Implementation of Sandhi Viccheda for Sanskrit WordsSentencesParagraphs

Article *in* International Journal of Computer Sciences and Engineering · September 2018

DOI: 10.26438/ijcse/v6i9.355360

2 authors:

Bhagyashree D Patil
Shram Sadhana Bombay Trust's College of Engineering and Technology
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Manoj Eknath Patil
Shram Sadhana Bombay Trust's College of Engineering and Technology
15 PUBLICATIONS **88** CITATIONS
SEE PROFILE

IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-9, Sept. 2018 E-ISSN: 2347-2693

Implementation of Sandhi Viccheda for Sanskrit Words/ Sentences/ Paragraphs

Bhagyashree Patil[1*], Manoj Patil[2]

[1]Department of Computer Engineering, SSBT's COET, Jalgaon, India
[2]Department of Computer Engineering, SSBT's COET, Jalgaon, India

[*]*Corresponding Author: pbhagyashree39@gmail.com, Tel.: 0257-2261096*

Available online at: www.ijcseonline.org

Accepted: 19/Sept/2018, Published: 30/Sept/2018

***Abstract*—**Sandhi is a technique in which joining of two or more words happens. Sandhi viccheda means splitting of word or sentences into its constituents. Languages like Hindi, Urdu, Marathi, Kannada, and Malayalam are used to implement the sandhi viccheda concept. Sanskrit is a language in which rules are used to form a word. These rules play an important role in sandhi viccheda process. So, to split the word or sentence rules are used in the system. As everyone knows BHAGWAT GEETA is a religious book which contains more complex words, sentences, which user can't understand. The proposed system uses those words, sentences from BHAGWAT GEETA for splitting purpose. For this, rules of sandhi viccheda like (vowels, consonant, and visarga) are used. The representation of the splits is shown using the directed acyclic graph to get a number of possible outputs.

Keywords— Sandhi splitting, lexical analysis, rules, sandhi viccheda algorithm, DAG

I. INTRODUCTION

Sanskrit language and its grammar had exerted an emphatic impact on Computer Science and related research areas. Many more researchers have realized the importance of Sanskrit language in NLP due to its computational grammar. Famous Vedas or religious books are defined in the Sanskrit language such as SamVed, YajurVed, RigVed, AtharvaVed, Ramayana, Mahabharata, and Bhagwatgeeta containing the complex words, sentences which user can't understand while reading. This data is the combination of two or more words so it is difficult to understand. For that, the system should be developed which will split those words/sentences/paragraphs and the user will get actual word of it.

In Bhagwatgeeta, there are many difficult words or sentences which a user can't understand easily. For that user must know from which words that complex word or sentence is made up of. So the main goal of this system is to provide an environment for users such that the root words of that difficult sentence or paragraph are obtained. The system uses the sandhi viccheda algorithm for splitting those words, sentences using DAG representation for possible outputs.

The proposed solution focuses on the sandhi viccheda of the difficult words, sentences, and paragraphs which are complicated to the user to understand. For this, the rules for sandhi viccheda are used in the algorithm so that any word in it is split into its constituent words. The input is traversed from left to right till the valid pada is found if so then splitting is done. The system is fulfilled by adding all the rules (like Vowel, Consonant, and Visarga) of sandhi viccheda.

The paper is organized as the follows: Section I depicts the introduction of Sanskrit and Sandhi viccheda. Section II contains the related work of sandhi viccheda, Section III contains the proposed approach for sandhi viccheda, Section IV contains the result and discussion of the proposed approach and Section V conclude research work with future directions.

II. RELATED WORK

Rupali Deshmukh et al., in [1], presented that natural language processing (NLP) is a field of artificial intelligence, and linguistics, concerned with the interactions between computers and human languages. The combination of two immediate sounds, that means union, is called as sandhi formation. Sandhi-splitting describes the process by which one letter is split to form two words. Sandhi-splitting is one subtask for a complete analysis of input text in NL. The proposed system recognizes sandhi word from an input text in Sanskrit; splits sandhi word and return what type of sandhi it is. Mrs. Namrata Tapaswi et al., in [2], presents a set of instructions for the formulation of LFG rules to parse Sanskrit. Some simple sentences are parsed which verifies the rules follow with grammatical construction of language.

