A Dataset for Sanskrit Word Segmentation

Amrith Krishna, Pavankumar Satuluri[*] and Pawan Goyal
[*]School of Linguistics & Literary Studies, Chinmaya Vishwavidyapeeth CEG Campus;
Dept. of Computer Science & Engineering, Indian Institute of Technology Kharagpur
amrith@iitkgp.ac.in

Abstract

The last decade saw a surge in digitisation efforts for ancient manuscripts in Sanskrit. Due to various linguistic peculiarities inherent to the language, even the preliminary tasks such as word segmentation are non-trivial in Sanskrit. Elegant models for Word Segmentation in Sanskrit are indispensable for further syntactic and semantic processing of the manuscripts. Current works in word segmentation for Sanskrit, though commendable in their novelty, often have variations in their objective and evaluation criteria. In this work, we set the record straight. We formally define the objectives and the requirements for the word segmentation task. In order to encourage research in the field and to alleviate the time and effort required in pre-processing, we release a dataset of 115,000 sentences for word segmentation. For each sentence in the dataset we include the input character sequence, ground truth segmentation, and additionally lexical and morphological information about all the phonetically possible segments for the given sentence. In this work, we also discuss the linguistic considerations made while generating the candidate space of the possible segments.

1 Introduction

Sanskrit was the prevalent medium of knowledge transfer in the demographic of Indian Subcontinent for over four millennia. The culture bearing language of India has about 30 million extant manuscripts that are potent for digitisation (Goyal et al., 2012). The last decade witnessed tremendous excitement in digitisation attempts of ancient manuscripts in Sanskrit. The Digital Corpus of Sanskrit[1], The Sanskrit Library[2] and GRETIL[3] are some such laudable efforts. These attempts aim to preserve the cultural heritage of the subcontinent embedded in the works written in Sanskrit.

The writings in Sanskrit follow a '*scriptio continua*' (Hellwig, 2016), thereby making Word Segmentation in Sanskrit a challenging task. Lack of visible markers in written scripts is a prevalent feature observed in numerous Asian languages. Additionally, the sentence constructs in Sanskrit show a high degree of inflection (Scharf and Hyman, 2011), phonemes at the word boundary undergo phonetic transformations called as '*sandhi*' (Goyal and Huet, 2016), and the sentence constructs in Sanskrit follow a loose word order (Hellwig, 2016; Kulkarni et al., 2015). The combination of the aforementioned properties makes the segmentation in Sanskrit a complex task.

Given an input Sanskrit sentence, the word segmentation task can be defined as identification of the semantically most valid split of the input sentence. There have been commendable efforts to tackle the word segmentation task in Sanskrit. Mittal (2010) designed Finite State Transducers (FST) incorporating the rules of *sandhi* obtained from documented grammatical tradition. With the defined FSTs, Mittal (2010) generates all possible splits followed by a probabilistic scoring procedure to select the ideal split. Natarajan and Charniak (2011) proposed 'S3 - Statistical Sandhi Splitter', a Bayesian word segmentation approach for Sanskrit. The work is an extension of Goldwater et al. (2006) and was adapted to handle *sandhi* formations. Hellwig (2015) proposed a neural model that jointly solves the problem of compound splitting and *sandhi* resolution. Krishna et al. (2016a) handled the word segmenta-

[1] http://kjc-sv013.kjc.uni-heidelberg.de/dcs/
[2] http://sanskritlibrary.org/
[3] http://gretil.sub.uni-goettingen.de/gretil.htm

