more opportunities for partial matching with available

Table 11 Language modeling complexity in terms of SPS. Lower SPS implies lower complexity

n-gram	Word SPS	Morf.[a]	BPE	Uni.[b]	Syl.[c]	S-BPE
2	45	88	82	108	157	78
3	42	68	64	84	109	62
4	42	63	61	79	93	60
5	42	62	61	78	85	60
6	42	62	61	79	82	60

[a] Morfessor

[b] Unigram

[c] Syllable

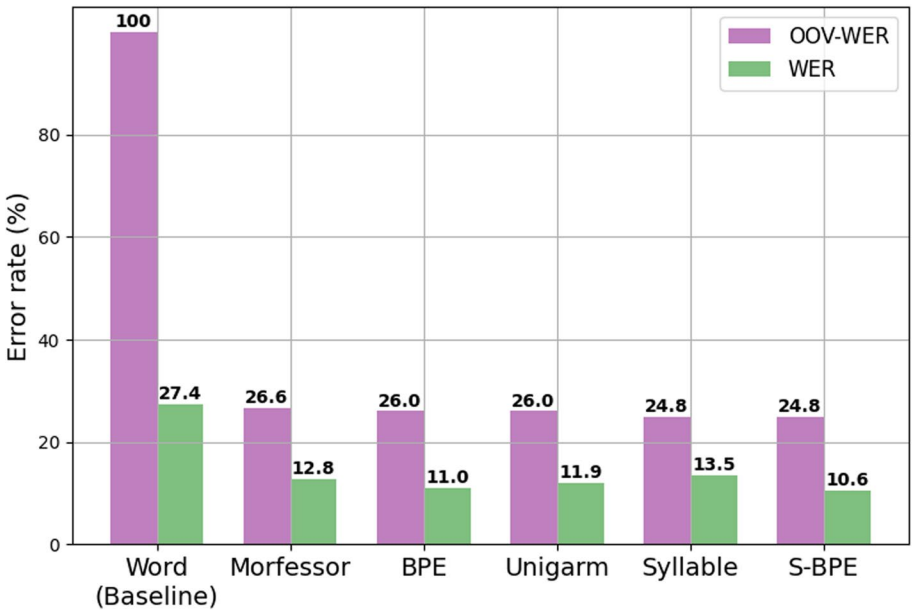

Fig. 6 The best WER for each tokenization method and the corresponding OOV-WER

lexical units, enabling better recovery of OOV words in ASR systems. However, this same property also makes syllable token-based ASR less suitable for general words, as it requires the decoder to recover a higher number of tokens per sentence, increasing the likelihood of errors.

Notably, the proposed hybrid S-BPE tokenization outperforms all other methods with the lowest WER of 10.6% and an OOV-WER of 24.8%, demonstrating its effectiveness in improving ASR performance. While both S-BPE and syllable tokenization exhibit comparable OOV-WER, S-BPE holds the added advantage of superior performance on non-OOV words. This is due to its ability to strike a balance between granularity and coverage. Overall, the results indicate that alternative tokenization algorithms offer improvements over the baseline word tokenization in terms of both WER and OOV-WER, with S-BPE yielding the best performance in this evaluation.

6 Ablation studies

In the preceding section, we presented the optimal WER achieved for each tokenization algorithm, leveraging 69 h of speech data for acoustic modeling and 227,686 sentences of textual data for language modeling, employing an n-gram order of $n = 6$. In this section, we investigate the influence of various tokenization algorithms by altering the n-gram order and adjusting the quantity of speech and textual training data used in the experiments. Our aim is to gain deeper insights into how these factors impact ASR performance and identify the most effective combination of tokenization and n-gram order to

Fig. 7 WER comparison for different tokenization algorithms in ASR with varying n-gram order

optimize the accuracy and memory efficiency of the ASR system.

6.1 Impact of n-gram order on WER for different tokenization algorithms

In this analysis, we investigate the impact of the n-gram order on the WER for different tokenization algorithms utilized in subword token-based ASR. The results are illustrated in Fig. 7. For ASR systems based on word tokens, the n-gram order shows little or no effect on the WER. However, in other subword token-based ASR models, we observe a significant reduction in WER, by at least 4%, when the n-gram order increases from $n = 2$ to $n = 3$. Further increases in the n-gram order beyond $n = 3$ result in only marginal improvements in WER.

Compared to the baseline word token-based ASR, all subword token-based ASR models perform better at all n-gram orders, except for the syllable token-based ASR at $n = 2$. Particularly, the S-BPE token-based ASR outperforms other tokens at corresponding n-gram orders.

Overall, this analysis offers valuable insights into how the n-gram order impacts WER for various tokenization algorithms in subword token-based ASR. It highlights the superiority of subword token-based approaches, especially S-BPE tokenization, and underscores the importance of choosing the right n-gram order to optimize ASR accuracy effectively.

