Kannada Sandhi Generator for *Lopa* and *Adesha Sandhi*

Musica Supriya † **Dinesh Acharya U** † **Ashalatha Nayak** † **Arjuna S R** ‡
† Department of Computer Science & Engineering
Manipal Institute of Technology
Manipal Academy of Higher Education, Manipal, Karnataka 576104 , India
‡ Department of Philosophy
Manipal Academy of Higher Education, Manipal, Karnataka 576104 , India
{musica.supriya, dinesh.acharya, asha.nayak, arjuna.sr}@manipal.edu

Abstract

Kannada is one of the major spoken classical languages in India. It is morphologically rich and highly agglutinative in nature. One of the important grammatical aspects is the concept of *sandhi*(euphonic change). There has not been a *sandhi* generator for Kannada and this work aims at basic *sandhi* generation. In this paper, we present algorithms for *lopa* and *Adesha sandhi* using a rule-based approach. The proposed method generates the *sandhied* word and corresponding *sandhi* without any help of dictionary. This work is significant for agglutinative languages especially to Dravidian languages and can be used to enhance the vocabulary for language related tasks.

1 Introduction

Kannada is one of the Dravidian Languages spoken predominantly by people of Karnataka. The Dravidian languages are highly agglutinative and morphologically rich in nature (Shashirekha and Vanishree, 2016). Euphonic change known as *sandhi* (Kumar et al., 2009) is quite common in this language.

Sandhi occurs at the character level between two valid words of a particular language based on a set of rules. It also occurs between a root word/nominal stem and suffix as well. As Huet (2009) rightly mentions the Sanskrit grammarians' point that the *sandhi* is a mandatory action in the case of compound words in Sanskrit, this condition aptly applicable to Kannada as well.

The classification of *sandhi* can be done as internal *sandhi* and external *sandhi*. The euphonic change that occurs, between a root word and suffix, and in the case of construction of a compound word, is considered as internal *sandhi*. For instance, $hU + annu$[1] $= hUvannu$ [2] which means *the flower (as an object of a verb)*. In this case, the first word *hU* is a nominal stem and *annu* is a suffix. It undergoes euphonic change and *vakAra-Agama sandhi* occurs here. Hence, it is an internal *sandhi*. An example for another internal type is - $parama + ISvara = parameSvara$ $[supreme + Lord = supreme\ Lord]$. Here both *parama* and *ISvara* are words forming the compound word $parameSvara$. If *sandhi* occurs between the characters present in two different words, then it can be called as external *sandhi*. For example: $illi + ixxAlYeV = illixxAlYeV$ $[here + she\ is = she\ is\ here]$. Both the words - $illi$ and $ixxAlYeV$ are two different words.

Another way of classification is Kannada *sandhi* and Sanskrit *sandhi*. This can be classified based on the words present in the usage. As described by Kittel (1920), Kannada has borrowed many words from Sanskrit. It is possible to identify *sandhi* corresponding to pure Kannada words or a combination of Kannada and Sanskrit words. If both the *pUrvapaxa*(first word) and *uwwarapaxa*(second word) are chosen to be Kannada words, then, Kannada *sandhi* occurs. For instance, $nAnu + illi = nAnilli$ $[I + here = I\ am\ here]$. Both the words are Kannada words and the Sanskrit *sandhi* is not applicable here. The non-application of *guNa sandhi* (a Sanskrit *sandhi*) affirms that through the previous instance. If either or both are Sanskrit words, then Sanskrit *sandhi* occurs. For instance, $deva + ISanu = deveSanu$ $[Lord + Lord\ of = Lord\ of\ all\ Lords]$. Both the words are borrowed words and will follow *guNa sandhi* of Sanskrit and, not any Kannada *sandhi*.

In this paper, our focus is on Kannada *sandhi*. There are mainly three Kannada *sand-*

[1] Accusative case marker

[2] We have used WX notation given by Gupta et al. (2010) throughout this paper to represent Kannada words.

