International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I1P126

Volume 73 Issue 1, 313-323, January 2025

Original Article

Novel Method to Annotate Text Properties for Indic Language Kannada

Vivekananda[1], K. C. Ravishankar[2]

[1]*Department of Artificial Intelligence and Data Science, Navkis College of Engineering, Affiliated to Visvesvaraya Technological University., Karnataka, India.*
[2]*Department of Computer Science and Engineering, Government Engineering College, Affiliated to Visvesvaraya Technological University., Karnataka, India.*

[1]*Corresponding Author: viveka.research@gmail.com*

Received: 03 August 2024 Revised: 11 December 2024 Accepted: 23 December 2024 Published: 31 January 2025

Abstract *- Natural Language Processing includes Machine Translation and Transliteration processes. These Processes are helpful in knowledge transfer across language barriers. The language processing tools and algorithms should analyse the source language for efficient results. Therefore, language processing tasks have source language analysis as one of the preliminary steps. Text analysis to understand syllable structures is an important initial step. Understanding syllable structures of the agglutinative and abugida languages is more challenging because of complex syllable structures. Various morphemes are stitched together in agglutinative languages to build long words with complex meanings. In abugida languages, letters are built using consonants and vowel sequences. Syllabification and syllable segmentation processes explore complex syllable structures of abugida languages. This paper proposes a novel approach to annotate and classify an abugida-type language, Kannada text syllables, by building a Syntax Directed Translation (SDT) and implementing it using PLY (A python LEX and YACC tool).*

Keywords *- NLP, Syllabification, Kannada text processing, Syllable annotation, Syllable classification, Syllable segmentation.*

1. Introduction

Language is the tool mastered by humans to become civilized, intelligent, and capable creatures on earth. In efforts to survive against time, humans built a strong communication mechanism to record and transfer knowledge. Language in different forms dates way back to prehistoric periods. Presently, there are approximately 7,151 languages spoken today across the world [1]. Historically, knowledge is localized in its nature. The effect of globalization opened the opportunities for interactions, so the access to knowledge treasures. Today, the sharing of knowledge and information has no language barriers. The barrier-free knowledge era is mainly aided by technological advancements towards the cross-cultural world. Natural Language Processing (NLP), in particular, enhances information access with minimal effort. Carrying information from one part of the world to another safely, accurately, and speedily was achieved long back, but understandability is the concern. Content written in a foreign language always poses difficulty in understanding. Extracting actual sentiments or preparing an abstract or a mere translation or synthesis are always challenging tasks for non-native language experts. To negotiate such a situation, NLP offers a handful of algorithms for translation and transliterations to start the proceedings. Once the content is translated by the tool into a known language (target language), further processing, such as sentiment analysis or language understanding, becomes an achievable goal with the help of the language expert. To build an intelligent language system to replace human experts, each language must be technically equipped with language processing algorithms and tools with a clear understanding of the basic components of the source language. Building language-specific processing tools for any low-resource language requires a primary understanding of the language to the finest granularity. The syllabification gives an intuition about the syllable properties of the given language text. The present paper proposes a rule-based automatic syllabification method. The proposed work presents a Syntax Directed Translation Scheme to collect the syllabic characteristics of each letter.

1.1. Background

Languages are majorly categorized as abjad, logographic, and abugida based on the scripts. Indic-Languages belong to the abugida family. Kannada is a member of the Indic-Language family. It is the state abugida or alphasyllabic in nature as that of the Indic-Languages family. In Indic-Languages ‘Akshara’ or ‘orthographic syllable’ is the basic unit. Each Akshara is made up of a Consonantal core. Akshara are categorized based on the number of consonant symbols present in them. The first set is Independent Vowels (Swara) having null consonants in the Consonantal core.

Table 1. Kannada syllable classes and members of the classes

Syllable Class and Members of the Class	
Independent Vowels (Swara)	
Short Vowels (Hraswa swara)	ಅ ಇ ಉ ಋ ಎ ಒ (a i u r e o)
Long Vowels (Dheerga Swara)	ಆ ಈ ಊ ಏ ಐ ಓ ಔ (ā ī ū ē ai ō au)
Anusvara	ಅಂ (aṁ)
Visrga	ಅಃ (aḥ)
Secondary Forms of Vowels (SFV) / Dependent Vowels	
short (Hrasva)	ಿ, ು,ೃ,ೆ,ೊ
long (dirgha)	ಾ, ೀ, ೂ, ೇ,,ೈ,ೋ,ೌ,ಂ,ಃ
Consonants (Vyanjana)	
classified Consonants (Vargeeya Vyanjana)	ಕ ಖ ಗ ಘ ಙ (ka kha ga gha ṅa) ಚ ಛ ಜ ಝ ಞ (ca cha ja jha ña) ಟ ಠ ಡ ಢ ಣ (ṭa ṭha ḍa ḍha ṇa) ತ ಥ ದ ಧ ನ (ta tha da dha na) ಪ ಫ ಬ ಭ ಮ (pa pha ba bha ma)
nasal (Anunasika)	ಙ ಞ ಣ ನ ಮ (ṅa ña ṇa na ma)
Unclassified Consonants (Avargeeya Vyanjana)	ಯ ರ ಲ ವ ಶ ಷ ಸ ಹ ಳ (ya ra la va śa ṣa sa ha ḷa)
Consonant Conjuncts (Samyuktakshras)	
Biconsonantal conjuncts	ಕ್ಕ (kka) (ಕ +್ + ಕ)
Tri-consonantal conjuncts	**ಸ್ತ್ರೀ** (strī) (ಸ + ್ + ತ + ್ + ರ + ೀ)

