ISSN: 2633-4828 Vol. 5 No. S1, (Jan - Feb 2023)

International Journal of Applied Engineering & Technology

EXPLORING THE INTRICACIES OF SANDHI IN SANSKRIT: PHONOLOGICAL RULES AND LINGUISTIC SIGNIFICANCE

Madhura Phadke Shreya Patankar
Assistant Professor, Department of Computer Enginering, K J Somaiya Institute of Information Technology, Sion, Mumbai

ABSTRACT

The combination of small words (morphemes) through a morph phonological process known as Sandhi creates compound words in Sanskrit. Sandhi splitting involves breaking down a compound word into its constituent morphemes. Despite the existence of rules governing this process, pinpointing the exact location of splits is exceedingly difficult. This challenge arises because a single compound word can be split in multiple syntactically correct ways. Sandhi is an important idea essential to morphological analysis of Sanskrit texts. Sandhi leads to word transformations at word boundaries. The rules of Sandhi formation are well defined but complex, sometimes optional and in some cases, require knowledge about the nature of the words being compounded. Sandhi split or Vichchhed is an even more difficult task given its non-uniqueness and context dependence. Current systems attempt to incorporate predefined splitting rules but often struggle with accuracy due to their inability to accurately identify the precise split locations.

This study aims to examine phonological rules and linguistic significance for handling sandhi in Sanskrit.

Key words: Language, Sandhi, Sanskrit, Morphemes, Compound Words.

1. INTRODUCTION

Sanskrit stands as one of the oldest among the Indo-Aryan languages, with its origins dating back to around 1500 BCE. It holds the distinction of being among the world's oldest surviving languages. A vast body of religious, philosophical, socio-political, and scientific texts from the diverse cultures of the Indian Subcontinent are composed in Sanskrit. In its various forms and regional dialects, Sanskrit served as the lingua franca of ancient India, facilitating communication across different regions and communities. [1]. Therefore, Sanskrit texts serve as invaluable resources for understanding ancient India and its people. The earliest known Sanskrit documents are written in a form known as Vedic Sanskrit. The Rigveda, the oldest among the four Vedas that constitute the principal religious texts of ancient India, is composed in Vedic Sanskrit. Around the 5th century BCE, a renowned Sanskrit scholar named Panini [2] wrote a treatise on Sanskrit grammar , in which Panini formalized rules on linguistics, syntax and grammar for Sanskrit. It is the oldest surviving text and the most comprehensive source of grammar on Sanskrit today. This literally means eight chapters and these eight chapters contain around 4000 sutras or rules in total. These rules completely define the Sanskrit language as it is known today. It is remarkable in its conciseness and contains highly systematic approach to grammar. Because of its well defined syntax and extensively well codified rules, many researchers have made attempts to codify the Panini's sutras as computer programs to analyze Sanskrit texts.[3][4][5]

1.1 Introduction of Sandhi and Sandhi Split in Sanskrit

Sandhi refers to a phonetic transformation at word boundaries, where two words are combined to form a new word. Sandhi literally means 'placing together' (samdhi-sam together + daDAti to place) is the principle of sounds coming together naturally according to certain rules codified by the grammarian Panini[5]

There are 3 different types of Sandhi:

(1) Swara (Vowel) Sandhi: Sandhi between 2 words where both the last character of first word and first character of second word, are vowels.

(2) Vyanjana (Consonant) Sandhi: Sandhi between 2 words where at least one among the last character of first word and first character of second word, is a consonant.

(3) Visarga Sandhi: Sandhi between 2 words where the last character of first word is a Visarga(a character in Devanagri script that when placed at end of a word, indicate an additional H sound

An example for each type Sandhi is shown below:

vidyA + AlayaH = vidyAlayaH (Vowel Sandhi)

vAk + hari = vAgGari (Consonant Sandhi)

punaH + api = punarapi (Visarga Sandhi)

Sandhi Split on the other hand, resolves Sanskrit compounds and "phonetically merged" (sandhified) words into its constituent morphemes. Sandhi Split comes with additional challenge of not only splitting of compound word correctly, but also predicting where to split.[6] Since Sanskrit compound word can be split in multiple ways based on multiple split locations possible, split words may be syntactically correct but semantically may not be meaningful. tadupAsanIyam = tat + upAsanIyam tadupAsanIyam = tat + up + AsanIyam

