Journal of Information Technology and Digital World (ISSN: 2582-418X)
www.irojournals.com/itdw/

Review on Sanskrit Sandhi Splitting using Deep Learning Techniques

Sreedeepa H S[1], Alok Nath M [2],Ajay K Mani[3],Arun Kumar C[4], Sumam Mary Idicula[5]

[1]Dept. of Computer Science, Cochin University of Science & Technology, India

[2,3]CSE, CE Adoor, APJAKTU, India

[4]ME, CE Adoor, APJAKTU, India

[5]DCS, CUSAT, Kerala, India

E-mail: [1]sreedeepa@cusat.ac.in [4]arunkumarc@cea.ac.in

Abstract

A procedure called sandhi is used in Sanskrit to join short words (morphemes) to create compound words. A composite word are broken down into their component morphemes by a process known as sandhi splitting. This study focuses on several performance technologies and methodologies used to perform the above operation on Sanskrit sentences. Various approaches were identified for the problem from the literature survey. Initial approaches involved use of Finite State Transducers. Earlier the approaches introduced to increase accuracy include use of mathematical models and various optimality theories. Graph based approaches and parser based techniques were introduced later. With the advancement of deep learning techniques Recurrent Neural Networks, Long-Short Term Memory models and Double decoder models were adopted which involved training machine learning models through neural networks and classifier algorithms. Bidirectional LSTM models with attention mechanism, transformer based models and large language models like BERT were the most recent methodologies adopted and proved to be of higher accuracy and performance.

Keywords: Deep learning, Natural Language processing, BiLSTM, Seq2seq models, RNN

DOI: https://doi.org/10.36548/jitdw.2024.2.003
Received: 03.04.2024. received in revised form: 27.04.2024, accepted: 11.05.2024, published: 21.05.2024

1. Introduction

Sanskrit was among the first of the Indo-Aryan languages. Small words (morphemes) in Sanskrit are merged to create compound words by a procedure called sandhi. The technique of separating a given compound word into its individual morphemes is known as sandhi splitting. The concept of sandhi is crucial to understanding how to analyse Sanskrit texts morphologically. Sandhi leads to word transformations at word boundaries. The requirements for sandhi formation are clearly laid out but difficult, occasionally discretionary, and occasionally requiring understanding of the nature of the words being compounded. Due to its context dependence and lack of uniqueness, sandhi splitting is a much more challenging task. It can be challenging to locate the splits in a word that is complex. NLP, contemporary deep learning methods, algorithms, and machine learning models are all used in the system.

In Sanskrit linguistics, sandhi refers to the phonological rules governing the combination and transformation of sounds when words are joined together. Sandhi splitting involves segmenting words back into their constituent parts to facilitate analysis or processing. Sanskrit input text must undergo an automated sandhi analysis as a prerequisite for thorough analysis since this process will simplify the text and allow for additional grammatical and Part of Speech (POS) analysis. The creation of a Sanskrit text search engine, the Sanskrit Indian Language Machine Translation System (MTS), the tagging of sizable text corpora, and the development of a Sanskrit spell checker are just a few NLP initiatives that could benefit from this research of Sanskrit sandhi splitter. This will be helpful for reading and comprehending Sanskrit texts on your own in addition to being an essential component of the NL Sanskrit system.

2. Literature Survey

A thorough analysis of the literature was done. Various methods for the implementation of sandhi splitting were studied. Different techniques of implementing various stages of the Sanskrit sandhi splitter were identified and compared for their accuracy and performance. The papers were studied and grouped on the basis of methodologies or technologies employed.