6.2 Impact of n-gram Order on ASR model memory requirement for different Tokenization algorithms

The order of the n-gram impacts the memory requirement of the ASR model. To study the model memory requirement, we computed the size of the weighted FST graph (`HCLG.fst`) used for decoding. `HCLG.fst` is composed of four FSTs namely, `H.fst`, `C.fst`, `L.fst`, and `G.fst`. The `H.fst` and `C.fst` together form the acoustic model, `L.fst` the phonetic lexicon, and `G.fst` the grammar of the language model. Thus the total memory includes the model size for both the acoustic model and the language model combined.

The bar plot in Fig. 8 illustrates the impact of n-gram orders on the model memory requirement of different ASR models employing various tokenization algorithms. The x-axis represents the tokenization algorithms and the y-axis shows the ASR model memory requirement in Megabytes (MB).

From the plot, we observe that the n-gram order significantly affects the memory requirement of ASR models for most tokenization algorithms. As the n-gram order increases, the model memory requirement generally tends to rise across all tokenization algorithms.

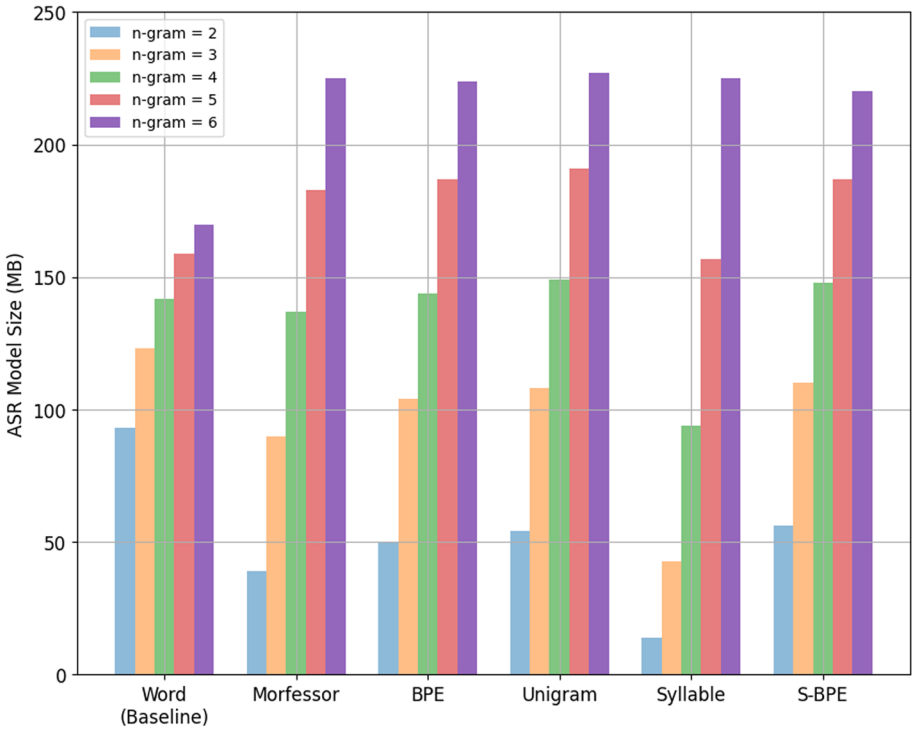

Fig. 8 Size of ASR model (MB) for different tokenization algorithms in ASR with varying n-gram order

This is expected as higher n-gram orders lead to larger language models, which require more memory to store the increased contextual information.

Notably, the word token-based ASR (baseline) shows relatively consistent memory requirements regardless of the n-gram order. However, other subword token-based ASR models, experience notable increases in memory requirement with higher n-gram orders. For n-gram orders ≤ 3, the syllable tokenization method exhibits the lowest memory requirement while S-BPE tokenization requires the most memory among the subword token-based models.

Based on the discussions about the WER and the model memory requirement, we find that higher n-gram orders, which lead to improved WER, come at the cost of increased model memory requirement. There exists a trade-off between achieving better ASR accuracy through more extensive context capture (higher n-gram order) and the computational resources needed to accommodate the larger language model in memory.

6.3 Trade-off between WER and model memory requirement

Through our investigation, we have observed that subword tokens of n-gram orders ≤ 3, exhibit a considerably smaller WER compared to the corresponding word-based models while having a lower model memory requirement. For n-gram orders above 4, the model size increases substantially without much improvement in WER and hence is not recommended. A comparative analysis of the WER and model size is presented in Table 12, for n-gram $= 3$.

In the trade-off diagram shown in Fig. 9, the model size of the word-based baseline ASR model does not change significantly with the model size. However, the error rate of the word-based baseline model is higher than all subword-based models, except for the syllable bigram ASR. Although the syllable bigram ASR has the smallest model size, its error rate is so high that it is not practical to use it.