Priyanka Gupta et al., in [3], showed that sandhi which means combining (of sounds). The Rule-based algorithm is proposed which gives an accuracy of 60-80% depending upon the number of rules to be implemented. Joshi Shripad S., in [4], presented that rule-based algorithm and rules for sandhi splitting of Marathi words. Latha R. Nair and S. David Peter, in [5], described that morphological analyzers are essential for any type of natural language processing works. As Malayalam like other Dravidian languages is an agglutinative language it needs a compound word splitter as a preprocessor. An algorithm has been developed and successfully used for splitting the compound words. 90% success has been established in the initial scrutiny of around 4000 compound words. The splitter is also used for developing and implementing a fully fledged morphological analyzer. Devadath V. V. et al., in [6], presented that the accurate execution of sandhi splitting is crucial for text processing tasks such as POS tagging, topic modeling, and document indexing. Different approaches to address the challenges of sandhi splitting in Malayalam are tried, and finally, they have thought of exploiting the phonological changes that take place in the words while joining. This resulted in a hybrid method which statistically identifies the split points and splits using predefined character level linguistic rules. Currently, the system gives an accuracy of 91.1%. M. Rajani Shree et al., in [7], showed the novel approach towards internal sandhi splitting for Kannada language. Using CRF (Conditional Random Fields) tool near about 1000are tagged and 400 raw split words are given to CRF tool. From the tagged list trained and testing data is produced. The accuracy using CRF tool for Kannada corpus nearly equals to 98.08, 92.91 and 95.43. Sachin Kumar, in [8], presented sandhi analyzer and splitter for the Sanskrit language. Rule base method and lexical lookup are used in the system. Preprocessing, the lexical search is done on examples before sandhi analysis process. Subanta analysis takes place respectively which looks up into lexicon for avyaya and verb words to remove them from processing. In preprocessing, punctuations are removed. Split words generated by reverse sandhi analysis are validated by subanta analysis. The system looks up for fixed word list of nouns, names, and MWSDD which if found in processing are remain unchanged. Shubham Bhardwaj et al., in [9], proposed SandhiKosh which evaluates the accuracy and completeness of Sanskrit sandhi tools. SandhiKosh uses five different methods to provide a corpus which is complete and able to give the performance of sandhi tools on Sanskrit literature. Amba Kulkarni and Sheetal Pokar et al., in [10], developed a parser which finds a directed tree giving a graph of nodes representing words and edges representing possible relations between them. Mimamsa constraint of akanksa is used to rule on nonsolution and sannidhi to prioritize the solution. Vaishali Gupta et al., in [11], presented that Urdu is a combination of several languages like Arabic, Hindi, English, Turkish, Sanskrit etc. Here a stemmer is used to convert a word to its root form. The suffix and prefix are removed from the word to extract the actual word from it. The accuracy of the system is up to 85% but there is drawback which is over stemming and under stemming. Anil Kumar, et al., in [12], presented an experience in Sanskrit for building an automatic paraphrase generator. 90% of compounds are used by paraphrase handler to produce correct paraphrases for which morphological analyzer is required. To know the meaning of the compound one should know relations between them. Pawan Goyal et al., in [13], proposed an approach called 'utsarga apavaada' for relation analysis and morphological analysis for deterministic finite automata (DFA). By giving Sanskrit text, root words identified by the parser and gives dependency relations between semantic constraints. The proposed parser creates semantic nets for Sanskrit paragraphs. Both the external and internal sandhi is implemented in the parser for Sanskrit words. Bhagyashree Patil et al., in [14] showed the comparative study of the different systems for sandhi viccheda of Sanskrit words.

III. PROPOSED APPROACH

In the proposed system, Sanskrit words, sentences and paragraphs are taken from Bhagwatgeeta. The system uses the rules of sandhi viccheda on the input to split the given data into its constituent words. The DAG representation is used to find out the possible ways of the output. The system works on all the sandhi viccheda rules. Here there are some examples of Bhagwatgeeta words:

A. *Examples of Bhagwatgeeta*

- भवान्भीष्मश्च → [भवान् भीष्मस् च] here the visarga rule has been applied, visarga added with च् becomes श्.
- प्रथमोऽध्यायः → [प्रथमस् अध्यायस्] here the vowel sandhi rule has been applied, ओ when added with अ becomes unchanged and avagraha is added.
- भगवद्गीता → [भगवत् गीता] here the consonant sandhi rule is applied, when hard consonant त् combines with vowel इ becomes द्.