a. FST Model

The computational architecture for the analysis of classical Sanskrit was offered by Gerard Huet [1]. The platform's components for segmentation, shallow syntax analysis, morphology, and phonology are arranged around a structured lexical database. It takes use of the data structures and algorithms provided by the functional Zen toolkit for finite state automata and transducers, which enables the modular construction and execution of finite state machines. Internal sandhi is used to build morphemes, and the inflected forms are then kept in dictionaries with morphological tags that can be used for lemmatizing. Following that, these dictionaries are combined into transducers that employ the study of external sandhi, a phonological procedure that combines words through euphony. This offers a segmenter for tagging that examines a phrase that is provided as a stream of phonemes and generates a stream of lexical entries that are tagged and hyperlinked to the lexicon. The syntax analyzer is the next layer, and it is controlled by semantic nets constraints that specify dependencies between word forms. According to valency patterns based on the voice of the form (active, passive) and the governance of the root (transitive, etc.), finite verb forms necessitate semantic roles. In contrast, noun/adjective forms offer actors who could play those parts if the requirements for agreement are met. In order to simulate linguistic phenomena like coordination by abstractly interpreting actor streams, tool words are mapped to transducers that operate on tagged streams. For the purpose of conclusive ambiguity validation, the parser ranks the various interpretations (which pair actors with roles) according to penalties and provides the user with the minimum penalty analysis. The entire platform is set up as a Web service, which enables the piece-wise tagging of a Sanskrit text. However, it is yet unclear how one may use this method to prioritize different splits. Additionally, this system needs some more work before its sandhi splitter can be utilised as a stand-alone system to accept the plugging in of various morphological analyzers. The "Sanskrit Heritage Reader," an automatic analyzer for thorough syntactically accurate analysis of a Sanskrit sentence, is now available thanks to Huet's work. However, the system offers every segmentation that is syntactically legal, and it takes human aid to select the appropriate segmentations in order to build the semantics.

A method for automated Sanskrit segmentations has been put out by Mittal [2] that relies on the highest posteriori estimate obtained from all possible sandhi splits for a given string. Mittal defined a baseline system which assumes that only two parts can be separated out

of each Sanskrit word. As long as both constituents are legitimate morphs, a word is traversed from left to right and subdivided by using the first relevant rule. Two approaches were used here:

1. Augmenting FST with Sandhi Rules: In this method, an FST that incorporates Sandhi rules within the FST itself and traverses it to locate Sandhi splittings, using the OpenFST toolkit. A Roman transliteration scheme called WX transliteration is used, which is one-to- one phoneme level representation of Devanagari script.

Validation using optimality theory: This strategy is based on optimality theory (OT), which contends that interactions between opposing constraints lead to the observed forms of a language. The theory's three fundamental elements are as follows: 1.GEN produces all potential results or candidates.

2. CON - outlines the standards and restrictions that will be applied to choose between candidates.

3. EVAL - picks the best candidate based on how the constraints conflict.

These components are considered universal by OT, and the grammars differ in how they rank the CON universal constraint set. Every constraint must have outperformed every lower ranked constraint according to the dominance order that each language's grammar assigns to the constraints. So, even though candidate A violates a lower-ranking criterion more frequently than candidate B, A is still considered to be optimal if it outperforms candidate B on the higher-ranking constraint. The GEN function generates every segmentation conceivable by applying the criteria when necessary. The input surface form is tokenized by the rules into its component parts. There may be some unnecessary words in there, but they will eventually be removed by the morphological analyzer in the EVAL function, leaving the best candidate. The winning candidate must satisfy all the constraints. The restrictions in effect are:

C1: Each component of a split has to be a valid morph. C2: Pick the split that has the most weight.

b. Mathematical Model

The 'S3 - Statistical Sandhi Splitter', a Bayesian word segmentation method that can handle sandhi forms, was proposed by Natarajan and Charniak [3]. Used posterior probability function to achieve better results. Found most probable split using Shannon noisy channel model framework. Significantly increased the proposed accuracy of Mittal's model. Their approach worked in two folds:

1. An ethical revision of the posterior probability function to provide better outcomes.

2. Algorithm based on Bayesian word segmentation methods.

c. Parsers

Verbal understanding of each utterance involves knowledge of how words in that utterance are related to one another, as demonstrated by Amba Kulkarni et al., [4][5]. Such information is typically accessible in the form of grammatical connection cognition. The way a language codes these relations is explained by generative grammars. Thus, it is possible to ascertain what information these grammatical relations send by looking at them from point of generation as opposed to the point of analysis. To build a parser based on any grammar, it is necessary to fully understand the semantic content of the grammatical relations presented in a language string, the cues for extracting these links, and finally whether these connections are expressed explicitly or implicitly. Given a graph with nodes denoting the words and edges denoting potential relationships between them, the parser for discovering a directed Tree is created based on the design principles that result from this information. Additionally, non-solutions were eliminated using expectation constraints, and the solutions were ordered using proximity constraints.