Table 12 Comparing the WER and model size of each subword tokenization method, at n-gram = 3. The relative reduction with respect to the baseline word model is also shown in percentage

Segmentation	WER (%)		Model size (MB)	
Word (baseline)	27.4		123	
Morfessor	11.7	↓ 57%	104	↓ 15%
BPE	13.7	↓ 50%	90	↓ 26%
Unigram	12.6	↓ 54%	108	↓ 12%
Syllable	14.7	↓ 46%	94	↓ 23%
S-BPE	11.4	↓ 58%	110	↓ 11%

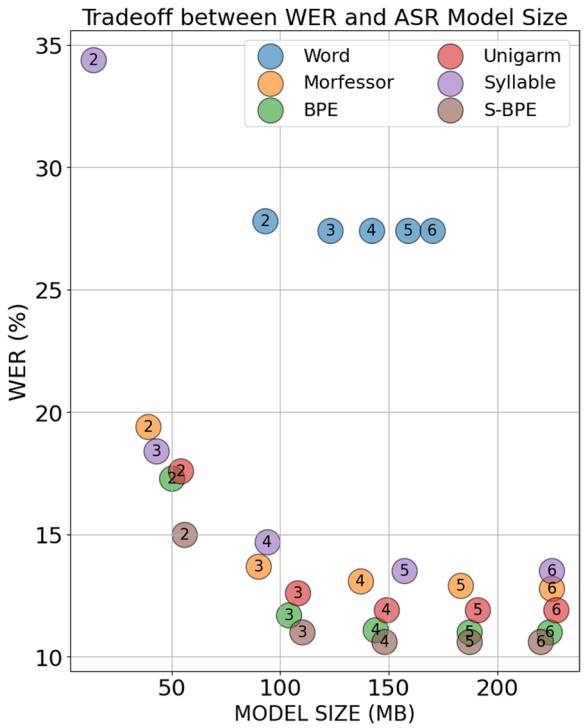

Fig. 9 Trade-off between ASR WER and Memory requirement. The n-gram order is indicated as labels within the circles

The performance of all subword models is better than the baseline, except for the bigram syllable language model. Syllables with the smallest mean length of tokens among all tokenization algorithms, require more n-gram context to make a reasonable prediction about the next subword segment. But once it has enough context, the predictions become more reliable as exemplified in the WER reduction in Fig. 9. The subword token-based ASRs show consistent improvement in WER with the increase in n-gram order. But the relative improvement diminishes with the n-gram order.

Researchers and practitioners need to strike a balance between n-gram order selection, WER performance, and available computational resources to design efficient and effective ASR systems. This involves considering the desired ASR accuracy, memory constraints, and computational capabilities when making decisions about n-gram order and other language modeling parameters to optimize ASR performance.

6.4 Impact of LM training data on WER for different tokenization algorithms

To investigate the influence of LM training data on the WER of the ASR model, we conducted experiments by creating language models using varying amounts of textual data, ranging from 12.5 to 100% of available 227,686 sentences utilizing an n-gram order of 3. The choice of n-gram was based on the fact that beyond n-gram=3, there would be a significant increase in model memory requirement without much improvement in WER. These language models were then combined with acoustic models built using the entire available audio corpora of about 69 h.

The plot in Fig. 10 depicts the relationship between WER and the percentage of available text corpora used for LM training. As the percentage of LM training data increases, there is a consistent reduction in WER for all tokenization algorithms. This demonstrates that more extensive LM training data leads to improved ASR accuracy, regardless of the tokenization approach used.

In the analysis, we observe that each tokenization algorithm exhibits distinct WER performance across different amounts of LM training data. Initially, when LM training data usage is low, all subword tokenizations demonstrate comparable performance. However, as the percentage of LM training data increases, the WER for these tokenizations starts to diverge. Morfessor, BPE, and Unigram tokenizations show competitive performance compared to the S-BPE tokenization, especially at lower levels of LM training data usage. However, with the increase in LM training data, the S-BPE tokenization consistently outperforms the others, showcasing the most robust WER reduction across all data sizes above 25%.

However, as the amount of LM training data increases, syllable tokenization, which initially showed competitiveness at lower data usage, gradually loses its competitive edge compared to other subword tokenizations. Its WER performance does not improve at the same rate as the other subword tokenizations, making it less favorable when utilizing the full available training data. In contrast, the word tokenization approach exhibits the highest WER among all algorithms, indicating its limitation in capturing the complexities of the language, especially in morphologically rich languages.

6.5 Impact of AM training data on WER for different tokenization algorithms

To investigate the influence of audio training data on the WER of ASR model, we conducted experiments by creating acoustic models using varying amounts of speech data, ranging from 4.5 to 69 h. These acoustic models were then combined with language models built using the entire available text corpora, utilizing an n-gram order of 3. The choice of n-gram was based on the fact that beyond n-gram=3, there would be a significant increase in model memory requirement without much improvement in WER (Fig. 11).