Sandhi means joining of two or more words to get the meaning of the sentence. But the sandhi viccheda (vigraha) means to separate two or more words to get their constituent words. Following are the examples of the sandhi viccheda rules:

B. Rules

- Vowel Sandhi Rule: Consider the vowel sandhi rule that (आ) is formed when (अ, आ) combines with (अ, आ). Then in the example, विद्यालयः will be splitted into विद्या and आलयः. Here in this example the word, will be splitted at the point द्याल where the आ is formed when combined with the आ of आलयः word. Consider the next example that (अर्) is formed when (अ, आ) combines with (ऋ, ॠ). Example ग्रीष्मर्तुः is splitted at the point ष्मर्तुः where अर् is found when combined with ऋ.

- Consonant Sandhi Rule: Class soft consonant except nasal followed by hard consonant changes to 1st consonant of class. Consider the example एतत्पतति which has the एतद् and पतति as its constituent words. Here in this example द् is, a soft consonant combined with प् hard consonant forms the word एतत्पतति. The next example is न् at the end of a word changes to [**anusvara** and श् (palatal sibilant)] when followed by (च्, छ्) (palatal hard consonants). Consider the example कांश्चित् where anusvara and श् is converted to its original न् and च्. It means that कांश्चित् is splitted into कान् and चित्.

- Visarga Sandhi Rule: Consider the visarga sandhi rule that visarga is dropped when आ & visarga standing for आस् is followed by a vowel or soft consonant. For example, नरा गच्छन्ति splitted into नराः and गच्छन्ति. Here visarga has been dropped. The other example is नमस्ते; here visarga changes to स् when followed by त् or थ् and the output is splitted into नमः and ते.

The proposed system works for sandhi viccheda of complex Sanskrit words, sentences and paragraphs. Sandhi viccheda gives the user the root words of the complex word which help the user to understand what it means to. The system uses the sandhi viccheda algorithm and DAG representation for the possible outcomes came in the output. Graphs are networks consisting of nodes connected by edges or arcs. In directed graphs, the connections between nodes have a direction, and are called arcs; in undirected graphs, the connections have no direction and are called edges. Algorithms in graphs include finding a path between two nodes, finding the shortest path between two nodes, determining cycles in the graph (a cycle is a non-empty path from a node to itself), finding a path that reaches all nodes (the famous "traveling salesman problem"), and so on. Here the Networkx package has been taken into account to get the DAG representation of the result i.e. the output is displayed in possible 10 paths using the shortest path.

C. Sandhi Viccheda Algorithm

- Procedure Sanskrit Lexical analyzer
  Input: Sanskrit word / sentence / Paragraph
- Action: Traverse the word/sentence, splitting it (or not) at each location to determine all possible valid splits.
- Traverse from left to right
- Using recursion (with memoization), assemble the results of all choices
- To split or not to split at each phoneme
- If split, all possible left/right combination of phonemes that may result
- Once split, check if the left section is a valid pada (use level 1 tool to pick pada type and tag morphologically).
- If the left section is valid, proceed to split the right section
- At the end of step 2, the system will have all possible syntactically valid splits with morphological tags
- Output: All semantically valid sandhi split sequences

In the Algorithm of the system, the Sanskrit word or sentence has been given as an input to the system. The word or sentence then traversed from left to right till all the splits are not found. When the left section has been completed finding all the splits then right section has been started, after that user will get all the valid splits with morphological tags.

IV. Results and Discussion

Implementation of the system is done using the Python environment. The version used for Python is 2.7 and the dataset used is Bhagwatgeeta. While implementing the result, it is in the form of Sanskrit language. The following Table 1 shows the analysis of sentences in Bhagwatgeeta. Approximately 05 sentences have been analyzed in the system.