The parser determines the root words in a Sanskrit text and provides dependency relations based on semantic constraints. For several types of Sanskrit paragraphs, the suggested Sanskrit parser can provide semantic networks. The parser handles both the exterior and internal sandhi in the Sanskrit words.

A system known as SAS (Sanskrit Analysis System), a comprehensive analysis system for Sanskrit, was proposed by Manji Bhadra et al [10]. A few of the system's modules have already been created. The system supports Devanagari Unicode (UTF-8) full text inputs. For

complicated tokens, the Sandhi module segments the text before handing it off for in-depth analysis. The shallow parser and the karaka analyzer are the two main parts of the SAS. The shallow parser consists of various modules, some of which have been fully implemented and others are in the process of being implemented. The modules were created utilising RDBMS approaches as java servlets that support Unicode. The SAS will have numerous applications, including machine translation from Sanskrit to other languages and as a reading assistant for Sanskrit.

d. Graph Model

Amrith Krishna et al.'s framework [6] for word segmentation, dependency parsing, and morphological parsing was based on a graph-based parsing technique. The process of detecting the morphemes in a sentence is known as morphological parsing. Finding the syntactic relationships between the words in a phrase and predicting a labelled dependency tree as the output is known as dependency parsing. Probability is calculated for segmentation.

e. Language Translations

The rule-based technique was used by Ved Kumar Gupta et al. [9] to propose a knowledge representation of the Sanskrit to English machine translation process. To create Lexemes for this, a parsing approach is utilized. Then it will be utilized as input throughout the translation process. For creating final results, the required output is created using some mapping rules and a dictionary-based patterns.

f. Neural Network Models

An English text (the source language sentence) is converted into an equivalent Sanskrit sentence (the target language phrase) by Vimal Mishra et al. [8] Artificial Neural Network

(ANN) model, which they combined with the conventional rule-based approach to machine translation. The feed forward ANN is used to choose Sanskrit words from English to Sanskrit User Data Vector (UDV), such as nouns, verbs, objects, and adjectives. The Sanskrit language has a rich morphology, therefore the system uses morphological markers rather than syntax to distinguish the subject, object, verb, preposition, adjective, adverb, and conjunctive phrases. In order to identify clauses, their Subject, Object, Verb, and other elements, as well as

the Gender-Number-Person (GNP) of nouns, adjectives, and objects, it requires limited parsing. This system represents the translation between the SVO and SOV classes of languages. The system handles English sentences of various kinds and provides translation results in GUI format.

In order to simultaneously handle the problems of compound splitting and sandhi resolution, Hellwig [11] presented a method based on neural networks. A Recurrent Neural Network is used for the labeling task. The method involved tokenizing Sanskrit by jointly splitting compounds and resolving phonetic merges. The model does not require feature engineering or external linguistic resources. It works well with just the parallel versions of raw and segmented text. The approach treated the problem as a character level sequence labeling task. The results of the approach were not at word level.

The sandhi splitter and analyzer for Sanskrit is presented by Sachin Kumar [12]. The system's analysis process employs both the rule base method and the lexical lookup method. Prior to the sandhi analysis approach, pre-processing, lexical searching of sandhi strings in the sandhi example base, and subanta-analysis are all carried out. The input's punctuation will be marked during pre-processing. The Sandhi sample base is then checked by the program. The terms of sandhi-exceptions and frequently occurring sandhi strings with their split forms can be found in this sample base. Without parsing each word for processing, these words are tested first to determine their split forms. The case terminations are split from the underlying word by the subanta analyzer following lexical search. Given that Sanskrit words in lexicon are stored in base form, subanta analysis will be useful in validating the split words produced using reverse sandhi analysis. Because sandhi-derived terms in the input Sanskrit text could have any of the case terminations, it is best to collect the words in base form. . After text input has been subanta-normalized, the system will search for a preset word list of location names and nouns. The words you find in these sources won't be processed.