As the amount of training data increases, all tokenization algorithms show a clear reduction in WER. This

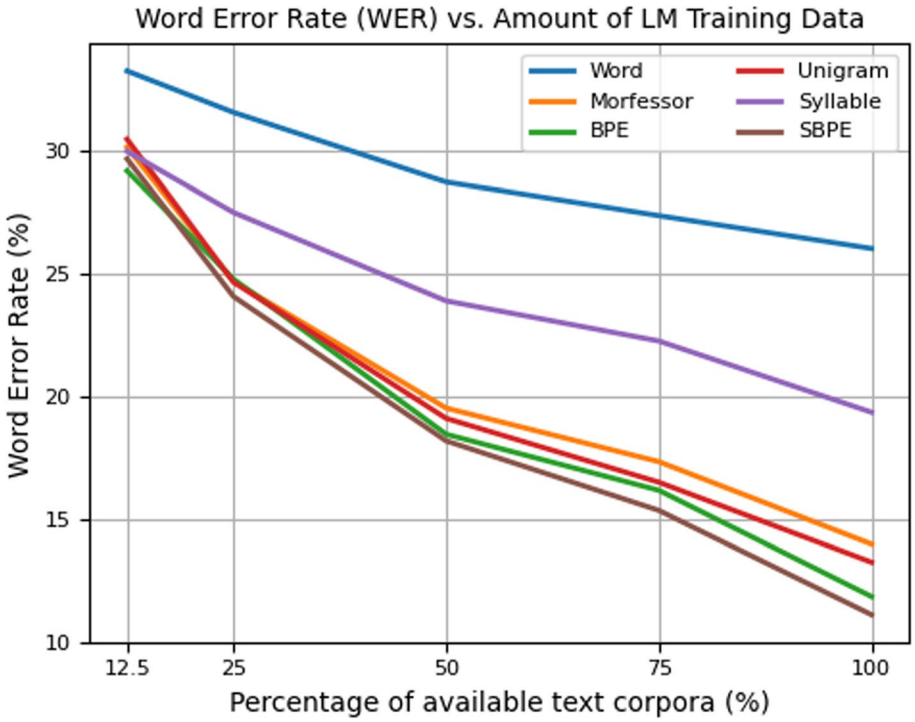

Fig. 10 WER vs. amount of textual data as a percentage of total available text corpora

finding depicted in Fig. 11 indicates that more extensive and diverse training data leads to improved ASR accuracy across all subword tokenization approaches.

Every subword tokenization algorithm maintains its relative position in terms of WER at all levels of available training data. This is because the tokenization algorithms directly impact the LM modeling and not the acoustic modeling and our LM model is constant thought out this experimental investigation. Among the analyzed tokenization algorithms, the S-BPE consistently exhibits the lowest WER.

7 Analysis and discussions

Based on our detailed experimental investigations on developing subword-based DNN-HMM ASR models for Malayalam, we have reached answers to our research questions posed in Section 1.4.

RQ1: Subword and word token-based language models were compared based on their complexity using the SPS metric. The results showed that all subword tokenizations resulted in a higher number of tokens per sentence compared to word-based tokenization. Consequently, the subword token-based language models exhibited higher values for SPS, indicating higher complexity compared to word token-based LMs. Syllable tokens demonstrated the highest values, suggesting that their higher granularity and token count adversely impacted the overall complexity of the language models. This shows the subword tokenization did not improve the LM modeling efficiency when evaluated using the SPS metric.

RQ2: The WER and LM complexity measured by the SPS metric were found to be uncorrelated. While subword tokenization did not significantly improve the intrinsic LM complexity metric, it indeed led to improved ASR performance compared to word tokenization. This is because ASR involves additional complexities related to acoustics, pronunciation variations, and OOV words. These factors influence WER independently of the LM complexity. Thus the subword tokenization method that exhibits the best language modeling complexity does not lead to the lowest WER for ASR tasks.

RQ3: Subword token-based ASR models exhibit reduced WER and decreased model memory requirements, especially when the n-gram order is less than $n = 4$ when compared to the baseline word token-based ASR. This finding suggests that subword tokenization can be highly beneficial for ASR tasks, particularly in morphologically complex languages or datasets with a large vocabulary. The hybrid method of S-BPE tokenization proposed in this work exhibited the lowest WER over diverse n-gram orders and AM and LM training data usage. Both S-BPE and Syllable token-based ASR could recover many OOV words resulting in the lowest OOV-WER of 24.8%.