Table 1. Analysis of Bhagwatgeeta Sentences

	Sentences	
Sr no.	**Input Sentence**	**Output**
1.	ज्यायसी चेत्कर्मणस्ते मता बुद्धिर्जनार्दन तत्किं कर्मणि घोरे मां नियोजयसि केशव	ज्यायसी + चेत् + कर्मणस् + ते + अमता + बुद्धिस् + जन + आर्दन् + अतत् + किम् + कर्मणि + घोरे + माम् + नि + योजयसि + केशव
2.	लोकेऽस्मिन् द्विविधा निष्ठा पुरा प्रोक्ता मयानघ ज्ञानयोगेन साङ्ख्यानां कर्मयोगेन योगिनाम्	लोके + अस्मिन् + द्विविधा + निष्ठा + पुरा + प्र + उक्ता + मयान् + अघ + ज्ञान + योगेन + साङ्ख्यानाम् + कर्म + योगेन + योगिनाम्
3.	ॐ तत्सदिति श्रीमद्भगवद्गीतासूपनिषत्सु ब्रह्मविद्यायां योगशास्त्रे श्रीकृष्णार्जुनसंवादे साङ्ख्ययोगो नाम द्वितीयोऽध्यायः	ओम् + तत् + सत् + इति + श्रीमत् + भगवत् + गीतासु + उपनिषत्सु + ब्रह्म + विद्यायाम् + योग + शास्त्रे + श्री + कृष्णा + अर्जुन + संवादे + साङ्ख्य + योगस् + अनाम + द्वितीयस् + अध्यायस्
4.	तस्मात्त्वमिन्द्रियाण्यादौ नियम्य भरतर्षभ पाप्मानं प्रजहि ह्येनं ज्ञानविज्ञाननाशनम्	तस्मात् + त्वम् + इन्द्रियाणि + आदौ + नि + यम्य + भरत + ऋषभ + पा + अप् + मानम् + प्र + जहिहि + एनम् + ज्ञान + विज्ञान + नाशनम्
5.	इमं विवस्वते योगं प्रोक्तवानहमव्ययम् विवस्वान्मनवे प्राह मनुरिक्ष्वाकवेऽब्रवीत्	इमम् + विवस्वते + योगम् + प्र + उक्तवान् + अहम् + अव्ययम् + विवस्वान् + मनवे + प्र + आह + मनुस् + इक्ष्वाकवे + अब्रवीत्

From Bhagwatgeeta 500 words have been analyzed showing 10 words in the following Table 2.

Table 2. Analysis of Bhagwatgeeta Words

	Bhagwatgeeta Words	
Sr no.	**Input word**	**Output**
01	प्रथमोऽध्यायः	प्रथमस् + अध्यायस्
02	पाण्डवाश्चैव	पाण्डवास् + च + एव
03	दुर्योधनस्तदा	दुर्योधनस् + तदा
04	आचार्यमुपसंगम्य	आचार्यम् + उप + सम् + गम्य
05	पाण्डुपुत्राणामाचार्य	पाण्डु + पुत्राणाम् + आचार्य
06	भीमार्जुनसमा	भीम + अर्जुन + समा
07	पुरुजित्कुन्तिभोजश्च	पुरु + जित् + कुन्ति + भोजस् + च

08	तान्ब्रवीमि	तान् + ब्रवीमि
09	भीष्माभिरक्षितम्	भीष्म + अभि + रक्षितम्
10	भीष्ममेवाभिरक्षन्तु	भीष्मम् + एव + अभि + रक्षन्तु

The system analyzes the Bhagwatgeeta data (i.e. words, sentences, and paragraphs) which show that rules in the Sanskrit language are implemented. As the objective is defined, the result shows that the rules are implemented in the system and the analysis is done using Bhagwatgeeta data. The analysis of rules of sandhi viccheda in Sanskrit has been shown in Table 4.3. Each and every rule of sandhi viccheda has been analyzed. Limitation for the system is that some rules are not implemented due to some symbols in the Sanskrit language which system does not support.

Table 3. Analysis of Rules of Sandhi Viccheda

Sr no.	**Sandhi**	**Type**	**No. of rules passed**	**No. of rules failed**
1.	Dirgha Sandhi Rule	Vowel	04	00
2.	Guna Sandhi Rule	Vowel	04	00
3.	Yana Sandhi Rule	Vowel	03	01
4.	Vrddhi Sandhi Rule	Vowel	04	00
5.	Ayadi Sandhi Rule	Vowel	06	00
6.	Soft to hard consonant	Consonant	00	01
7.	Hard to soft consonant	Consonant	01	00
8.	ह् to 4th of class	Consonant	01	00
9.	Dental to palatal	Consonant	04	00
10.	Dental to celebral	Consonant	03	01
11.	Dental to ल्	Consonant	01	01
12.	Consonant to nasal of class	Consonant	02	00
13.	श् to छ्	Consonant	01	00
14.	म् and न् to anusvara	Consonant	05	01
15.	Consonants ङ्, ण्, न्	Consonant	01	00
16.	च् before छ् and न् **to** ण्	Consonant	02	01
17.	Visarga to 'ओ'	Visarga	02	00
18.	Visarga is dropped	Visarga	05	01
19.	Visarga to र्	Visarga	01	02
20.	Visarga to श्, ष्, स्	Visarga	05	01
21.	Visarga to ardha-visarga	Visarga	02	00
Total			57	10

V. Conclusion and Future Work

Sandhi viccheda is a technique in which a long word, sentences, and paragraphs are separated into its constituent words. It is a challenging task to implement in the Sanskrit language itself. Till the vowel sandhi has implemented but adding the other rules the system increases the accuracy. The difficult task is to provide output in the Sanskrit language itself. The result is obtained when the Sanskrit word is given to the system as an input. Though there are many works has been done but the dataset used is different that is Bhagwatgeeta. In future, the text summarization is done in Sanskrit language using different methods so that the summary of that paragraph is understood by the user.