Swara is again subdivided into Hraswa Swara (Short Vowel) and Deerga Swara (Long vowel), depending on the time it takes to pronounce. Two special symbols were identified, namely Anuswara and Visrga. There are dependent vowel signs (Matras) to represent one-to-one correspondence to the independent Vowels. Matras do not have an independent appearance in the scripts; rather, they will be augmented to a single consonant or cluster of consonants to form an Akshara. Consonants (vyanjana) set having a single consonant sound augmented with an inherent vowel. Consonants in Kannada are of two types: one is a set of classified consonants (Vargeeya Vyanjana), and the other is unclassified consonants (Avargeeya Vyanjana). Vargeeya Vynajan set has the subclass Anu-nasika (Nasal) symbols. Consonant conjuncts (Samyuktakshras) are formed using a cluster of consonants as a series of dead consonants (ending with a halanth or without an inherently augmented vowel, represented as consonant Cd) followed by normal consonants and then by optional Dependent vowels [2]. Consonants with a Secondary Form of Vowels (SFV) are called 'Gunithakhsra'. The series of gunithakshara is known as Kagunitha. The summary of the Kannada syllables and their classification is given in Table 1.

1.2. Literature Review

Chinnakotla M. K et al. [3] proposed a transliteration model by employing manual character mapping rules for Hindi, English, and Persian Languages. The base rule adapted is to map a single Hindi character to one or more English candidate characters. An optimal substitute is selected by running a ranking function driven by word generation probability using the Character Sequence Model (CSM). This adopts a rule-based approach and suggests more exploration of the phonological properties of characters to avoid ambiguities. Loitonbam Gyanendro Singh et al. [4] proposed a hybrid approach for automatic syllabification of the Manipuri Language using a combination of Entropy-based Phonetic Segmentation (EPS) and Conditional Random Field (CRF) methods. This work performs statistical analysis on manually syllabified Manipuri corpus by annotating the positional information for each character at the training phase. The phonetics of the Manipuri language is explored with the help of the probability distribution of phonemes. The results are validated using a 5-fold cross-validation method. The classification accuracy achieved is 98%. This work is focused on identifying Syllable structures based on their positional distribution. Orthographic features of the test corpora syllables are processed automatically. Anoop Kunchukuttan and Pushpak Bhattacharyya [5] proposed a Statistical Machine Translation (SMT) based on orthographic syllables for the Hindi language. Hindi is a member of the Indic-Language family, and it is written in Devanagari script. Syllabification is achieved using a straightforward approach to add space between each text character by verifying the vowels, dependent vowels, and consonants. If nasalizers like Anusara or Chandra-Bindu are found, they are handled separately. Later, this information is used in SMT of language pairs. The

objective of the syllabification is to segment the text. Prakash Padakannaya [6] insisted on developing the theories and models primarily for the alphabetic system applicable to the Indian alphasyllabary. The Western models or any universal models do not fully fit Indic-Languages' processing. A hypothetical model is proposed for Indic languages' reading and writing process. This model has a component named 'Mental Lexicon' with the functionality of transforming orthographic lexicon to phonological lexicon and vice-versa.

This model expects to collect orthographic information at first glance. The paper reports that an orthographic lexicon builder (lexical pathway) is the weak link in the system. Anoop Kunchukuttan et al. [7] proposed Multilingual Transliteration Systems (MTS) for orthographically similar languages. The experiment setup incorporates phonetic information in processing. Phonetic information is stored in a bit-vector, which has a bit for each property of the character.

Later, this vector is used to feed input to the Convolution Neural Network (CNN) encoder to achieve better results. The information extraction available at [8] is based on the range of Unicode values representing different Indic-Languages. Tables representing different languages possess identical offsets to represent a different category of syllables. This fact is explored with simple if-else logic to extract the syllabic properties. Indic Layout Requirements, W3C Working [9], outlined Augmented Backus-Naur form (ABNF) grammar for text segmentation of Indic-orthographic Syllables.

The grammar has three abstract rules with limited grammar symbols. For example, a terminal named 'V' is used to represent an independent vowel set, undermining the fact that two types of vowels (short and long) are present. Literature available concerning Indic-Languages syllabification with the intent of properties annotation is very little. To the best of our knowledge, no exclusive literature is available for syllabification of Kannada Language text. The work reported in [8] is built using rule-based if-else-driven logic for a set of Indic Languages. The properties considered are {VOWEL, CONSONANT, NUKTA, HALANT, ANUSVAAR, MISC}. The are no other recent state-of-the-art references found in the literature addressing the syllabification of the Kannada language. The proposed work presents an expressive way of identifying a rich set of attributes through a novel Context-Free Grammar.

2. Methodology

This paper proposes an intuitive, rule-based novel method to annotate the syllabic features of Kannada Akshara. Akshara in Kannada is written horizontally from left to right. Table 2 illustrates the formation of the set of letters (consonant-based) and the classes that may be associated with them accordingly. In the example, there is a constant in all the classes; also, the other attributes are common in many classes (Short and long vowels, Consonants with SFV, anusvara and consonant Conjuncts). Hence, we should consider the longest attribute set to be the preferable match for the correct classification. Similar letter formation rules can be expected from close relatives of the Kannada Language, such as Telugu and Malayalam; also, with little modification, this is applicable for other Indic-language family members. A context-free grammar is proposed to accept Unicode character streams as the Kannada syllables.

Then, semantic actions are defined to produce annotated attributes and classify Kannada syllables. This is a novel approach to implementing a syllable annotator for the Kannada language using the Syntax Directed Translation (SDT) scheme. The General Structure of a Kannada letter is as follows:

consonant → c_d c_d c [SFV] ([Anuswra] |[visarga])
| c_d c [SFV] ([anuswra]|[visarga])
| c [SFV]([anuswra]|[visarga])
| c_d
vowel → V([anuswra]|[visarga])

Grammar KanSyllable= {V, T, S, P} is defined in Table 3. Table 4 is the proposed Syntax Directed Translation with postfix semantic actions to the respective production rules of the proposed CFG. Then, an Algorithm is outlined to implement the proposed SDT using the PLY tool.