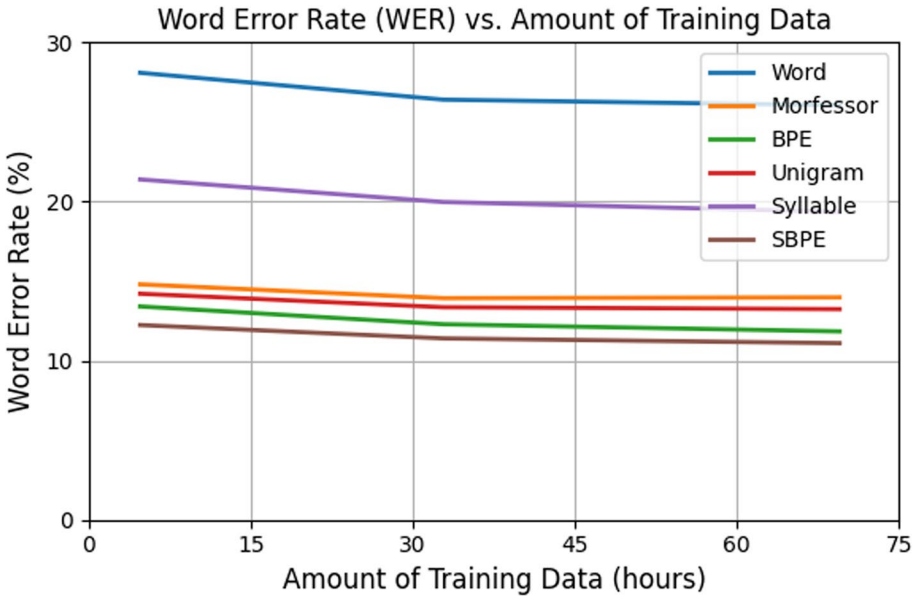

Fig. 11 WER vs. Amount of training data in hours

RQ4: The n-gram order significantly influences WER and ASR model size, with limited WER improvements beyond n-gram = 3, but higher orders increase model memory significantly. The optimal trade-off between WER and memory requires careful consideration of n-gram selection. Additionally, increasing the amount of AM training data generally leads to improved WER across all tokenization algorithms. On the other hand, the impact of increased LM training data varies for different tokenization algorithms. Some subword tokenizations (Morfessor, BPE, Unigram, and S-BPE) benefit more from additional LM training data, while others (word and syllable) did not show significant improvements. This underscores the importance of considering the interaction between tokenization algorithms and the amount of training data, both for LM and AM, to optimize ASR accuracy effectively. In all these ablation studies, S-BPE stood out with the best WER.

8 Conclusions

The presented study holds significant importance as it represents the first comprehensive investigation into improving speech recognition systems for the morphologically complex Malayalam language using subword language modeling techniques. By exploring various subword tokenization algorithms, we have conducted a detailed analysis of statistical n-gram language models' usage in the context of a hybrid ASR task.

The results of our study have demonstrated the exceptional performance of the proposed hybrid S-BPE tokens, achieving a remarkable 10.6% WER, which represents a 16.8% improvement over the baseline word-level ASR. While the linguistically informed syllable tokenization approach yielded a WER of 13.5%, it was unable to surpass the WER performance of other data-driven tokenization algorithms.

The comprehensive ablation study highlights that increasing the n-gram order of the language model beyond $n = 3$ offers little benefit, as it leads to significant model size growth without substantial WER improvement. On the other hand, augmenting the acoustic model training data consistently enhances WER across all tokenization algorithms. However, for data-driven tokenizations, increasing the LM training data proves especially beneficial, outperforming word and syllable tokenizations in terms of WER improvement.

In conclusion, the adoption of S-BPE subword tokens offers the advantage of reduced model memory requirements. It enables the efficient deployment of ASR models on memory-constrained devices, facilitating on-device speech recognition. Additionally, S-BPE and syllable subwords exhibit the lowest error rate for out-of-vocabulary words, effectively identifying more than 75% of these words, with an OOV-WER of 24.8%. The findings highlight the benefits of subword tokenization, including decreased model memory demands and improved accuracy, thereby greatly benefiting languages with complex morphology like Malayalam.

By addressing the challenges specific to Malayalam and offering valuable insights into subword tokenization techniques, our research makes a significant contribution to the field of speech recognition and lays the foundation for further advancements in ASR systems for morphologically rich languages.

Abbreviations

AM	Acoustic model
API	Application program interface
ASR	Automatic speech recognition
BPE	Byte pair encoding
DNN	Deep neural network
E2E	End to End
FFT	Fast Fourier transform
FST	Finite state transducers
HMM	Hidden Markov model
LM	Language model
MB	Megabytes
MFCC	Mel frequency cepstral coefficients
OOV	Out of vocabulary
PASM	Pronunciation-assisted subword modeling
PL	Pronunciation lexicon
RQ	Research questions
S-BPE	Syllable - byte pair encoding
SOTA	State of the art
SPS	Surprisal per sentence
TDNN	Time delay neural network
WER	Word error rate

Acknowledgements
We would like to acknowledge the publishers of all datasets and the authors of all open-source toolkits and libraries that made this research possible. We also thank the anonymous reviewers for helping shape this manuscript in its current form.

Authors' contributions
Kavya Manohar is responsible for the experimental design, implementation and interpretation of the results. Kavya Manohar drafted the manuscript and A. R. Jayan and Rajeev Rajan revised it critically for intellectual content. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The experiments in this work use open-licensed, publicly available speech and text datasets described in Section 4.1.
The source code for implementing the proposed subword tokenization algorithm is presented in this repository. All other subword tokenizations rely on open-licensed libraries: Morfessor (Morfessor Python Library), BPE (Subword NMTPython Library), Unigram (Sentence Piece Python Library) and Malayalam Syllabifier (Mlphon Python Library).
The acoustic model was created using Kaldi speech recognition toolkit. Its training requires a GPU. Statistical language model was created using SRILM toolkit. The acoustic model, the language model and the pronunciation lexicon are combined to form the ASR decoder using Kaldi. We have made the experimental script available in this repository.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]College of Engineering Trivandrum, Thiruvananthapuram, 695 016 Kerala, India. [2]APJ Abdul Kalam Technological University, Thiruvananthapuram, Kerala, India. [3]Government Engineering College, Thrissur, Kerala, India.