Rahul Aralikatte et al.,[16] proposed a Seq2Seq model approach with the help of double decoder RNN (DD-RNN) models for improved performance and accuracy. The split

prediction performance was enhanced by adding global attention (abbreviated as B-RNN-A) to the decoder, which allowed the model to pay attention to the characters surrounding the probable split location(s). The double decoder included location decoder and character

decoder to facilitate most accurate split locations. Location decoder to generate binary vector of split indices. Character decoder generates split words. Encoder and attention weights are fine-tuned and decoder learns with the help of pre-trained attention models.

Sushant Dave et al.,[17] proposed a method for Sanskrit Sandhi generation and splitting using RNN and BiLSTM models. The problem was formulated as a sequence to sequence prediction task. Used RNN with BiLSTM for decoder. Basic RNN with LSTM for encoder. Model was trained using RMSProp optimizer. The model does not necessitate the employment of any outside resources, such as language models, morphological analyzers, or phonetic interpreters. The model was only meant for training and testing and wasn't deployed as a functional tool.

g. POS Tagging

A unique method to the internal Sandhi splitting technique on the Kannada language is proposed by M. Rajani Shree et al. [16]. According to legitimate morph patterns, each Kannada word is divided into morphemes. Text processing was done using POS tagging and parsing as a result. Each word has been manually categorized into its root-begins, root- continuous, and suffixes after being divided into its lexical morphemes. With a list of 1000 tagged words as input and about 400 raw split words (untagged words), the system uses a CRF (Conditional Random Fields) tool. The system creates a list of tagged split words for the input based on the learned data. Data that has been manually labelled has been compared to the output of the system.

A character tagging approach was used by Xue[14] to address the segmentation of Chinese words. Chinese word segmentation is approached as a character labelling job, in which each character of the input sequence is assigned one of the four labels L = {B, M, E, S,} which stand for character in the beginning, middle, or end of the word or single character word. The tags are applied separately to each character using a maximum entropy tagger. Linear conditional random fields were employed to try this strategy in the sequence modelling job, and the results were state-of-the-art.

h. Classifier Algorithms

A machine learning strategy for detecting compound words from Sanskrit was proposed by Premjith B, et al.,[19] using KNN classifier algorithm and fast Text embedding. The compound word identification was modelled as a binary classification problem. The sentence was tokenized to obtain words. When vectorizing Sanskrit words, the fast Text embedding method was used. KNN classification algorithm was used to train the model at 80:20.

Many other authors have proposed a noun-noun compound multiword expression identification for Bengali language. It involved candidate extraction using chunk information and various heuristic rules and using Random forest algorithm to classify the candidates.

3. Related Works

The comparison of traditional approaches and deep learning based approaches which can be employed for the segmentation/sandhi splitting are given in Table 1. From the comparison it's clearly seen that deep learning based approaches are faster and gives more accurate result. Earlier approaches like FSA, graph based approaches are failed in finding correct split locations as they are more concentrate on states and prediction based on probability [20-24].

Table 1. Comparison of Traditional and Deep learning Approaches for Sandhi Splitting

Methods	Description	Approach	Observations
FST	Finite State Transducer implemented using OpenFST Optimality theory for validation	Finite State Automata	Lack of mechanism for identifying split location Primitive model
Statistical Sandhi Splitting	Improved performance for FST using Shannon noisy framework Probability Function	Finite State Automata	Lack of mechanism for identifying split location

PCRW	A multidigraph with pos tag using path constrained random walks & probability function for segmentation	Graph Based Approach	More complex procedure & implementations no prediction on split location
RNN	Tokenize Sanskrit by jointly splitting compounds and resolving phonetic merges.	Deep Learning	Encoded only context of characters appeared before split location. Les s accuracy. Multiple Split locations
Seq2seq model	Double Decoder RNN model with Seq2Seq approach. Neural network Based approach	Deep Learning	Location decoder & Character decoder for identification & Split words
RNN with Bi LSTM with Attention Mechanism	Improved accuracy in finding split locations with the help of Bidirectional LSTMs for forward & reverse traversal on compound words	Deep Learning	BiLSTM improved accuracy of finding potential split locations and predict most accurate split position among multiple splits along with attention mechanism