Proceedings of the 18th International Conference on Natural Language Processing, pages 347–351
Silchar, India. December 16 - 19, 2021.

his: *lopa*(elision) *sandhi*, *Agama*(addition) *sandhi* and *Adesha*(substitution) *sandhi* as mentioned by Sharma (2015).

- *Lopa(elision) sandhi*: When *pUrvapaxa* is ending with a Kannada vowel , *uwwarapaxa* begins with a Kannada vowel, then euphonic union of this will cause the vowel at the *pUrvapaxa* to be eliminated and this word be combined with the *uwwarapaxa* to form the new word.
  Example: $nAnu + illi = nAnilli.$ $[I + here = I\ am\ here]$
  Here the last vowel of *pUrvapaxa*, *u* is eliminated from *pUrvapaxa* and then joined to the *uwwarapaxa* to form the new word.

- *Agama(addition) Sandhi*: When *pUrvapaxa* is ending with a Kannada vowel , *uwwarapaxa* begins with a Kannada vowel, then euphonic union of this will cause the addition of *y* or *v* at the beginning of *uwwarapaxa*. This combination leads to *Agama sandhi*.
  Example: $maneV + ixu = maneVyixu$ $[house + this = this\ is\ house]$. Here the first vowel *i* of *uwwarapaxa* is prefixed with *y* and then joined to form the new word.

- *Adesha(substitution) Sandhi*: When *pUrvapaxa* is a Kannada word, *uwwarapaxa* begins with either *k* , *w* or *p* then it is replaced with *g*, *x*, *b* respectively .This combination of *pUrvapaxa* and *uwwarapaxa* leads to *Adesha sandhi*.
  Example: $malYeV + kAla = malYeVgAla$ $[rain + season = rainy\ season]$. Here *uwwarapaxa* begins with *k*, in the euphonic change, it is replaced with *g* to form the new word.

The Kannada *sandhi* can be further sub-categorized as *svara*(vowel) *sandhi* and *vyaFjana*(consonant) *sandhi*. The *svara sandhi* has vowel at the end of the *pUrvapaxa* and at the beginning of *uwwarapaxa*. *vyaFjana sandhi* must have a consonant character at the beginning of *uwwarapaxa* and there can be any character *svara* or *vyaFjana* at the end of the *pUrvapaxa*.

This classification helps us to apply the *sandhi* rules unambiguously to some extent. However, in some cases, we can see some overlapping of the rules of the *sandhi*. One such instance is $nAnu + illi = nAnilli$ $[I + here = I\ am\ here]$. Though the sandhi rule of *vakAra-Agama sandhi* explained by Sharma (2015) is applicable, we have to choose *lopa sandhi*, based on the usage. There are no research or reason present for the preference of *lopa sandhi* over the *Agama sandhi*. Due to this complexity, we have excluded the *Agama sandhi* in this paper.

The work on *sandhi* generation for Kannada has not been successfully carried out. There are *sandhi* generator tools developed for basic Malayalam by Kleenankandy (2014) and for Sanskrit by Amba (2002), Huet (1994) and many others. We may find some works in *sandhi* generation for other Indian languages (Nirmala and Kalpana, 2015) as well. The *sandhi* splitter tasks were carried out for Kannada by Shashirekha and Vanishree (2016), but generator was left out due to the complexity and ambiguity involved. We are addressing this issue by proposing a novel idea by defining the algorithm necessary to generate *sandhi* formation for the given input Kannada words using a rule based approach. The *sandhi* generator is useful for students who wish to learn the concept of euphonic change in Kannada and to researchers working on NLP applications for the tasks like morphological analysers and Machine Translation. The paper is organized as follows: Section 1 introduction, section 2 literature review, section 3 methodology, section 4 result analysis and section 5 conclusion and future scope followed by references.