Table 2. Syllabic features of Kannada Akshara illustration

Letter	Features
ಕ (ka)	Consonant
ಕಂ (kaṁ)	Consonant with anusvara
ಕು (ku)	Consonant with short SVF
ಕುಂ (kuṁ)	Consonant with short SVF and anusvara
ಕ್ಕ (kka)	Biconsonantal conjunct
ಕ್ಕಂ (kkaṁ)	Biconsonantal conjunct with anusvara
ಕ್ಕು (kku)	Biconsonantal conjunct with Short SFV
ಕ್ಕುಂ (kkuṁ)	Biconsonantal conjunct with Short SFV and anusvara
ಕ್ಕ್ಕ (kkka)	Tri-consonantal conjuncts
ಕ್ಕ್ಕಂ (kkkaṁ)	Tri-consonantal conjuncts with anusvara
ಕ್ಕ್ಕು (kkku)	Triconsonantal conjunct with Short SVF
ಕ್ಕ್ಕುಂ (kkkuṁ)	Biconsonantal conjunct with Short SVF and anusvara
ಗಳ್ (gaḷ)	Consonant with Ardhakshar (Half Consonant)

*Biconsonantal conjuncts (One Dead Consonant) *Tri-consonantal conjuncts (Two Dead Consonants) *Half Consonant (consonant core without embedded vowel).

Table 3. Grammar kansyllable

Component	Members
Variables(V)	{ANUSVARA, LONG_SFV, LONG_VOWEL, HALANT, SHORT_SFV, SHORT_VOWEL, NEWLINE, SPACE, VISARGA, CONSONANT, error}
Terminals (T)	{anuswara, anuvisarga,dgunisu, hgunisu, dheerga, empty, gunisu, hraswa, next, nl, space, start, va, vh, visarga}
Start Symbol(S)	Start
Production set (P)	• {start→ hraswa | dheerga | vh • vh→CONSONANT HALANT next | va • next→ CONSONANAT HALANT CONSONANT gunisu | va • va→ CONSONANT gunisu • gunisu → hgunisu | dgunisu • anuvisarga → anuswara | visarga • hgunisu → SHORT_SFV anuvisarga • anuswara → ANUSWARA • dgunisu → LONG_SFV anuvisarga • visarga → VISARGA • hraswa → SHORT_VOWEL anuvisarga • dheerga → LONG_VOWEL anuvisarga }

Table 4. Proposed Syntax Directed Translation

Rule No.	Production Rule	Semantic Action
7	vh→ CONSONANT HALANT next	**if** next is not empty, then {vh.val = CONSONANT.val + HALANT.val + next.val; Recognize a letter as Biconsonantal conjunct or Triconsonantal conjunct depending on the next.val} **else** {vh.val= letter recognized immediate before+ CONSONANT.val + HALANT.val ; Recognize a letter as Consonant with Half Consonant}
8	vh → va	vh.val=va.val, Append letter Property as Consonant. Recognize letter as a consonant
9	next→ CONSONANT HALANT CONSONANT gunisu	**if** gunisu is empty, then { next.val = CONSONANT .val + HALANT.val + CONSONANT.val } **else** { next.val = CONSONANT.val + HALANT.val + CONSONANT.val + gunisu.val } Append letter Property as Triconsonantal conjunct
10	next→va	next.val=va.val Append letter property as Biconsonantal conjunct
11	next→ space	next.val=space.val
12	next→ nl	next.val = nl.val
13	va→ CONSONANT gunisu	**if** gunisu is not empty, then{va.val=CONSONANT.val + gunisu.val} **else** {va.val = CONSONANT.val}
14	gunisu→hgunisu	{gunisu.val=hgunisu.val}
15	gunisu→dgunisu	{gunisu.val=dgunisu.val}
16	gunisu→ anuvisarga	{gunisu.val= anuvisarga.val}
17	hgunisu → SHORT_SFV anuvisarga	**if** anuvisarga is empty then {hgunisu.val = SHORT_SFV.val} **else** { hgunisu.val = SHORT_SFV.val + anuvisarga.val}

		Append letter property as SHORT_SFV
18	dgunisu → LONG_SFV Anuvisarga	**if** anuvisarga is empty, then { dgunisu.val = LONG_SFV.val} **else** {dgunisu.val = LONG_SFV.val + anuvisarga.val} Append letter Property as LONG_SFV
19	hraswa→ SHORT_VOWEL anuvisarga	**if** anuvisarga is empty, then {hraswa.val = SHORT_VOWEL.val} **else** {hraswa.val = SHORT_VOWEL.val + anuvisarga.val} Append letter property as a short vowel Recognize a letter as a short Vowel.
20	dheerga→ LONG_VOWEL anuvisarga	**if** anuvisarga is empty then{ dheerga.val= LONG_VOWEL.val } **else** { dheerga.val= LONG_VOWEL.val + anuvisarga.val } Append letter Property as long vowel, Recognize a letter as long vowel.
21	anuvisarga→ anusvara	Anuvisarga.val = anusvara.val
22	anuvisarga→ visarga	Anuvisarga.val= visarga.val
23	anuvisarga→empty	anuswara.val=<empty> Initiate letter as empty
24	anuswara→ ANUSVARA	anuswara.val= ANUSVARA.val Initiate letter with anusvara
25	visarga→VISARGA	visarga.val= VISARGA.val Initiate letter with visarga
26	nl→ NEWLINE	nl.val=<empty>
27	space → SPACE	space.val=<empty>
28	empty → <em5pty>	empty.val=<empty>

Input: Unicode Kannada Text Stream
Output: Text steam with annotated Properties
Method:
//1. Recognize Consonants, Biconsonantal conjuncts, //Triconsonantal conjuncts, short vowels, long vowels

//2. Recognize the Presence of optional secondary vowel //forms associated with all forms of consonants

//3. Recognize the presence of optional Anusvara or //Visarga associated with all forms of consonants and //vowels.