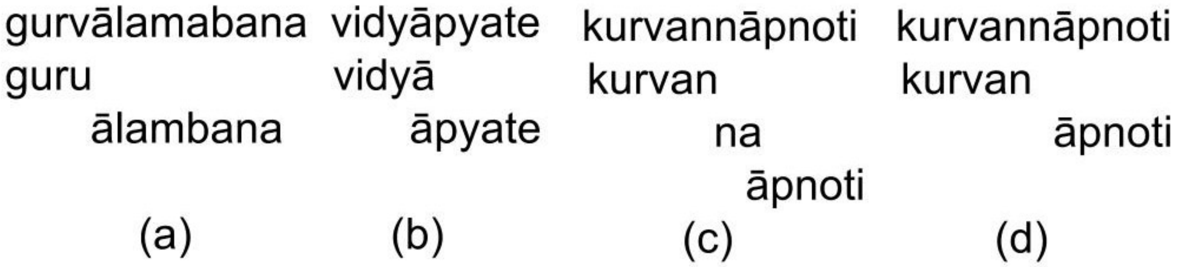


Figure 1: Example instances of sandhi formation in Sanskrit. a) Phonetic transformation of '*u*' and '*ā*' to '*vā*' in the joint form. b) '*ā*' and '*ā*' at the word boundaries of *vidyā* and *āpyate* join together to form a single '*ā*' in the final form. Both the split words have an overlap at the juncture (Goyal and Huet, 2016). c) and d) Two possible analyses for the sandhied chunk '*kurvannāpnoti*' (Krishna et al., 2016a), where c) is the negation of d).

tion problem as an iterative query expansion task. The authors used Path Constrained Random Walks (PCRW) (Lao and Cohen, 2010) for identifying word segments that are likely to co-occur in the given sentence. This work considers compound splitting as part of the word segmentation task itself.

As the task of Sanskrit word segmentation is gaining traction, it also calls for uniformity and easiness in comparing the competing models. For instance, the aforementioned approaches vary in their defined objectives and are evaluated under different settings, making a direct comparison of the models difficult. For example, Mittal (2010); Natarajan and Charniak (2011) do not discuss the effect of compounds and compound splitting on the dataset. Hellwig (2015) presents the same as a separate task from that of word segmentation, while Krishna et al. (2016a) do not make an explicit difference between both the tasks. Krishna et al. (2016a) report an F-Score of 90.61 % when tested on a curated dataset of 2148 sentences, compared to competing models with an F-Score of 70.07% (Natarajan and Charniak, 2011) and 66.26 % (Mittal, 2010). But, Krishna et al. (2016a) report an F-Score of 77.72 %, when the authors tested their method on a larger dataset of about 10,000 sentences obtained from a digitised corpus. Additionally, The aforementioned systems focus primarily on the correctness of the word-form predicted. Sanskrit is an agglutinative language and the same inflection of a lemma can signify multiple possible morphological classes. Therefore, correctness of the morphological class is also important, when the segmentation is performed. Though Krishna et al. (2016a) report the performance of their system when considering the correctness of lemma and morphological class prediction, they primarily focus on the word-from prediction task.

In perspective of the current scenario of this field, our contributions in this work are two fold:

1. We formally define the objective for word segmentation task in Sanskrit. We see word segmentation not as an end but as a means to facilitate further processing of text in Sanskrit. We define our requirements with an end-goal of making the current digitised content accessible to an end-user when seen from the perspective of an Information Retrieval system. To achieve this, the segmentation task should output the information that is valuable for the subsequent syntactic and semantic tasks such as POS Tagging, dependency parsing, sentence summarisation, etc. The distinction for the correctness of lemma and morphological class, and not just the correctness of the final word-from, is of utmost importance for this.

2. We release a dataset of 115,000 sentences which can be used for further research in word segmentation task in the Sanskrit. With this dataset we aim to alleviate the effort and time that often needs to be spent in preposssessing data. The pre-processing efforts often require the use of multiple sub-systems, and this can lead to inconsistencies with the assumptions made by each of the subsystems involved.

In Section 2, we discuss in detail about the challenges that need to be tackled in word segmentation in Sanskrit. Section 3 discusses the preliminaries followed by the formal definition of word segmentation task. Section 4 details the structure of the dataset we use. In Section 5, we explain various linguistic considerations that we have made, while preparing the dataset.