2 Literature Survey

Kannada spell checker and *sandhi* splitter work was carried out by Murthy et al. (2017) making use of transliterated dictionaries which stored the words and affixes. The *Agama sandhi* splitter was developed by Shashirekha and Vanishree (2016) using a rule based approach with manual annotation of suitable words and their affixes. There are other Kannada *sandhi* splitter works but here we have reviewed a few as our focus is on generation than splitting. Kleenankandy (2014) has implemented *sandhi*-rule based compound word generator for Malayalam, the basic *sandhi* rules for Malayalam was addressed along with supplementary details for words to identify the *sandhi*. This work was semi-automatic and required human intervention to resolve ambiguities. Significant work on Sanskrit word segmentation was carried out by Huet (2003).

The disambiguation of a given word was performed using a rule-based approach.

Though there are *sandhi* generator tools developed for Sanskrit by Amba (2002), Huet (1994), Sachin Kumar and many others, we see that there has not been a successful *sandhi* generator for pure Kannada *sandhi*.

3 Methodology

The implementation of a basic *lopa* and *Adesha sandhi* using regular expression is presented in this section. We have excluded the internal *sandhi* (root + suffix) and considered only compound words and external *sandhi*. As we mentioned earlier in the introduction section, some rules of *Agama sandhi* overlaps with the rules of *lopa sandhi*. Let us consider the *yakAra-Agama sandhi* as an example - *maneV + alli = maneVyalli [house + locative case marker = in the house]*. The rules for *lopa sandhi* and *Agama sandhi* overlap here and there is no semantic information which can distinguish and avoid the overlap. By convention and practical usage, *maneVyalli* is the expected output, whereas due to the overlapping rules, our system generates *maneli* as the output. Though the word *maneli* is also correct in colloquial Kannada and the meaning is also exactly similar as *maneVyalli*, we may not find the word *maneli* in formal usage. Hence, we have skipped writing the algorithms to *Agama sandhi* at this stage.

The block diagram of the methodology is shown in Figure 1.

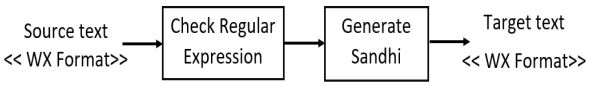

Figure 1: Block diagram of the system for *sandhi* generation

The user inputs two Kannada words in WX notation. The *Check Regular Expression* module will validate the input words to check if the words follow the pattern suitable to perform *lopa* or *Adesha sandhi*. If it follows the pattern, then the inputs are passed into *Generate Sandhi* module. In this module, the rules are defined. The rules to perform *lopa* and *Adesha sandhi* are different. We have referred mainly to the Kannada grammar book by Sharma (2015) and inspired by the Sanskrit Sandhi works of Amba (2002) and Huet (1994) to obtain these rules and implementation was performed following the same. The generated output is the *sandhi* word in WX notation. However, we may render the WX input in UTF-8 (*Kannada scripts*) to ease the process of learning or understanding the *sandhi* concept, using the existing standard transliteration schemes.

3.1 Algorithm for *lopa* and *Adesha Sandhi*

The algorithm for *lopa sandhi* and *Adesha sandhi* is shown in Algorithm 1. As we mentioned, we have followed rule based approach and is based on the morphological rules of Kannada by Sharma (2015). The characters are extracted using extract function as defined in Algorithm 2 and are joined with respect to the predefined morphological rules. The input words are analysed to check which *sandhi* can be applied. Once we decide on the *sandhi*, unification of characters can be done based on rules. No dictionary is used here and we assume that user has typed valid Kannada words as input. We would like to make use of a dictionary in the future to validate input words. As of now, if the rules are satisfied by the input words, the corresponding output is generated.

3.1.1 Definitions

Svara: Vowels in Kannada language - *a, A, i, I, u, U, q, eV, e, E, oV, o, O*
ktp: characters *k* or *w* or *p*

3.2 Implementation details

The text in WX form is checked against the regular expressions. There are four separate regular expressions two each for *lopa* and *Adesha sandhi*. For instance, in *lopa sandhi* the *pUrvapaxa* has to end with a *svara*. The corresponding regular expression is $[A-Za-z]*([aAiIuUqeEoO]|(eV)|(oV))\$$. We have made use of regular expression library[3] available for Python 3 to match these and implemented on Google Colab[4] platform.