Step 1: BEGIN
While not the end of the Unicode text stream then
STEP1: Read the next Unicode Character
If SHORT_VOWEL then
Recognize the property of letters as short vowels go to step 3
If LONG_VOWEL then
Recognize the property of letter as long vowel go to step 3
If CONSONANT then read next Unicode Character
If HALANT then read the next Unicode character
If NEWLINE or SPACE
Recognize Property as Half-Consonant attach this to consonant attached in the previous Step and go to Begin
Else if CONSONANT then read next Unicode Character
If NEWLINE or SPACE then
Recognize Property as Biconsonantal Conjunct and go to go to Begin
Else If HALANTH then read next Unicode Character
If CONSONANT then read next Unicode character
Recognize property as Triconsonantal Conjunct and go to Step 2
Else
Recognize property as consonant go to Step2
STEP2: Read the next Unicode Character
If SHORT_SVF then
Recognize property as short SVF
Go to Step 3
If LONG_SVF then
Recognize property as short SVF
Go to Step 3
If NEWLINE or SPACE then
Go to Step 3
STEP3: Read the next Unicode Character
If ANUSVARA then
Recognize property as anuswara
Go to Begin
If VISARGA then
Recognize property as VISARGA
Go to Begin
If NEWLINE or SPACE then
Go to Begin
END

3. Results and Discussion

3.1. Results

The results are tested on a made dataset by collecting articles from the world's first Kannada online magazine and the website http://vishvakannada.com/. The data cleaning is carried out at the lexical analysis phase with a scanner using the token specification descriptions and regular expressions implemented using PLY lexer. All non-Kannada letters, numeric symbols, and punctuation marks, such as spaces, new lines, etc., are filtered out before tokenizing the text. The pre-processed text is made available to PLY implemented parser for annotation and classification. The experiments are conducted on files of varied sizes and in different genres written by various authors.

One such test was conducted with 1994 lines of text having 10873 words using 22518 Kannada letters saved in a Unicode text file format. The results are manually verified against the noted ground truth. The proposed method annotated all the letters and classified them with 100 per cent accuracy. The tabulated results in Table 5 are matching with the expected outcomes. The syllable annotation and Akshara classification reported by the syllabification process are as expected. Figure 1 represents the parse tree representing parser moves to derive the syllable ಕ್ಕುಂ having Unicode consonant core: K_d+K_d+K+u+M, where K_d is a dead consonant, 'K' is live consonants and 'u' is the SFV and 'M' is Anusvara.

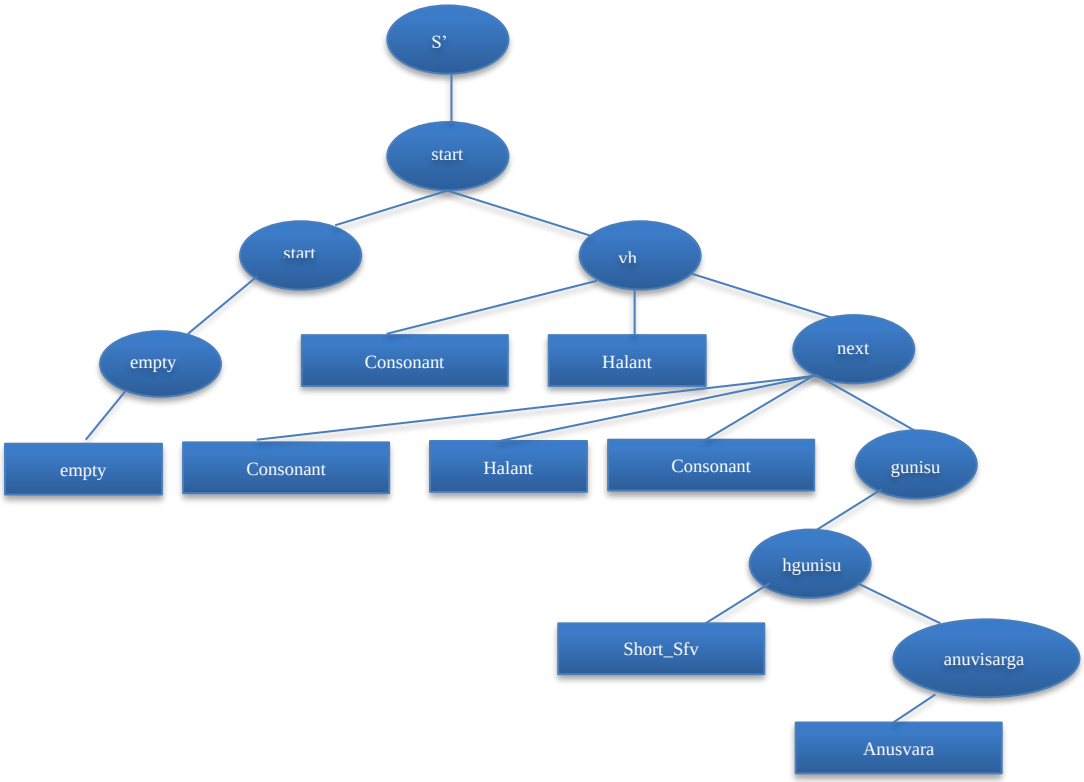

Fig. 1 A Parse tree to derive the Kannada Letter ಕ್ಕುಂ (kkkuṁ)

Table 5. Proposed Syntax Directed Translation Performance

Raw Input file Character Count (Unicode Sequence Count)	Letters Recognized and Annotated by Grouping the Unicode Sequence using the Proposed Method (Results are Matched Accurately with Ground Truth)
155784	67472
85012	37157
74710	32816
69020	30315
6647	2878
183	79

Figure 2(a) illustrates how a biconsonantal conjunct has one dead consonant and a live consonant followed by SFV and an anusvara. Similarly, Figure 2(b) illustrates the annotation process built using a bottom-up approach for the triconsonantal conjunct having two dead consonants and a live consonant followed by an SFV and then by an anusvara.