4. Methodology and Deep Learning Approaches

Finite State Automata, Graph Based Approach and Rule based approaches are the traditional methods used for Sandhi splitting. Let's compare traditional methods and deep learning approaches for Sanskrit sandhi splitting, highlighting how deep learning can offer higher accuracy and performance:

1. Rule-based Traditional Methods

Traditional methods often rely on rule-based approaches informed by linguistic principles. Linguists manually encode rules based on phonological and morphological characteristics of Sanskrit. Rule-based methods can achieve reasonable accuracy when the linguistic rules are well-defined and cover a broad range of cases. However, they may struggle with exceptions or irregularities not covered by the rules. The performance of rule-based

methods depends on the comprehensiveness and accuracy of the rules. They may require frequent updates or adjustments to handle new cases or dialectical variations.

2. Statistical and Machine Learning Traditional Methods

These methods involve statistical models or machine learning algorithms trained on annotated data. Features such as character n-grams, linguistic context, and morphological properties are used to train models like Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs). Statistical and machine learning methods can achieve higher accuracy compared to rule-based approaches, especially when trained on large, diverse datasets. They can capture patterns and variations not easily encoded by hand-crafted rules. These methods may require significant feature engineering and manual annotation efforts. While they can achieve good performance, they may struggle with limited availability of annotated data, especially for low-resource languages like Sanskrit.

3. Deep Learning Methods

Deep learning methods, particularly sequence-to-sequence models like recurrent neural networks (RNNs) or transformer-based architectures like the Transformer, learn to map input sequences (words) to output sequences (segmented words) directly from data. Deep learning methods excel in capturing complex patterns and dependencies in data, including those present in Sanskrit sandhi. They can automatically learn hierarchical representations of the input, potentially capturing nuances that rule-based or statistical methods might miss. Deep learning models can offer state-of-the-art performance in Sanskrit sandhi splitting, especially when trained on large dataset. They require less manual feature engineering compared to traditional methods and can adapt to different dialects or variations more effectively.

4.1 Advantages of Deep Learning

End-to-End Learning: Deep learning models learn directly from data without the need for hand-crafted rules or extensive feature engineering.

Representation Learning: Deep learning models automatically learn meaningful representations of the input data, capturing complex patterns and dependencies.

Scalability: Deep learning models can scale effectively with larger datasets and computational resources, potentially improving performance with more data.

In summary, while traditional methods for Sanskrit sandhi splitting have their strengths, deep learning approaches offer higher accuracy and performance by leveraging end-to-end learning and representation learning capabilities. Deep learning methods can capture intricate patterns in Sanskrit sandhi more effectively and adapt to different linguistic variations, making them a promising avenue for improving Sanskrit natural language processing tasks.

This work presents the comparative study of various deep learning models like LSTM-based Sequence-to-Sequence Models, Transformer-based Models, BiLSTM-CR and BERT-based Models that were implemented over a dataset generated manually.

For the implementation of various approaches, created a comprehensive collection of compound words and sandhi split words. It contains almost 1.25 lakhs words and its splits, created from various traditional Sanskrit text books like Ashtangahridaya, Bhagavadgita, Ramayana etc.

Different architectures such as Transformer models, LSTM-based Seq2Seq models, BiLSTM-CRF Models, BERT based models and hybrid architectures were implemented and trained using the dataset created to find the one that best captures the complex dependencies in Sanskrit sandhi. Pre-trained language model BERT is used to initialize model parameters and fine-tune them on the sandhi splitting task. Adjusted the number of layers, hidden units, and attention mechanisms to balance model complexity and performance. Sanskrit words are usually represented using character-level embeddings or subword-level embeddings like Byte Pair Encoding (BPE) to capture the morphophonemic variations. Annotated datasets consisting of Sanskrit words with their sandhi-split counterparts are used for training. The model is trained to predict the sandhi boundary between two words, using a cross-entropy loss function. Parameters like learning rate, batch size, and number of layers, hidden units, and dropout rates are tuned through experimentation to optimize performance.