4 Results and Analysis

We have checked in total for 386 unique pairs of Kannada words manually extracted from the data we requested from the work carried out by Reddy and Sharoff (2011). We were able to identify 255 pairs for *lopa sandhi* and 131 pairs for *Adesha sandhi*. The sample data is shown in table 1[5]. The

[3]https://docs.python.org/3/library/re.html
[4]https://colab.research.google.com
[5]We have not used diacritics in this table to highlight that the input to the system must be in WX notation.

Algorithm 1: Kannada *sandhi* generator for *lopa* and *Adesha sandhi*

Data: *string1, string2*
Result: *sandhi,output_string*
$len1 \leftarrow length(string1) - 1$
$len2 \leftarrow length(string2) - 1$
if *string1 ends with svara* **then**
  if *string2 begins with svara* **then**
    if *string1 ends with 'eV' or 'oV'* **then**
      $e1 \leftarrow extract(string1, 0, len1 - 2)$
    else
      $e1 \leftarrow extract(string1, 0, len1 - 1)$
    end
  $e2 \leftarrow string2$
  $k \leftarrow concatenate(e1, e2)$
  print *"Lopa Sandhi", k*
else if *string1 contains only characters* **then**
  if *if string2 begins with ktp* **then**
    if *first_char(string2)='N' or 'n'* **then**
      $temp \leftarrow extract(string1, 0, len1 - 1)$
      $e1 \leftarrow concatenate(temp, ' M')$
    else
      $e1 \leftarrow string1$
    end
  $e2 \leftarrow extract(string2, 1, len2)$
  $check \leftarrow string2[0]$
  if *check='k'* **then**
    $add \leftarrow ' g'$
  else if *check='w'* **then**
    $add \leftarrow ' x'$
  else
    $add \leftarrow ' b'$
  end
  $temp2 \leftarrow concatenate(e1, add)$
  $f \leftarrow concatenate(temp2, e2)$
  print *"Adesha Sandhi", f*
else
  print *"Check your input"*
end

generated words were examined by a Kannada linguistics expert for its accuracy.

The proposed work is useful in the generation of euphonic change without a dictionary and maybe applicable to other Dravidian languages. The summary of result and the accuracy is shown in table 2.

Algorithm 2: Extract substring

Data: *string, start_len, end_len*
Result: *substring*
$j \leftarrow 0$
for *each i ←start_len to end_len* **do**
  $m[j] \leftarrow concatenate(m, string[i])$
  $j \leftarrow j + 1$
end
$m[j] \leftarrow ' \backslash 0'$

Input1	Input2	Output string	Sandhi
nAnu [*I*]	illi [*here*]	nAnilli [*I am here*]	lopa
nanna [*me*]	iMxa [*by*]	nanniMxa [*by me*]	lopa
nanna [*my*]	Uru [*place*]	nannUru [*My place*]	lopa
kaliyaxe [*not*]	iruvuxu [*to learn*]	kaliyaxiruvuxu [*to not learn*]	lopa
xevaru [*God*]	iMxa [*by*]	xevariMxa [*by God*]	lopa
maneV [*house*]	keVlasa [*work*]	maneVgeVlasa [*house work*]	Adesha
adi [*foundation*]	kallu [*stone*]	adigallu [*foundation stone*]	Adesha
mE [*body*]	woVlYeV [*wash*]	mExoVlYeV [*wash the body*]	Adesha
hullu [*hay*]	kAvalu [*land*]	hullugAvalu [*hay land*]	Adesha
betta [*mountain*]	wAvareV [*lotus*]	bettaxAvareV [*mountain lotus*]	Adesha

Table 1: Sample output for *lopa* and *Adesha sandhi*

Sandhi	**Test pairs**	**Correct output**	**Accuracy**
Lopa	255	255	100%
Adesha	131	130	99.2%