1.2 Computational Aspect of Sandhi

A The computational aspect of *sandhi* has two dimensions: forward computation and reverse computation [7][8]. These two can be described as follows:

1.2.1 Forward Computation of Sandhi

Forward computation of *sandhi* means computing Pāninian rules for *sandhi* formation leading to resultant sounds from a *samhita* situation. The rules, according to which either one or more sounds are modified in such cases, are called the rules of *sandhi*. *Sandhi* can be within a word or between two or more words. To do this computationally, a computational representation of *sandhi* rules and an algorithm to generate *sandhi* is needed. For example; to combine the word

rāmasya+ācāryah;

the four rules of simple vowel combination are required:

(1) /ā/+/ā/→/ā/,

(2) /ā/+/a/→/ā/,

(3) /a/+/ā/→/ā/,

(4) /a/+/a/→/ā/

In other words, this can be given as [simple vowel] → [+long simple vowel] / - [vowel of same type] (*akah savarne dirghah* in Pāninian terms)

1.2.2 Reverse Computation of Sandhi

In the reverse *sandhi* case, the above procedure is reversed. Reverse computation of *sandhi* means applying Pāninian rules in reverse form to split the *sandhi* derived words into their constituent morphemes. This parsed or simplified Sanskrit text will be useful in various NLP applications for Sanskrit. This process needs computational representation of *sandhi* rules in reverse format, an algorithm to parse Sanskrit words and linguistic resources to validate the split words. For example, to split the word *rāmasyācāryah* into *rāmasya+ācāryah*; the four rules of simple vowel combination are required in reverse format.

1.3 Computational Morpho-Phonemics

1.3.1 Computational Phonology

Phonology is a subfield of linguistics which studies the sound system of a language. It deals with the analysis, classification and organization of the phonemes of a language. It differs from phonetics in the sense that phonetics is the study of the production, transmission, and perception of speech sounds whereas phonology studies how

they are combined, organized, and convey meaning in a particular language. An important part of phonology is to study which sounds are distinctive units within a language.[9] In Sanskrit, for example, /k/ and /n/ are distinctive sounds (i.e., they are phonemes). This can be seen from minimal pairs such as '*karah*' and '*narah*', which mean different things, but differ only in one sound in identical position (word initial position in this case). Similarly, /t/ and /r/ respectively in *kukkutah / kukkurah* (word non-final position) and /v/ and /m in gacchavah / gacchamah (word non-final position) Computational Phonology is the field which deals with the computational techniques of the representation and processing of phonological rules and behaviour.[10] This can be useful in NLP applications such as speech recognition, text-to-speech etc. Computational phonology can be generative as well as analytical. For example, formulation of a phonological rule for voicing alternation can be as follows:

+ cons -> [+ voice] /- [+ voice]

(a consonant becomes voiced if a voiced sound follows)

The same rule can be analyzed in the following way:

[+voiced cons] □ [-voiced cons] /- [+voiced sound]

1.3.1.1 Issues in Computational Phonology

There are various issues related to the representation, procedures and implementation of Computational Phonology.[11][12] These issues can be described as follows:

1. **Representations:**

a. What are the representation formalisms for phonological knowledge, computational and cognitive reasoning, data structures for phonemes, strings of phonemes, syllable structures, feature matrices and Procedures.

b. What procedures are required for mapping one phonological representation to another and implementing phonological rules

2. **Implementations:**

How does one set about designing and making an implementation?

1.3.2 Computational Morphology

Morphology is a branch of linguistics which deals with the formation of word. It studies the patterns and rules of grouping sounds into words, their grammatical paradigms and grammatical properties. The basic building blocks of words are morphemes.[13] A morpheme is the smallest meaningful linguistic unit. Morphemes are of two types: free morphemes and bound morphemes. Free morphemes are those which can occur as a word by themselves, for example *rama*. Bound morphemes are the morphemes which occur only in combination with other forms.[14] All affixes are bound morphemes, for example */su>h/* in *ramah.*

Computational morphology is analysis and generation of word-forms through computational techniques[15]. This morphological information is very useful in analyzing a language because syntactic analysis requires morphological analysis. This morphological information can be used in various NL applications such as parsing, lemmatization, text to-speech, Machine Translation (MT), spell checker, spell corrector, automatic word separator, text generation and word paradigm builder.