2 Challenges in Word Segmentation

Word segmentation is an important prerequisite for further processing of Sanskrit Texts. In addition to having no visible markers in Sanskrit sentence constructs, the linguistic peculiarities inherent in the language make the task a non-trivial one. Sanskrit, primarily being a language used orally, has passed on the euphonic assimilation of phones into the writing as well (Goyal and Huet, 2016). This leads to phonetic transformations at the word boundaries called as '*sandhi*'. Figure 1 shows some cases of sandhi. In Figure 1a, the words '*guru*' and '*ālambana*' join together to form '*gurvālambana*', where the '*u*' at the boundary of '*guru*' gets transformed to '*v*'. In case of *vidyā* + *āpyate* → *vidyāpyate*, the '*ā*' at the word boundaries of both the words join together to form a '*ā*'. Here, the position for '*ā*' in the joint form is shared by both the words. Due to *Sandhi*, the word boundaries disappear and the sounds are often replaced or elided. Given two words, the generation of the joint form or what can be called as the '*sandhied*' form is deterministic. The ancient grammar treatise *Aṣṭādhyāyī* mentions the *sandhi* rules by which phonemes at the word boundaries undergo changes, when two words are combined. But analysis of a sentence with *sandhi* in it leads to multiple phonetically valid segments. The analysis for a given sentence may often lead to several semantically valid segments as well. For example, consider the expression *kurvannāpnoti kilbiṣam*. The expression may be split into two possible statements, '*kurvan āpnoti kilbiṣam*' (While doing, you will accumulate sin) and '*kurvan na āpnoti kilbiṣam*' (While doing, you will not accumulate any sin) (Krishna et al., 2016a). The splits shown in Figure 1c and 1d gives a very specific example, where both the possible segmentation analyses differ by a negation term *na*. A more generic example is shown in Figure 2. Figure 2a shows all possible segmentations for the given sentence. Figure 2b the only analysis for the input sentence which is semantically valid.

Sandhi operation does not modify the sentence constructs at syntactic or semantic levels. It only makes difference at the phonetic level. There is no rationale for two words to undergo 'sandhi', other than the proximity of these words at the time of enunciation. Simply put, successive words tend to form sandhi, and it is the discretion of the composer to choose whether or not to perform *sandhi* between the pair of words. This brings us to the second challenge. Writings in Sanskrit follow free (loose) word order especially in poetry (Kulkarni et al., 2015; Melnad et al., 2015). The free word order in Sanskrit is a convenient tool for poets to arrange the words in accordance with the prescribed meter. By this, the proximity between two words cannot be deemed as an indicator for syntactic or semantic compatibility between them. We need to consider the entire sentence context while processing the text and use the entire co-occurrence context of the words in the sentence.

In addition to the aforementioned linguistic peculiarities, the fact that Sanskrit has a rich morphology, compounds the problem further. Sanskrit expresses high degree of inflection, with 90 and 72 different inflections for verbs and nouns respectively. The inflections for a word generally happen at the boundary of a word. The inflections lead to different final word-forms for a given lemma. To illustrate the problem, let us consider the character sequence '*nagarāṇi*'. Now, '*nagarāṇi*' can be analysed as 'na garāṇi' which means 'no diseases'. But the character sequence '*nagarāṇi*' can have an alternative analysis, where it is the nominative case neuter gender plural of the lemma '*nagara*' (town) and the word '*nagarāṇi*' is an inflection of the lemma '*nagara*'. The scenario points to an isntance where we have to decide whetehr there exists a split for the character sequence or not.