Table 2: Summarized results for *lopa* and *Adesha sandhi*

In one of the instances of *Adesha sandhi*, "*kaN + kAvalu = kaNgAvalu [eye + security = surveillance*]", the generated output was *kaMgAvalu* which is not a valid *sandhied* word. In the case of *lopa sandhi*, all the generated outputs are valid outputs. Since the input words are not validated by a morphological analyser, any non-Kannada word which follows the pattern prescribed for *lopa* and *Adesha sandhi* will generate a *sandhi* output as per the rules is the major flaw in this tool. However, for a given input, no pair can undergo both *lopa* and *Adesha sandhi* at once. The evaluation of this task is performed manually by checking the generated output for the correct euphonic change and the type

of that *sandhi*, and is verified by a senior linguist and a few native speakers as well.

5 Conclusion and Future Scope

A basic *sandhi* generator for *lopa* and *Adesha sandhi* is carried out in this work. This work can be helpful for beginners and teachers who are engaged in the Kannada language and trying to understand/teach the concept of *sandhi*. In future, we would like to enhance this work by adding the non-overlapping and unambiguous rules for *Agama sandhi*. We will include the dictionary in the process to check the validity of the input words. We shall make a user-friendly interface to interact with the system. The generated words can be used to enhance the vocabulary for performing language related tasks viz. Kannada Machine Translation(MT) systems, Kannada Computational Linguistics tools etc. in future.

Acknowledgments

The first author is thankful to Prof. S. A. Krishnaiah, National Trust for Computation and Archival of Oriental Media (NTC-AOM), Udupi for verifying the accuracy of the generated outputs, Mrs. Shwetha Rai, Department of Computer Science & Engineering, Manipal Institute of Technology, Manipal for inputs in designing the algorithms and Dr. Siva Reddy, Mila, McGill University for sharing the PoS data for Kannada. We thank the anonymous reviewers for their constructive suggestions.

References

Kulkarni Amba. 2002. Samsaadhanii. =https://sanskrit.uohyd.ac.in/scl. Accessed: 04-Dec-2021.

Rohit Gupta, Pulkit Goyal, and Sapan Diwakar. 2010. Transliteration among indian languages using wx notation. In *KONVENS*.

Gérard Huet. 1994. The sandhi engine. =https://sanskrit.inria.fr/DICO/sandhi.fr.html. Accessed: 04-Dec-2021.

Gérard Huet. 2003. Lexicon-directed segmentation and tagging of sanskrit. In *in XIIth World Sanskrit Conference*, pages 307–325.

Gérard Huet. 2009. Sanskrit segmentation. In *South Asian Languages Analysis, round table*.

Rev. F Kittel. 1920. *Shabdamanidarpanam, Kesiraja's Jewel Mirror of Grammar*. Kanarese Mission Book and Tract Depository.

Jeena Kleenankandy. 2014. Implementation of sandhi-rule based compound word generator for malayalam. *2014 Fourth International Conference on Advances in Computing and Communications*, pages 134–137.

Anil Kumar, V. Sheeba, and Amba P. Kulkarni. 2009. Sanskrit compound paraphrase generator.

S Rajashekara Murthy, A. N. Akshatha, Chandana G Upadhyaya, and P. Ramakanth Kumar. 2017. Kannada spell checker with sandhi splitter. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 950–956.

K. Nirmala and M. K. Kalpana. 2015. Modern thamizh sandhi rules generator in nlp. In *SOCO 2015*.

Siva Reddy and Serge Sharoff. 2011. Cross language pos taggers (and other tools) for indian languages: An experiment with kannada using telugu resources.

Dr. Girish Nath Jha Rajneesh Kumar Pandey Sachin Kumar, Diwakar Mani. Sanskrit sandhi generator. `http://sanskrit.jnu.ac.in/sandhi/gen.jsp`. Accessed: 04-Dec-2021.

Vidhwan N Ranganatha Sharma. 2015. *Hosagannada Vyakarana*. Kannada Sahitya Parishat.

H. L. Shashirekha and K. S. Vanishree. 2016. Rule based kannada agama sandhi splitter. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 549–553.