Morphological analysis is a complex task. It has various dimensions which can be described as follows–

1.3.2.1 **Complexity of Word Formation**

Words are built up by joining morphemes according to the permissible patterns in a language [16]. Typologically, languages are of Agglutinative, Isolating, Inflectional and Polysynthetic types based on how morphemes combine to form words productively

1. **Morphological processes:**

There are essentially three types of morphological processes which determine the function of morphemes. These three processes are inflectional, derivational and compounding.

2. **Morpheme combination**

Morphemes can be combined in a variety of ways to build the words such as concatenation, infixation, circumfixation, templatic combination and reduplication.

1.3.3 Issues in Computational Morphology

a. What kind of data needs to be compiled?

b. What are the morphological rules and how to represent them for computational purposes?

c. What are possible implementation strategies?

d. What are potential ambiguities and how to resolve them?

1.3.4 Morphophonemics or Morphophonology

Words are composed by concatenating morphemes. Morphotactics governs the rules for this combination of the morphemes. Sometimes in this concatenation process, there occur some phonological changes at morpheme boundary [17] [18]. These modifications and their underlying reasons are studied under morphophonemics or morphophonology.5 For example, assimilation in Sanskrit where two segments influence each other at word boundary i.e. *tat+ca=tacca*, *tat+tikā=tattikā.* Here /t/ (dental) changes to /c/ (palatal) and */t/ (retroflex)* respectively.

1.3.4.1 Issues in Morphophonemics

a. What are the morphophonemic rules which explain these changes

b. How to represent theoretical rules for computational purposes

c. How to restrict the generation of ungrammatical words

d. How to handle ambiguities

1.3.5 Morphophonemics in Sanskrit

Typologically, Sanskrit belongs to the inflectional category. Words get their forms when bound morphemes combine with the bases and get fused with them. For example, *rama +ta (ina)* □ *ramena. Sandhi* governs these morphophonemic changes at morpheme or word boundary in terms of alteration to the sounds due to the neighboring sounds or due to the morphological behaviour of adjacent words. *Sandhi* can take place between vowel and vowel, vowel and semivowel, semivowel and semivowel, consonants and consonants and between visarga and other sounds. *Sandhi* is useful in internal structuring of constituents like verbs, and padas (internal sandhi), as well as for the combination of two words (external sandhi).6 This sandhi is compulsory within the internal structure of a word, in concatenation of *dhātu* (root) and *upasarga* (prefix), and in *samāsa* (compounds), but in a sentence i.e. in the case of the finals and initials of the different words in a sentence, It depends on the will of the writer [19]

To analyze this, semantic consideration is also required because *sandhi* overlaps with *samāsa*. *Samāsa*, or compounding in Sanskrit, may consist of two or more words. In *samāsa*, only the last word takes case marker and the remaining words are used as *prātipadika* (crude form). In joining these words as well, the *sandhi* rules apply. The final consonant or vowel of preceding word, according to the *sandhi* rules, combines with the initial letters of the following words. For this purpose, a separate *samāsa* analyzer is needed, but *sandhi* analyzer will also be partially helpful by segmenting *samsata pada* (compound word) into *sandhi*-free constituents.

1.4 Need for the Sandhi Analyzer

Sandhi analyzer will be a very important component in any NL system that attempts to analyze and understand Sanskrit for computational purposes. In the architecture of a computational Sanskrit platform, various linguistic

resources such as lexicon, POS Tagger, *kāraka* analyzer, *subanta* analyzer, *tinanta* analyzer, *linga* analyzer, *sandhi* analyzer, *samāsa* analyzer etc. will be needed. All these resources will be interlinked but *sandhi* analyzer will be a pre-requisite for analyzing a Sanskrit text because words in Sanskrit language are generally written with no explicit boundaries [20].

This *sandhi* analyzer module will be useful in many ways. Sanskrit has a vast knowledge reserve of diverse disciplines. To make this knowledge available to the users of other languages, an automatic MTS from Sanskrit to other Indian languages will have to be developed. *Sandhi-viccheda* will be an essential initial step for this work. The other applications of this segmented form of Sanskrit text may be in building a search algorithm and spell checker for Sanskrit corpora. A '*sandhi-aware*' system thus will not only be essential for any larger Sanskrit NL system, but will also be helpful for selfreading and understanding of Sanskrit texts by those readers who do not know or want to go through the rigors of *sandhi viccheda*. It will also be helpful for interpretation and simplification of Sanskrit text. Any NL or NL like Sanskrit compiler will have *sandhi viccheda* as a necessary initial component.