Figure 2(c) is the snippet of an annotated parse tree in which a live consonant is followed by an SFV and then by an anusvara. The classification of syllables is made at a node labelled 'vh'. The decision depends on the attribute value held by nodes labelled 'next'. The different scenarios at node 'next' are listed as follows:

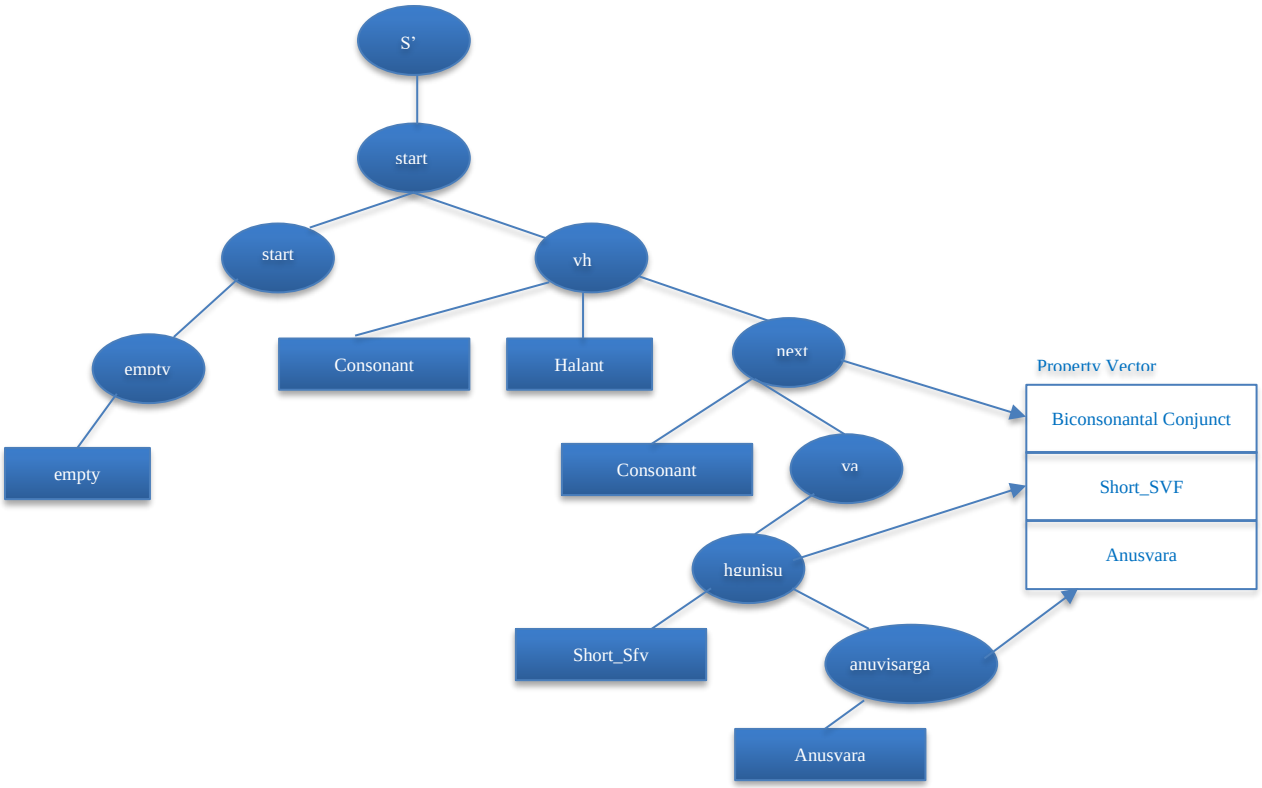

Fig. (2a) Illustrates the annotation process of a biconsonantal conjunct

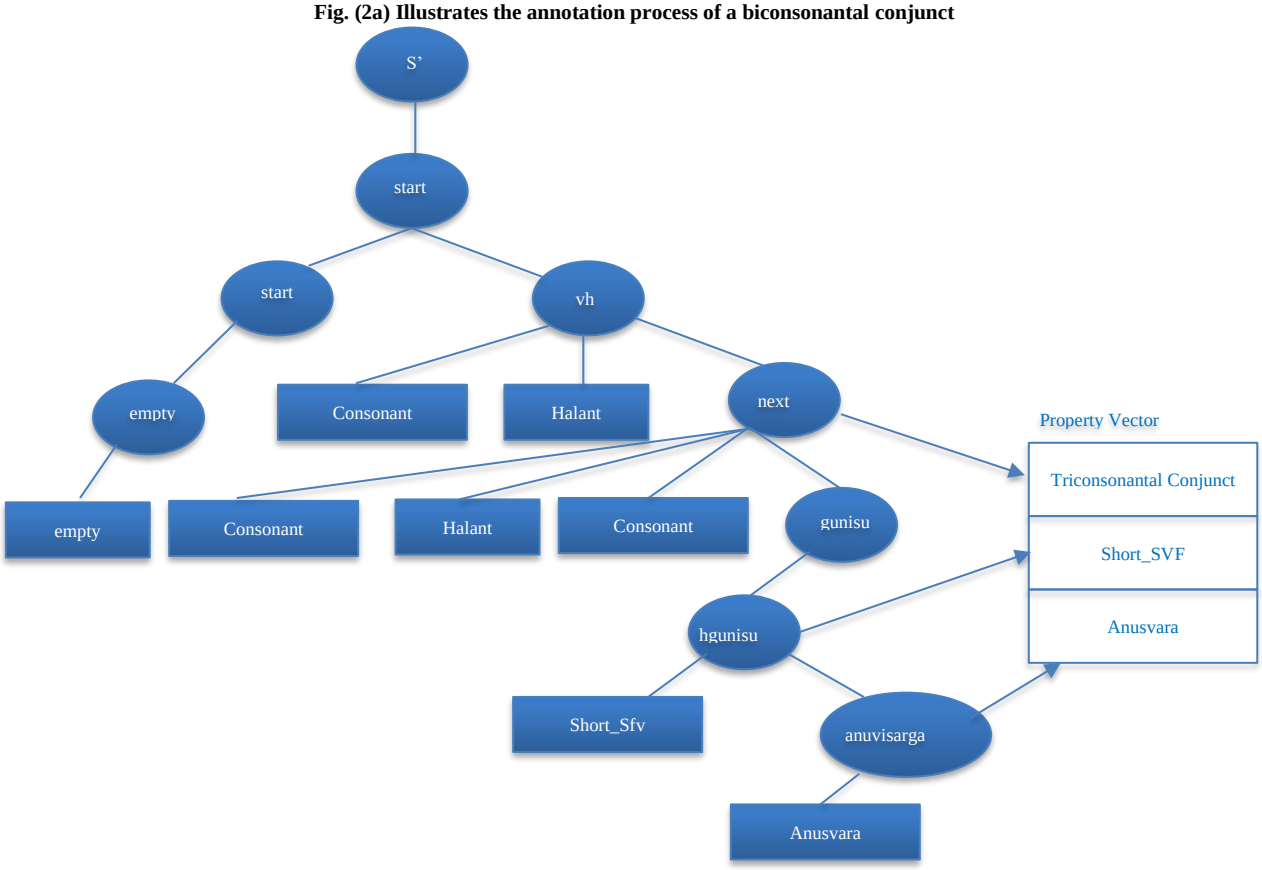

Fig. (2b) Illustrates the annotation process for the triconsonantal conjunct

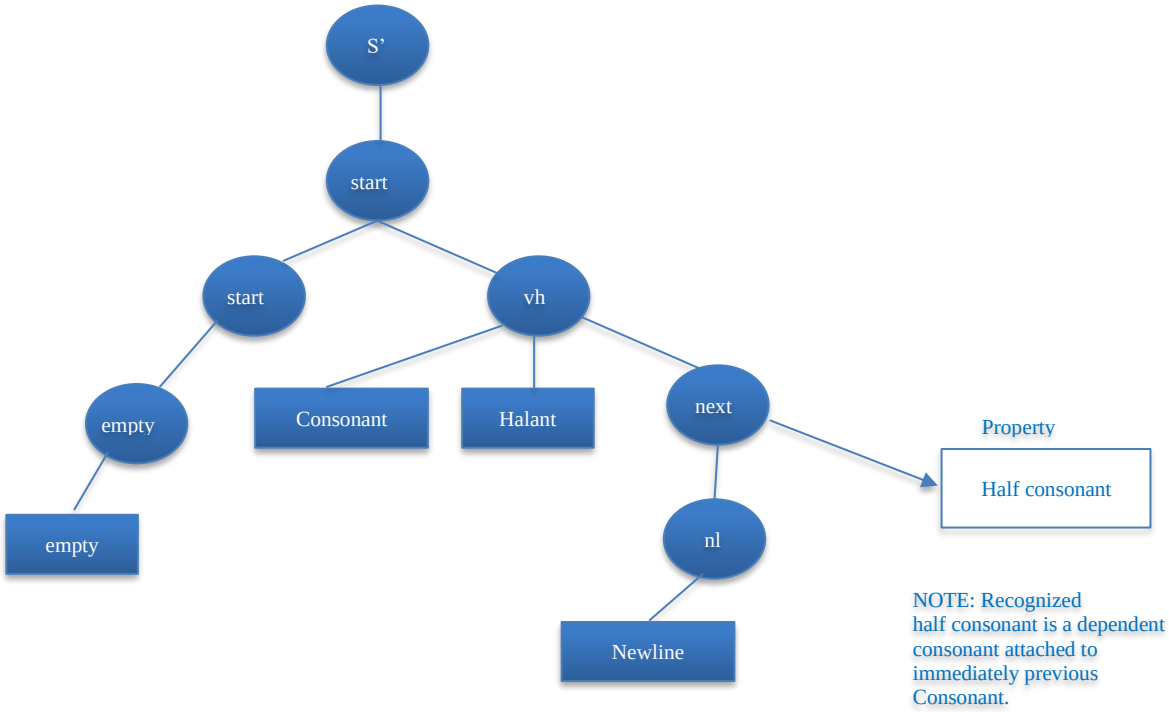

Fig. (2c) Parse tree to derive half consonants

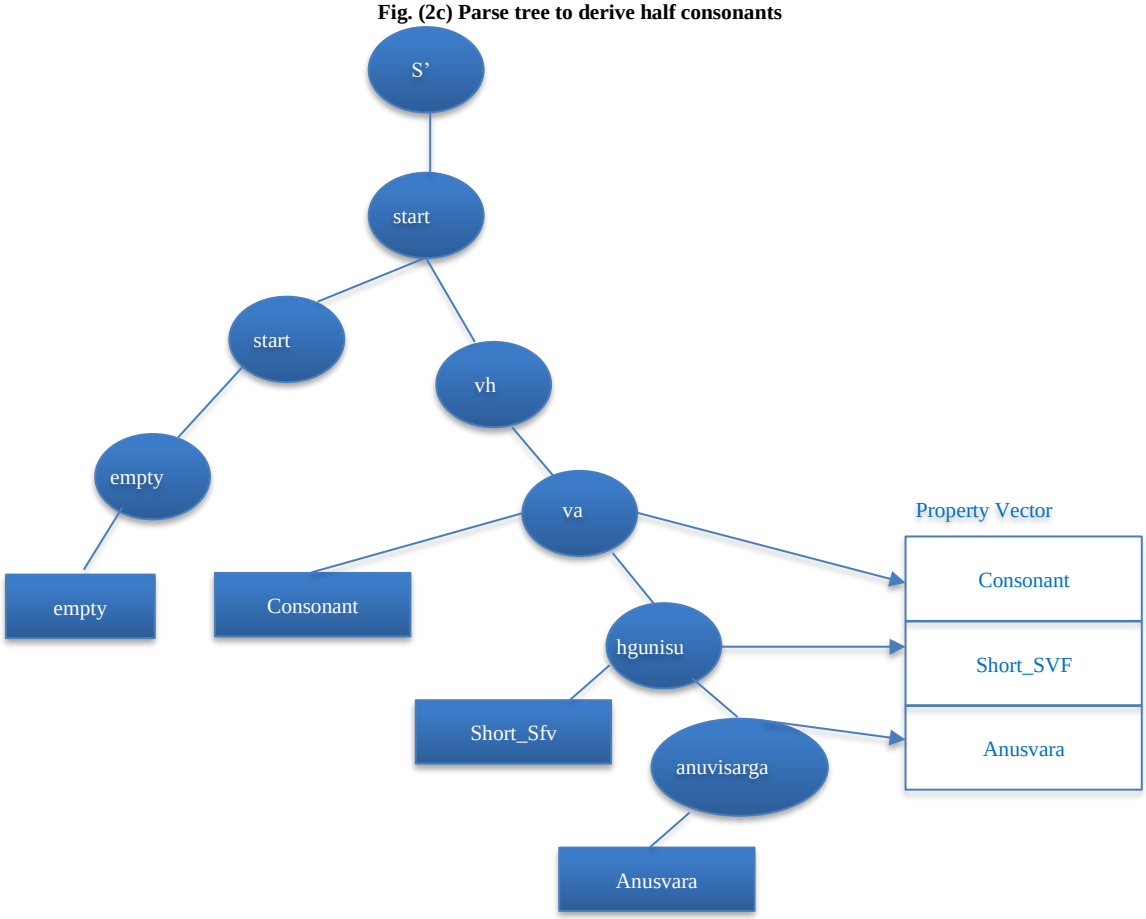

Fig. (2d) Parse tree to annotate simple consonants

- If ‘next’ is reduced by ‘CONSONANT HALANT CONSONANT’ with an optional dependent vowel and /or optional anuvisrga (Rule 9 of SDT), then the syllable is classified as a triconsonantal conjunct.
- If ‘next’ is reduced by ‘va’ having only CONSONANT with an optional dependent vowel and /or optional anuvisrga (Rule 10 of SDT), then the syllable is classified as a biconsonantal conjunct.