5. Comparison

The comparative study of the results given by various models given on the created data set is given in Table 2.

Table 2. Comparison of Deep Learning Techniques

Sl.No.	Deep learning techniques	Approaches	Accuracy
1	LSTM-based Sequence-to-Sequence Models	LSTM is used for split prediction	89.3%
2	Transformer-based Models	Used for predicting splits by capturing long-range dependencies in sequences efficiently.	93.4%
3	BiLSTM-CRF Models	BiLSTM is used for capturing sequential patterns in compound words and CRF helps to split position labelling.	91.2%
4	BERT-based Models	BERT-based model is fine-tuned for Sanskrit sandhi splitting.	96.3%
5	Hybrid Models	Included some preprocessing stages like anvay generation and BERT	98.96%

The results shows by increasing the size of the dataset and using hybrid model approaches it's possible to increase the accuracy. The correct splitting of compound word has a major role in machine translation of Sanskrit shlokas. So this sandhi splitter can be used in the preprocessing stage of Sanskrit shlokas machine translation process to increase the accuracy of text generation in target language.

The accuracies obtained using various models are given in the Figure 1.

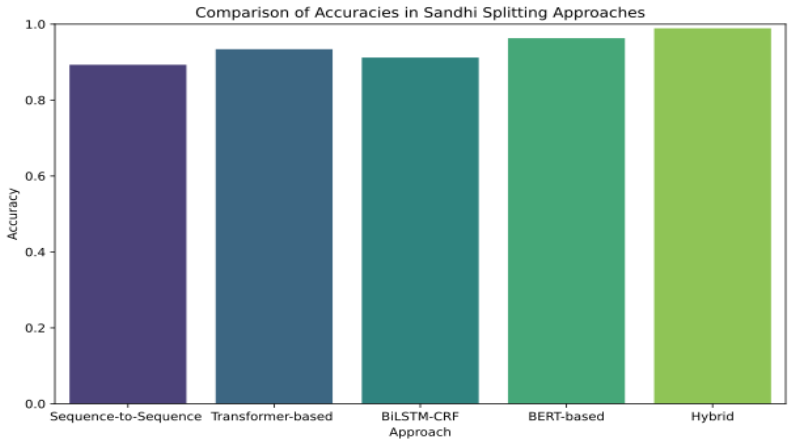

Figure 1. Comparison of Accuracies of Different Methods

6. Conclusion

Various approaches were identified for the problem from the literature survey. Initial approaches involved use of Finite State Transducers. Approaches introduced to increase accuracy include use of mathematical models and various optimality theories. Graph based approachers and parser based techniques were introduced later. With the advancement of deep learning techniques RNN, LSTM and Double decoder models were adopted which involved training ML models through neural networks and classifier algorithms. BiLSTM models with attention mechanism were the most recent methodologies adopted and proved to be of higher accuracy and performance. Among the above techniques BiLSTM models with attention mechanism gives more accuracy. By considering the contributions and limitations of each approach, future research can advance the state-of-the-art in Sanskrit natural language processing and contribute to broader efforts in preserving and analyzing linguistic heritage.The accuracy of the machine translation can be increased by introducing sandhi splitter module in preprocessing state of Sanskrit shlokas.

References

[1] Gérard, Huet. "Lexicon-directed segmentation and tagging of Sanskrit." In XIIth World Sanskrit Conference, Helsinki, Finland, Aug, pp. 307-325. 2003.

[2] Vipul Mittal. 2010. Automatic Sanskrit Segmentizer Using Finite State Transducers. In Proceedings of the ACL 2010 Student Research Workshop. Association for Computational Linguistics, Uppsala, Sweden, 85–90. https://www.aclweb.org/ anthology/P10-3015.

[3] Abhiram Natarajan and Eugene Charniak. 2011. S3 - Statistical Sandhi Splitting.InProceedings of 5th International Joint Conference on Natural LanguageProcessing. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, 301– 308.

[4] Amba Kulkarni and Devanand Shukl. 2009. Sanskrit Morphological Analyser: Some Issues. Indian Linguistics 70 (01 2009), 169–177.