2. SANDHI IN SANSKRIT

The word *sandhi* refers to a wide variety of phonological changes at morpheme or word boundary in which two letters combine and they have certain changes. Pānini, has not used the word '*sandhi*', instead he uses the word '*sanhitā*' which is defined as '*parah sannikarah samhitā'*. *Samhitā* means the close proximity of two letters either within a word or between two words which results into the natural phonetic combination of these letters.

2.1 Sandhi: Morphophonological or Morpholexical Alternation

There are two main types of alternation: morphophonological and morpholexica [21]. In the first, the variation is determined by the phonetic environment while in the second type, the alternation depends not upon any phonetic environment, but upon the selection of neighbouring morphemes without regard to their phonetic form. He explains these two types of alternation from Sanskrit by illustrating the different past participal forms *matta* and *panna*. He says that it is simply the selection of the roots *mad*and *pad*- respectively that determines whether the suffix shall be /ta/ or /na/, i.e. the /ta/ or /na/ alternation is morholexical. On the other hand, the fact that the root takes the form *mat*-in *matta* and *pan*-in *panna*, is a matter of morphophonological alternation, being determined by the nature of the suffixial initial, viz. /t/ in the one case and /n/ in the other. One more example from the infixation process in the verb also fits here. In the verb forms *bhavati* and *karoti,* it is the selection of the root *bh*□ *bhvādigana*) and *kr*(*tanādigana*) that determines respectively whether the infix will be /a/ (sap) or /u/. This is morpholexical lternation. On the other hand, the process in which the /u/ of *bhu* changes to /o/→/av/ and /r/ of *kr* changes to /ar/ in *karoti*, is a matter of morphophonological alternation because it is being determined by internal process of euphonic combination.

Sandhi is intricately linked with morphophonological alternation, which refers to changes in sound structure depending on phonetic context. Its primary purpose is to facilitate smoother pronunciation in speech. For instance, in Sanskrit, when a word ends with a voiceless stop consonant, and the following word begins with a voiceless consonant, or when a word ends with a voiced consonant followed by a voiced initial in the next word, these sandhi rules ensure that transitions between sounds are more fluid and less complex.

By adhering to these rules, speakers can pronounce words more naturally and effortlessly, avoiding the potential awkwardness or difficulty that might arise if the phonetic sequence were less harmonized. This aspect of sandhi not only enhances the oral clarity and flow of Sanskrit speech but also contributes to its overall phonetic elegance and beauty as a classical language.

2.2 External and Internal Sandhi

Sandhi, a pivotal aspect of Sanskrit grammar, operates through two distinct processes: internal and external. The internal process focuses on how suffixes combine with roots or stems during declension, conjugation, and derivation. This aspect is crucial for forming grammatically correct words within the language's intricate structure.

In contrast, the external process of sandhi governs the rules that dictate changes in the final and initial letters of words within a sentence, as well as in compound words (samāsa). Understanding these rules is essential for comprehending Sanskrit sentences accurately. They play a vital role in ensuring the linguistic coherence and semantic clarity of texts.

Therefore, mastery of sandhi rules holds practical significance for anyone studying Sanskrit. It not only facilitates correct grammatical formation but also enhances the comprehension and interpretation of Sanskrit literature and texts, thereby enriching the overall understanding of the language's nuances and complexities.

Generally the rules of internal *sandhi* agree with the rules of external *sandhi*, but on some occasions they have exceptions too. For example, final /i/or /i'/, /u/ or /u'/, /r/ or /r'/ and /l/,

if followed by vowel or dipthongs, are generally changed to /y/,/v/,/ar/,/al/ respectively, but in declension or conjugation, /i/or /i'/, /u/ or /u'/, /r/ or /r'/ are changed to /iy/, /uv/, /ir/ respectively. For example: *bhu+i= bhuvi*, *gr'+ati=girati*.