- If ‘next’ is reduced by either newline or space (Rule 11 or 12 in SDT) with an empty attribute value, the syllable is classified as a dead consonant. In the language, dead consonants have no independent existence, which do not yield any meaning if they were used independently. They might be thought of as dependent consonants, either part of Consonant conjuncts or appear as end syllables, making the case for akharas that appear like ‘ಗಳ್’ or 'ಮೇಣ್'. Finally, Figure 2(d) shows the handling of simple consonants with optional SFV and/or optional anuvisarga. In this case, ‘vh’ doesn’t reach out for ‘next’, rather, an alternate path defined by rule 8 in SDT is opted.

3.2. Applications

3.2.1. Syllable segmentation

The SDT proposed here can offer better syllable segmentation than the orthographic syllabifier of the *indicNlp library*. Following is the illustration: >>> input='"ಕನ್ನಡ ಒಂದು ಸುಂದರ ಭಾಷೆ. ಜಯ ಹೇ ಕರ್ನಾಟಕ ಮಾತೆ."' *[“kannaḍa oṁdu suṁdara bhāṣe.jaỳa hē karnāṭaka mate” (Kannada is a beautiful language. Victory to you Mother Karnataka)]*

>>> from indicnlp.syllable import syllabifier >>> lang='kn'

>>>print('|'.join(syllabifier.orthographic_syllabify(input, lang)))

ಕ|ನ್ನ|ಡ|||ಒ|ಂದು|||ಸು|ಂದ|ರ|||ಭಾ|ಷೆ|.|ಜ|ಯ|||ಹೇ|||ಕ|ರ್ನಾ|ಟ|ಕ|||ಮಾ|ತೆ|.

As shown, the major issue is the position of the anusvara. Anusvara is segmented as prefixed to later syllables, but it actually needs to appear as a postfix to the earlier syllable. The manner in which output is rendered disagrees with the natural law of the Kannada language. This issue is addressed using the proposed SDT. The output generated by the proposed method is given below:

('ಕ',['Consonant'],

'ನ್ನ',['Biconsonatal conjunct'],

'ಡ',['Consonant'],

'ಒಂ',[‘Anusvara','Short_vowel’],

'ದು',['Short_SFV','Consonant'],'

ಸುಂ',['anusvara',’short_SFV’,'Consonant'], 'ದ',['Consonant’],

'ರ',['Consonant’],

'ಭಾ',['Long_SFV','Consonant'],

'ಷೆ',['Short_SFV','Consonant’],

'ಜ',['consonant'],

ಯ',['consonant'],

'ಹೇ', ['long_SFV','consonant'],

'ಕ', ['consonant'],

'ರ್ನಾ', ['long_SFV','biconsonantal conjunct’],

'ಟ',['consonant'],'

ಕ', ['consonant'],

'ಮಾ’,['long_SFV','consonant'],

'ತೆ',['Short_SFV', 'consonant'])

3.2.2. The production rules set of ABNF grammar suggested for Syllable segmentation at [9] is of the form

V[m] | CHC[v][m] | CH. In the abstract rule mentioned above, there is no provision for separating short and long dependent vowels and Anusara/visarga. Further, as per the available rules, the identification of a single dead consonant as an independent syllable is possible. Then, the rule defined to derive the consonant conjuncts permits the multi-consonantal conjuncts. However, in the case of Kannada, a combination of more than two consonants does not occur [22]. The proposed SDT is designed to handle this condition satisfactorily.