References

[1] Rupali Deshmukh and Varunakshi Bhojane, "*Building Vowel Sandhi Viccheda System for Sanskrit*", International Journal of Innovations and Advancement in Computer Science, Vol. 4, December 2015.

[2] Mrs. Namrata Tapaswi, Dr. Suresh Jain and Mrs. Vaishali Chourey, "*Parsing Sanskrit Sentences Using Lexical Functional Grammar*", Proceedings of the International Conference on Systems and Informatics, IEEE, pp 2636-2640, 2012.

[3] Priyanka Gupta, Vishal Goyal, "*Implementation of Rule-Based Algorithm for Sandhi-Viccheda of Compound Hindi Words*", IJCSI International Journal of Computer Science Issues, Vol. 3, 2009.

[4] Joshi Shripad S., "*Sandhi Splitting of Marathi Compound Words*", International Journal on Advanced Computer Theory and Engineering (IJACTE), Vol. 2, no. 02, 2012.

[5] Latha R. Nair, S. David Peter, "*Development of a Rule-Based Learning System for Splitting Compound Words in Malayalam Language*", Proceedings of the Recent Advances in Intelligent Computational Systems, IEEE, pp 751-755, 2011.

[6] Devadath V V, Litton J Kurisinkel, Dipti Misra Sharma, and Vasudeva Varma, "*A Sandhi Splitter for Malayalam*", Proceedings of the 11th International Conference on Natural Language Processing, 2014.

[7] M. Rajani Shree, Sowmya Lakshmi, "*A novel approach to Sandhi splitting at Character level for Kannada Language*", Proceedings of the 2016 International Conference on Computational Systems and Information Systems for Sustainable Solutions, IEEE, pp 17-20, 2016.

[8] Sachin Kumar, "*Sandhi Splitter and Analyzer for Sanskrit*", With Special Reference to aC Sandhi, Special Centre for Sanskrit Studies, Jawaharlal Nehru University, New Delhi, 2007.

[9] Shubham Bhardwaj, Neelamadhav Gantayat, Nikhil Chaturvedi, Rahul Garg, Sumeet Agarwal, "*SandhiKosh: A Benchmark Corpus for Evaluating Sanskrit Sandhi Tools*", Language Resources and Evaluation Conference, 2018.

[10] Amba Kulkarni, Sheetal Pokar and Devanand Shukl, "*Designing a Constraint-Based Parser for Sanskrit*", Proceedings of the International Sanskrit Computational Linguistics Symposium, SpringerLink, vol. 6465, pp 70-90, 2010.

[11] Vaishali Gupta, Nisheeth Joshi, Iti Mathur, "*Rule-Based Stemmer in Urdu*", Proceedings of the 2013 4th International Conference on Computer and Communication Technology, IEEE, pp 129-132, 2013.

[12] Anil Kumar, V.Sheebasudheer, Amba Kulkarni, "*Sanskrit Compound Paraphrase Generator*", Proceedings of the ICON, 2009.

[13] Pawan Goyal, Vipul Arora, and Laxmidhar Behera, "*Analysis of Sanskrit Text: parsing and Semantic Nets*", Springerlink, Proceedings of the Sanskrit Computational Linguistics, Vol. 5402, pp 200-218, 2009.

[14] Bhagyashree D. Patil and Manoj E. Patil, "*A Review on Implementation of Sandhi Viccheda for Sanskrit Words*", Proceedings of the International Conference in ICGTETM, IJCRT, vol.5, no. 12, pp 489-493, December 2017.

Authors Profile

Miss Bhagyashree D. Patil pursuing Masters in Computer Science and Engineering from Shrama Sadhana Bombay Trust, College of Engineering Bambhori, Jalgaon in the year 2018.

Mr. Manoj E. Patil pursed Ph.D. and currently working as Assistant Professor in Department of Computer science and engineering, North Maharashtra University, Jalgaon. He is a member of ISOC, IAENG. He has published more than 15 research papers in reputed international journals. He has 13 years of teaching experience.