The main characteristics of Sanskrit euphonic rules55 and their classification are as follows:

a. Assimilation
Assimilation refers to the influence exercised by one sound segment upon the articulation of another, so that the sound becomes more alike or identical. For example: palatization and retroflexation. If dental sounds are followed by palatal or retroflex sounds, the dental sound is changed to palatal or retroflex .respectively.

b. Voicing
If an unvoiced sound at word boundary is followed by a voiced sound, then the unvoiced sound is changed to the voiced sound.

c. Devoicing
If a voiced sound at the end of a word is followed by an initial unvoiced sound of the next word, voiced sound is changed to unvoiced sound.

d. Nasalization
If the unvoiced sound is followed by the nasal sound, the unvoiced sound is changed to the nasal sound of its own group.

e. Deaspiration
If the aspirated sound in the end of a word is followed by the unaspirated sound of a next word, the aspirated sound is changed and this aspiration is transformed.

f. Visarga
In Sanskrit, *visarga* in the end of a word, followed by different sound, is changed to different forms.

g. Doubling
An aspirated sound between vowel sounds is doubled and this doubled sound is unaspirated sound of that sound class.

h. Vowel Lengthening
When the two similar simple vowels (a, i, u, r, r') come together, their long form replaces both of them.

3. DESIGNING A MODEL FOR SANSKRIT SANDHI SPLIT

Learning the process of sandhi splitting for Sanskrit could provide linguistic insights into the formation of words in a wide-variety of Dravidian languages. From an NLP perspective, automated learning of word formations in Sanskrit could provide a framework for learning word organization in other Indian languages as well. . Thus, knowing all the rules of splitting is insufficient and it is essential to identify the location(s) of split(s) in a given compound word. Possible choices of architecture for the same are sequence to sequence model, RNN, LSTM [22][23].

3.1 Issues with standard architectures

Consider an example of splitting a sequence pqrstuvw as pqrst + uvw. The primary task is to identify s as the split location. Further, for a given location s in the character sequence, the algorithm should take into account (i) the context of character sequence abc,

(ii) The immediate previous character r,

(iii) The immediate succeeding character u, to make an effective split.

For such sequence learning problems, RNNs have become the the most promising models.

3.2 Comparison with Publicly Available Datasets and Tools

The Datasets studied are : The UoH corpus, created at the University of Hyderabad contains 113, 913 words and their splits. This dataset is noisy with typing errors and incorrect splits. The recent SandhiKosh corpus (Shubham Bhardwaj, 2018) is a set of 13, 930 annotated splits. We combine these datasets and heuristically prune them to finally get 71, 747 words and their splits. The pruning is done by considering a data point to be valid only if the compound word and it's splits are present in a standard Sanskrit dictionary (Monier-Williams, 1970). We use this as our benchmark dataset and run all our experiments on it. Tools: There exist multiple Sandhi splitters in the open domain such as (i) JNU splitter (Sachin, 2007), (ii) UoH splitter (Kumar et al., 2010) and (iii) INRIA sanskrit reader companion (Huet, 2003) (Goyal and Huet, 2013). Though each tool addresses the splitting problem in a specialized way, the general principle remains constant. For a given compound word, the set of all rules are applied to every character in the word and a large potential candidate list of word splits is obtained. Then, a morpheme dictionary of Sanskrit words is used with other heuristics to remove infeasible word split combinations. However, none of the approaches address the fundamental problem of identifying the location of the split before applying the rules, which will significantly reduce the number of rules that can be applied, hence resulting in more accurate splits

4. CONCLUSION

This study presented here lays a crucial foundation for advancing Sanskrit-based NLP tasks, with translation serving as a prime example. Sanskrit's unique feature of forming compound words by combining multiple simpler words poses a significant challenge for translation tasks. Ancient texts, especially from the Vedic era, abound with such complex compounds, exponentially expanding the vocabulary and complicating translation efforts.

However, by integrating a pre-processing step that systematically breaks down these compound words before training translation models, we can streamline the learning process. This study effectively reduces the number of unique words in the vocabulary, thereby enhancing the model's ability to grasp and interpret the language more efficiently.