Received: 27 January 2023 Accepted: 12 October 2023
Published online: 04 November 2023

References

1. L. Besacier, E. Barnard, A. Karpov, T. Schultz, Automatic Speech Recognition for Under-resourced Languages: A Survey. Speech Commun. **56**, 85–100 (2014). https://doi.org/10.1016/j.specom.2013.07.008
2. M. Baerman, D. Brown, G.G. Corbett, *Understanding and measuring morphological complexity* (Oxford University Press, USA, 2015)
3. S. Thottingal, in *Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages*. Finite State Transducer based Morphology analysis for Malayalam Language (European Association for Machine Translation, Dublin, 2019), pp. 1–5. https://aclanthology.org/W19-6801. Accessed 4 Sept 2023.
4. R.E. Asher, T.C. Kumari, *Malayalam (Descriptive grammars)* (Routledge, London and New York, 1997)
5. G.B. Kumar, K.N. Murthy, B. Chaudhuri, Statistical Analyses of Telugu Text Corpora. IJDL. Int. J. Dravidian Linguist. **36**(2), 71–99 (2007)
6. K. Manohar, A. Jayan, R. Rajan, in *International Conference on Text, Speech, and Dialogue*. Quantitative Analysis of the Morphological Complexity of Malayalam Language (Springer, 2020), pp. 71–78. https://doi.org/10.1007/978-3-030-58323-1_7
7. P. Smit, S. Virpioja, M. Kurimo, Advances in Subword-Based HMM-DNN Speech Recognition Across Languages. Comput Speech Lang. **66**, 101158 (2021). https://doi.org/10.1016/j.csl.2020.101158
8. S.J. Mielke, R. Cotterell, K. Gorman, B. Roark, J. Eisner, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. What Kind of Language Is Hard to Language-Model? (Association for Computational Linguistics, Florence, 2019), pp. 4975–4989. https://doi.org/10.18653/v1/P19-1491
9. H.H. Park, K.J. Zhang, C. Haley, K. Steimel, H. Liu, L. Schwartz, Morphology Matters: A Multilingual Language Modeling Analysis. Trans. Assoc. Comput. Linguist. **9**, 261–276 (2021). https://doi.org/10.1162/tacl_a_00365
10. P. Smit, S. Virpioja, M. Kurimo, in *Proc. Interspeech 2017*. Improved Subword Modeling for WFST-Based Speech Recognition (2017), pp. 2551–2555. https://doi.org/10.21437/Interspeech.2017-103
11. M. Creutz, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arisoy, M. Saraçlar, A. Stolcke, Morph-Based Speech Recognition and Modeling of out-of-Vocabulary Words across Languages. ACM Trans. Speech Lang. Process. **5**(1) (2007). https://doi.org/10.1145/1322391.1322394
12. S. Manghat, S. Manghat, T. Schultz, in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Hybrid Sub-word Segmentation for Handling Long Tail in Morphologically Rich Low Resource Languages (2022), pp. 6122–6126. https://doi.org/10.1109/ICASSP43922.2022.9746652
13. H.S. Chadha, A. Gupta, P. Shah, N. Chhimwal, A. Dhuriya, R. Gaur, V. Raghavan, Vakyansh: Asr toolkit for low resource indic languages. arXiv preprint arXiv:2203.16512 (2022)
14. A.L. Georgescu, A. Pappalardo, H. Cucu, M. Blott, Performance vs. Hardware Requirements in State-of-the-art Automatic Speech Recognition. EURASIP J. Audio Speech Music. Process. **2021**(1), 1–30 (2021). https://doi.org/10.1186/s13636-021-00217-4
15. S.P. Bayerl, K. Riedhammer, in *Text, Speech, and Dialogue*, ed. by K. Ekštein. A Comparison of Hybrid and End-to-End Models for Syllable Recognition (Springer International Publishing, Cham, 2019), pp. 352–360. https://doi.org/10.1007/978-3-030-27947-9_30
16. A. Rouhe, A. Van Camp, M. Singh, H. Van Hamme, M. Kurimo, in *Speech and Computer*, ed. by A. Karpov, R. Potapova. An Equal Data Setting for Attention-Based Encoder-Decoder and HMM/DNN Models: A Case Study in Finnish ASR (Springer International Publishing, Cham, 2021), pp. 602–613
17. M. Creutz, K. Lagus, in *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*. Unsupervised Discovery of Morphemes (2002), pp. 21–30
18. S. Virpioja, P. Smit, S.A. Grönroos, M. Kurimo, et al., Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. (Aalto University, 2013), pp. 38. http://urn.fi/URN:ISBN:978-952-60-5501-5