3.2.3. Syllable Counts

Applications such as music composers and verse/poetry prosody processors can scan the attribute sets to assign ‘matra’ (time intervals) to syllables. These syllable counts are time duration counts that can be used to predict or to compose music tunes for the lyrics in any language. This also can help compare phonetic features of the different languages**.**

3.2.4. Transliterations

Processing orthographic syllable level processing yielded better results than word-level, phrase level and morpheme-level transliteration of multiple language pairs [5]. The precise syllabification achieved through the proposed SDT is expected to improve the efficiency of such systems.

3.2.5. Translations

Detailed syllabification can help in resolving possible disambiguation.

4. Conclusion

The present work is part of Kannada poetry text analysis and prosody classification research work. Syllabification is the primary task that needs to be carried out for matra assignment, followed by grouping, pattern search, and

classification. In this context, the results obtained by the proposed SDT for syllabification of Kannada Unicode text are accurate, and no misinterpretations or misrepresentations were found. The objective of building an efficient tool for syllabification in the Kannada language is implemented successfully. The results obtained by the proposed method are used to implement a syllable property-based key generation symmetric cryptographic algorithm for the Kannada language text by the same authors. Non-Kannada syllables, digits, punctuation marks, and special symbols (except newline and space) are filtered at the tokenization step and are not included in the grammar production set. The arkavtthu and dead consonants are treated as biconsonantal conjuncts, playing similar roles in the matra distribution process. A new concept of a dependent consonant is coined in the work.

The work proposed does not differentiate consonants with slight breathing or weak aspiration (Alpaprana) from consonants with hard breathing or strong aspiration (Mahaprana), as they are treated the same in text prosody processing. Punctuation marks may play an important role in other applications; in that case, the rules can be augmented appropriately to include them in the syllabification process.

References

[1] Languages of the World, Ethnologue, 1951. [Online]. Available: http://www.ethnologue.com

[2] C. V. Srinatha Sastry, UNICODE for Kannada, (U+0C80 to U+0CFF), UNICODE, 2016. [Online]. Available: https://www.unicode.org/L2/L2003/03068-kannada.pdf,

[3] Manoj K. Chinnakotla, Om P. Damani, and Avijit Satoskar, "Transliteration for Resource-Scarce Languages," *ACM Transactions on Asian Language Information Processing*, vol. 9, no. 4, pp. 1-30, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[4] Loitongbam Gyanendro Singh, Lenin Laitonjam, and Sanasam Ranbir Singh, "Automatic Syllabification for Manipuri language," *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, pp. 349-357, 2016. [Google Scholar] [Publisher Link]

[5] Anoop Kunchukuttan, and Pushpak Bhattacharyya, "Orthographic Syllable as the Basic Unit for SMT Between Related Languages," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, pp. 1912-1917, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[6] Prakash Padakannaya, "Indian Orthography and Teaching How to Read: A Psycholinguistic Framework," *Psychological Studies*, vol. 49, no. 4, pp. 262-271, 2004. [Google Scholar]

[7] Anoop Kunchukuttan et al., "Leveraging Orthographic Similarity for Multilingual Neural Transliteration," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 303-316, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Anoop Kunchukuttan, IndicNLP Library, 2014. [Online]. Available: https://anoopkunchukuttan.github.io/indic_nlp_library/

[9] Indic Layout Requirements, W3C, 2020. [Online]. Available: https://www.w3.org/TR/ilreq/#h_indic_orthographic_syllable_boundaries

[10] PLY (Python Lex-Yacc), Dabeaz, 2024. [Online]. Available: https://www.dabeaz.com/ply/

[11] Mohd Sanad Zaki Rizvi, 3 Important NLP Libraries for Indian Languages You Should Try Out Today!, Analytics Vidhya, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/01/3-important-nlp-libraries-indian-languages-python/

[12] Sonal Kulkarni-Joshi, "Linguistic History and Language Diversity in India: Views and Counterviews," *Journal of Biosciences*, vol. 44, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[13] Richard Sproat, "A Formal Computational Analysis of Indic Scripts," *International Symposium on Indic Scripts: Past and Future*, Tokyo, pp. 1-32, 2003. [Google Scholar] [Publisher Link]

[14] Kuche Anurag, Kuche Bhavani Priya, and Karthik Kashyap, "Transliteration of Kannada Text to English Text," *International Journal of Recent Engineering Research and Development*, vol. 3, no. 10, pp.19-23, 2018. [Publisher Link]

[15] Jong-Hoon OH, and Key-Sun CHOI, "Machine Learning Based English-to-Koren Transliteration Using Grapheme and Phoneme Information," *IEICE Transactions on Information and Systems*, vol. E88-D, no. 7, pp. 1737-1748, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[16] M. Latha, M. Shivakumar, and R. Manjula, "Performance Analysis of Kannada Phonetics: Vowels, Fricatives and Stop Consonants Using LP Spectrum," *SN Computer Science*, vol. 1, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Sarika Hegde, K.K. Achary, and Surendra Shetty, "Statistical Analysis of Features and Classification of Alpha Syllabary Sounds in Kannada Language," *International Journal of Speech Technology*, vol. 18, pp. 65-75, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[18] Jeffrey Lidz, "The Grammar of Accusative Case in Kannada," *Language: Linguistic Society of America*, vol. 82, no. 1, pp. 10-32, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[19] Google Translate, Google.in, 2024. [Online]. Available: https://translate.google.co.in/?sl=auto&tl=en&op=translate

[20] R. Sproat," Multilingual Text Analysis for Text-to-Speech Synthesis," *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, Philadelphia, PA, USA, vol. 3, pp. 1365-1368, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[21] Transliteration, Karnataka.gov, 2022. [Online]. Available: https://ekannada.karnataka.gov.in/transliterate/

[22] Rev. F. Kittel, and M. Mariappa Bhat, Kittel Kannada Dictionary: Free Download, Borrow, and Streaming, Internet Archive, 2022. [Online]. Available: https://archive.org/details/kittel-kannada-dictionary