REFERENCES

1. Agrawal, Muktanand, 2006, '*Computational Identification and Analysis of Sanskrit Verb-forms*', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, pp.126-127.
2. V Sheeba Akshar Bharati, Amba Kulkarni. 2006. Building a Wide Coverage Sanskrit Morphological Analyzer : A Practical Approach. In TheFirst National Symposium on Modelling and Shal-low Parsing of Indian Languages, IIT-Bombay.
3. Kumar, Sachin & Girish Nath Jha. 2006, 'Issues in sandhi processing of Sanskrit', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p.129.
4. Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran, and Senthil Mani. 2018. Sanskrit Sandhi Splitting using seq2(seq)2. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium, 4909–4914. https://doi.org/10.18653/v1/D18-1530

5. Mishra, Sudhir Kumar & Girish Nath Jha. 2005, 'Identifying Verb Inflections in Sanskrit Morphology', In the Proceedings of SIMPLE-05, IIT-Kharagpur, pp.79-81.
6. George Cardona. 1997. Panini: A Survey of Research. Motilal Banarsidass.
7. Vyas, Pankaj K & Sivaja S Nair. 2006, 'Computer Implementation of Sanskrit Sandhirules (Version 0.1)', In the Souvenir Abstracts of 28th AICL, BHU, Varanasi, p.128.
8. Abhiram Natarajan and Eugene Charniak. 2011. S3 - Statistical Sandhi Splitting. In Proceedings of 5th International Joint Conference on Natural Language Processing. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, 301–308. https://www.aclweb.org/anthology/I11-1034
9. Jha, D., Jha, R., & Varshney, V. (2014). Natural Language Processing and Sanskrit. International Journal of Computer Engineering & Technology, 5(10), 57-63.
10. Adiga, D., Kumar, R., Krishna, A., Jyothi, P., Ramakrishnan, G., & Goyal, P. (2021). Automatic speech recognition in Sanskrit: a new speech corpus and modelling insights. arXiv preprint arXiv:2106.05852.
11. Mishra, D., Jha, G. N., & Bali, K. (2011, March). Challenges in Developing a TTS for Sanskrit. In International Conference on Information Systems for Indian Languages (pp. 228-231). Springer, Berlin, Heidelberg.
12. Chaudhari, P. R., Gangurde, P. C., & Kulkarni, N. L. (2015). Study of methodologies for utilizing Sanskrit in computational linguistics. International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE).
13. Briggs, R. (1985). Knowledge representation in Sanskrit and artificial intelligence. AI magazine, 6(1), 32-32.
14. Jha, R., Jha, A., Jha, D., & Jha, S. (2016, March). Is Sanskrit the most suitable language for natural language processing?. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 211-216). IEEE.
15. Mishra, V. (2015). Sanskrit as a Programming Language: Possibilities & Difficulties. International Journal of Innovative Science, Engineering & Technology, 2(4).
16. Nair, R. R., & Devi, L. S. (2011). Sanskrit Informatics: Informatics for Sanskrit studies and research. Centre for Informatics Research and Development. Book
17. Raulji, J. K., & Saini, J. R. (2016). Stop-word removal algorithm and its implementation for Sanskrit language. International Journal of Computer Applications, 150(2), 15-17.
18. Sanyala, K., & Paula, P. (2021). A Study of aindra School of Sanskrit Grammar in the Light of Paninian Framework in Natural Language Processing. Multicultural Education, 7(11).
19. Chakraborty, S. (2021). The Role of Specific Grammar for Interpretation in Sanskrit. Journal of Research in Humanities and Social Science, 9 (2), 107-187.
20. Deshpande, S., & Kulkarni, M. N. (2020). A Review on various approaches in Machine Translation for Sanskrit. International Journal of Future Generation Communication and Networking, 13(2), 113-120.
21. Gilda, K. S., Dixit, S. S., & Narote, S. V. (2019). General Structure of Sanskrit Machine Translation System.

22. Punia, R., Sharma, A., Pruthi, S., & Jain, M. (2020, December). Improving Neural Machine Translation for Sanskrit-English. In Proceedings of the 17th International Conference on Natural Language Processing (ICON) (pp. 234-238).

23. Panchal, B., Parmar, A., Dholitar, S. (2019). An Approach of Splitting Upsarg and Pratyay of Sanskrit Word Using Paninian Framework of Sanskrit Grammar. In: Abraham, A., Dutta, P., Mandal, J., Bhattacharya, A., Dutta, S. (eds) Emerging Technologies in Data Mining and Information Security. Advances inIntelligent Systems and Computing, vol 814. Springer, Singapore.