19. P. Gage, A New Algorithm for Data Compression. C Users J. **12**(2), 23–38 (1994)
20. R. Sennrich, B. Haddow, A. Birch, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Neural Machine Translation of Rare Words with Subword Units (Association for Computational Linguistics, Berlin, 2016), pp. 1715–1725. https://doi.org/10.18653/v1/P16-1162
21. T. Kudo, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates (Association for Computational Linguistics, Melbourne, 2018), pp. 66–75. https://doi.org/10.18653/v1/P18-1007
22. Adiga, Devaraja and Kumar, Rishabh and Krishna, Amrith and Jyothi, Preethi and Ramakrishnan, Ganesh and Goyal, Pawan, in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Automatic Speech Recognition in Sanskrit: A New Speech Corpus and Modelling Insights (2021). https://doi.org/10.18653/v1/2021.findings-acl.447
23. H. Xu, S. Ding, S. Watanabe, in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Improving End-to-end Speech Recognition with Pronunciation-assisted Sub-word Modeling (2019), pp. 7110–7114. https://doi.org/10.1109/ICASSP.2019.8682494
24. A. Kunchukuttan, P. Bhattacharyya, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Orthographic Syllable as basic unit for SMT between Related Languages (Association for Computational Linguistics, Austin, 2016), pp. 1912–1917. https://doi.org/10.18653/v1/D16-1196
25. H. Singh, R.K. Sharma, V. Singh, Online Handwriting Recognition Systems for Indic and non-Indic scripts: A Review. Artif. Intell. Rev. **54**(2), 1525–1579 (2021). https://doi.org/10.1007/s10462-020-09886-7
26. C. Toraman, E.H. Yilmaz, F. Şahïnuç, O. Ozcelik, Impact of tokenization on language models: An analysis for turkish. ACM Trans. Asian Low-Resour. Lang. Inf. Process. **22**(4) (2023). https://doi.org/10.1145/3578707
27. V.R.P. Nair, ഭാഷാശാസ്ത്രപരിചയം /bʱaːʂaːʃaːstɾapaɾiʧajam/ *Introduction to Linguistics* (MaluBen Publications, Thiruvananthapuram, 2016)
28. R. Rajeev, E. Sherly, in *Proceedings of 20th Kerala Science Congress*. A suffix Stripping based Morph Analyser for Malayalam Language (Kerala State Council for Science, Technology and Environment, Thriruvananthapuram, 2007), pp. 482–484
29. O. Rinju, R. Rajeev, P.R. Raj, E. Sherly, Morphological Analyzer for Malayalam: Probabilistic Method vs Rule based Method. Int. J. Comput. Linguist. Nat. Lang. Process. **2**(10), 502–507 (2013)
30. P. Antony, K. Soman, Computational Morphology and Natural Language Parsing for Indian Languages: A Literature Survey. Int. J. Sci. Eng. Res. **3**, 136-146 (2012)
31. V. Abeera, S. Aparna, R. Rekha, M. Anand Kumar, V. Dhanalakshmi, K. Soman, S. Rajendran, in *International Conference on Data Engineering and Management*. Morphological analyzer for Malayalam using Machine Learning (Springer, 2010), pp. 252–254
32. B. Premjith, K.P. Soman, M.A. Kumar, A Deep Learning Approach for Malayalam Morphological Analysis at Character Level. Procedia Comput. Sci. **132**, 47–54 (2018). https://doi.org/10.1016/j.procs.2018.05.058
33. V. Zouhar, C. Meister, J. Gastaldi, L. Du, T. Vieira, M. Sachan, R. Cotterell, in *Findings of the Association for Computational Linguistics: ACL 2023*. A formal perspective on byte-pair encoding (Association for Computational Linguistics, Toronto, 2023), pp. 598–614. https://doi.org/10.18653/v1/2023.findings-acl.38
34. K. Manohar, A.R. Jayan, R. Rajan, Mlphon: A Multifunctional Grapheme-Phoneme Conversion Tool Using Finite State Transducers. IEEE Access **10**, 97555–97575 (2022). https://doi.org/10.1109/ACCESS.2022.3204403
35. G. Berry, R. Sethi, From regular expressions to deterministic automata. Theor. Comput. Sci. **48**, 117–126 (1986)
36. D. Jurafsky, J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (Pearson, India, 2009)
37. A.I.E.D. Mousa, Sub-word based language modeling of morphologically rich languages for lvcsr. Ph.D. thesis, RWTH Aachen University (2014)
38. T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, J. Pylkkönen, Unlimited Vocabulary Speech Recognition with Morph Language Models Applied to Finnish. Comput. Speech Lang. **20**(4), 515–541 (2006). https://doi.org/10.1016/j.csl.2005.07.002
39. G. Choueiter, D. Povey, S. Chen, G. Zweig, in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Morpheme-Based Language Modeling for Arabic LVCSR, vol. 1 (2006), pp. I–I. https://doi.org/10.1109/ICASSP.2006.1660205
40. B. Pilar, et al., Subword Dictionary Learning and Segmentation Techniques for Automatic Speech Recognition in Tamil and Kannada. arXiv preprint arXiv:2207.13331 (2022)