[5] Amba Kulkarni and D. Shukl, “Designing a constraint based parser for Sanskrit," SpringerLink, Sanskrit Computational Linguistics pp 70-90, 2010.

[6] Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016. Word Segmentation in Sanskrit Using Path Constrained Random Walks,Proceedings of COLING2016, the 26th International Conference on Computational Linguistics: TechnicalPapers. The COLING 2016 Organizing Committee, Osaka, Japan, 494–504. https://www.aclweb.org/anthology/C16- 1048V.

[7] A. Pawan Goyal and L. Behera, “Analysis of Sanskrit text: parsing and semanticnets," Springerlink, Sanskrit Computational Linguistics pp 200-218, vol. 5402, 2009.R.

[8] Patil, B., and M. Patil. "A review on implementation of Sandhi Viccheda for Sanskrit words." In Proceedings of the international conference in ICGTETM, IJCRT, vol. 5, no. 12, pp. 489-493. 2017.

[9] Patil, B., and M. Patil. "A review on implementation of Sandhi Viccheda for Sanskrit words." In Proceedings of the international conference in ICGTETM, IJCRT, vol. 5, no. 12, pp. 489-493. 2017.

[10] Bhadra, Manji, Surjit Kumar Singh, Sachin Kumar, Subash, Muktanand Agrawal, R. Chandrasekhar, Sudhir K. Mishra, and Girish Nath Jha. "Sanskrit analysis system

(SAS)." In Sanskrit Computational Linguistics: Third International Symposium, Hyderabad, India, January 15-17, 2009. Proceedings, pp. 116-133. Springer Berlin Heidelberg, 2009.

[11] Hellwig, Oliver. "Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit." In 4th Biennial workshop on less-resourced languages. 2015.

[12] 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 http://sanskrit.jnu.ac.in/rstudents/mphil/sachin.pdf

[13] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[14] Xue, Nianwen. "Chinese word segmentation as character tagging." In International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing, pp. 29-48. 2003.

[15] Shree, M. Rajani, Sowmya Lakshmi, and B. R. Shambhavi. "A novel approach to Sandhi splitting at Character level for Kannada Language." In 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), pp. 17-20. IEEE, 2016.

[16] 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, Belgium4909–4914. https://doi.org/10.18653/v1/D18-1530.

[17] Sushant Dave, Arun Kumar Singh, Dr. Prathosh A.P., and Prof. Brejesh Lall. 2021. Neural Compound-Word (Sandhi) Generation and Splitting in Sanskrit Language. In

8th ACM IKDD CODS and 26th COMAD (CODS COMAD 2021). Association for Computing Machinery, New York, NY, USA, 171–177. https://doi.org/10.1145/3430984.3431025

[18] Premjith B, Chandni Chandran V, Shriganesh Bhat, Soman Kp, and Prabaharan P. 2019. A Machine Learning Approach for Identifying Compound Words from a Sanskrit Text. In Proceedings of the 6th International Sanskrit Computational Linguistics Symposium, pages 45–51, IIT Kharagpur, India. Association for Computational Linguistics..

[19] Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit Word Segmentation Using Character-level Recurrent and Convolutional Neural Networks. In Proceedingsof the 2018 Conference on Empirical Methods in Natural Language Processing.Association for Computational Linguistics, Brussels, Belgium, 2754–2763. https://doi.org/10.18653/v1/D18-1295 .

[20] S. Sharma and M. Nirkhe, "Sanskrit Sandhi Splitting using LSTM-based Sequence-to-Sequence Models," in Proceedings of the International Conference on Natural Language Processing (ICON), 2018.K.

[21] Patel and R. Singh, "Transformer-based Models for Sanskrit Sandhi Splitting," in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2020

[22] A. Deshmukh and P. Joshi, "Bidirectional LSTM-CRF Models for Sanskrit Sandhi Splitting," in Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2019.

[23] N. Gupta and S. Kumar, "BERT-based Models for Sanskrit Sandhi Splitting," in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2021.

[24] R. Mishra et al., "Hybrid Models for Sanskrit Sandhi Splitting," in Proceedings of the IEEE International Conference on Computational Linguistics (COLING), 2019.