41. B. Pilar, et al., Knowledge-driven subword grammar modeling for automatic speech recognition in tamil and kannada. arXiv preprint arXiv:2207.13333 (2022)
42. K. Manohar, A.R. Jayan, R. Rajan, in *Proceedings of the Third International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2022) co-located with ICNLSP 2022*. Syllable subword tokens for open vocabulary speech recognition in Malayalam (Association for Computational Linguistics, Trento, 2022), pp. 1–7. https://aclanthology.org/2022.nsurl-1.1
43. A. Baby, A.L. Thomas, N. Nishanthi, T. Consortium, et al., in *Proceedings of Text, Speech and Dialogue*. Resources for Indian languages (Springer, Cham, 2016)
44. F. He, S.H.C. Chu, O. Kjartansson, C. Rivera, A. Katanova, A. Gutkin, I. Demirsahin, C. Johny, M. Jansche, S. Sarin, K. Pipatsrisawat, in *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*. Open-source Multi-speaker Speech Corpora for Building Gujarati, Kannada, Malayalam, Marathi, Tamil and Telugu Speech Synthesis Systems (European Language Resources Association (ELRA), Marseille, 2020), pp. 6494–6503. https://www.aclweb.org/anthology/2020.lrec-1.800
45. D.P. Gopinath, V.V. Nair, et al., Imasc–icfoss malayalam speech corpus. arXiv preprint arXiv:2211.12796 (2022)
46. K. Manohar. Releasing Malayalam speech corpus. (2020). https://blog.smc.org.in/malayalam-speech-corpus/. Accessed 1 Sept 2023
47. K. Prahallad, E.N. Kumar, V. Keri, S. Rajendran, A.W. Black, in *Thirteenth annual conference of the international speech communication association*. The IIIT-H Indic speech databases (ISCA, Portland, 2012)
48. S.M. Computing. Malayalam text corpora (Swathanthra Malayalam Computing, Kerala, 2020). https://gitlab.com/smc/corpus. Retrieved on Spetember 01, 2023
49. P. Żelasko, S. Feng, L. Moro Velázquez, A. Abavisani, S. Bhati, O. Scharenborg, M. Hasegawa-Johnson, N. Dehak, Discovering Phonetic Inventories with Crosslingual Automatic Speech Recognition. Comput. Speech Lang. **74**(C) (2022). https://doi.org/10.1016/j.csl.2022.101358
50. M. Mohri, F. Pereira, M. Riley, Weighted Finite-state Transducers in Speech Recognition. Comput. Speech Lang. **16**(1), 69–88 (2002). https://doi.org/10.1006/csla.2001.0184
51. R.W. Hamming, *Digital filters* (Courier Corporation, USA, 1998)
52. S. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. **28**(4), 357–366 (1980)
53. S.S. Stevens, J. Volkmann, E.B. Newman, A scale for the measurement of the psychological magnitude pitch. J. Acoust. Soc. Am. **8**(3), 185–190 (1937)
54. N. Dehak, P.J. Kenny, R. Dehak, P. Dumouchel, P. Ouellet, Front-end factor analysis for speaker verification. IEEE Trans. Audio Speech Lang. Process. **19**(4), 788–798 (2010)
55. V. Peddinti, D. Povey, S. Khudanpur, in *Sixteenth annual conference of the international speech communication association*. A time delay neural network architecture for efficient modeling of long temporal contexts (ISCA, Dresden, 2015)
56. G. Saon, H. Soltau, D. Nahamoo, M. Picheny, in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. Speaker adaptation of neural network acoustic models using i-vectors (IEEE, Olomouc, 2013), pp. 55–59
57. D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, S. Khudanpur, in *Interspeech*. Purely sequence-trained neural networks for asr based on lattice-free mmi. (2016), pp. 2751–2755
58. A. Stolcke, in *Seventh international conference on spoken language processing*. SRILM-an extensible language modeling toolkit (ISCA, Denver, 2002)
59. R. Kneser, H. Ney, in *1995 International Conference on Acoustics, Speech, and Signal Processing*. Improved backing-off for m-gram language modeling, vol. 1 (1995), pp. 181–184. https://doi.org/10.1109/ICASSP.1995.479394
60. D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al., in *IEEE 2011 workshop on automatic speech recognition and understanding*. The Kaldi speech recognition toolkit (IEEE, Columbia, 2011), CONF

61. P. Kłosowski, Statistical analysis of orthographic and phonemic language corpus for word-based and phoneme-based polish language modelling. EURASIP J. Audio Speech Music Process. **2017**(1), 1–16 (2017)
62. J. Benesty, M.M. Sondhi, Y. Huang et al., *Springer handbook of speech processing*, vol. 1 (Springer, Berlin, 2008)
63. M. Bisani, H. Ney, in *Proc. Interspeech 2005*. Open vocabulary speech recognition with flat hybrid models (2005), pp. 725–728. https://doi.org/10.21437/Interspeech.2005-11
64. R.A. Braun, S. Madikeri, P. Motlicek, in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. A Comparison of Methods for OOV-Word Recognition on a New Public Dataset (IEEE, 2021), pp. 5979–5983. https://doi.org/10.1109/ICASSP39728.2021.9415124

Publisher's Note