2.1 Computational Analysis of Sandhi

The rules related to *sandhi* are well documented in the ancient Sanskrit grammar treatise *Aṣṭādhyāyī* by *Pāṇini*. Hyman (2008) observed that the external *sandhi* can be computationally tackled using finite state methods. An efficient Finite State Transducer for segmentation in Sanskrit was later developed by (Huet, 2009). Sanskrit Heritage Reader (Goyal et al., 2012; Goyal and Huet, 2013, 2016), a lexicon driven morphological analyser for Sanskrit, when input with a sentence, provides all the phonetically valid segmentations for the given sentence. Figure 2a shows the shallow analysis provided by the Sanskrit Her-

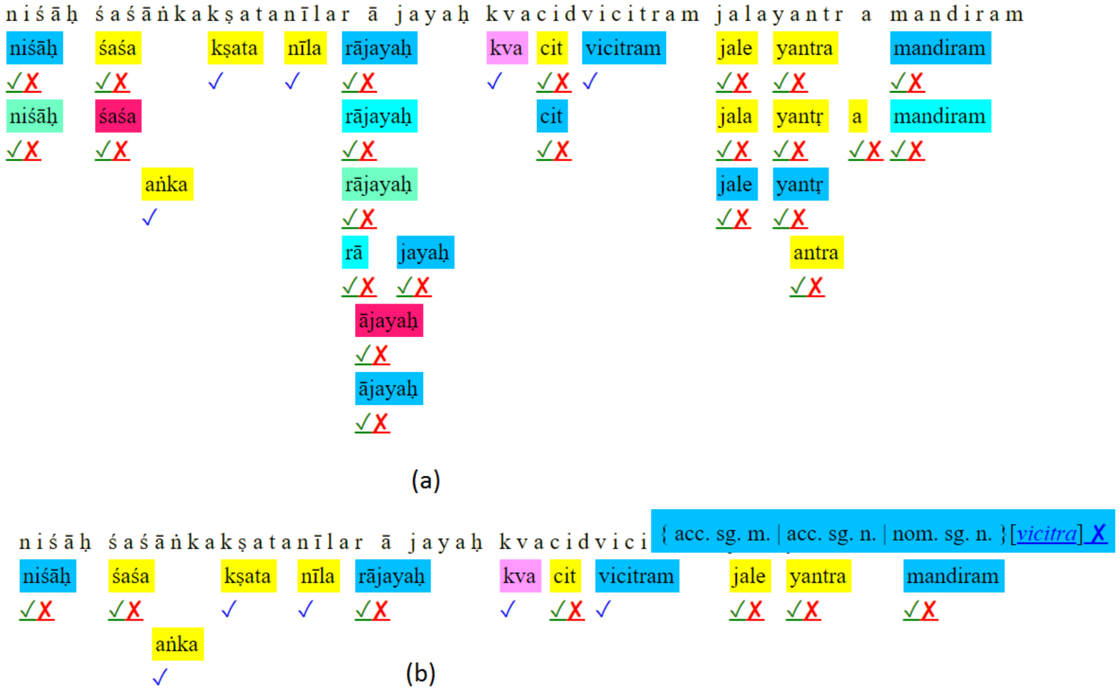


Figure 2: (a): Candidate segments for the sentence '*niśāḥ śaśāṅkakṣhatanīlarājayaḥ kvacidvicitam jalayantramandiram*' as output by the Sanskrit Heritage Reader (b): The correct segmentation for the given sentence. The image also shows the lemma information and possible morphological classes for the word '*vicitram*' (in a blue box) as shown by the morphological analyser

itage Reader for a given sentence. The analyser not only provides the possible word forms, but also gives the morphological analysis for each of the segments. Figure 2b shows the morphological analysis for the word *vicitram*. For the word *vicitram*, it shows that the lemma for the word is '*vicitra*', and the given word-form can belong to three possible morphological classes, namely accusative case singular with masculine or neuter, or it can be nominative case singular neuter class. Similar to the case of *Sandhi*, while the generation of word form for a given morphological class is deterministic, the analysis of given word form is not deterministic and can lead to ambiguities.

3 Sanskrit Word Segmentation Task

Given an input sentence s, represented as a sequence of characters, it is possible to obtain all the possible segmentations for the sentence as output by the Sanskrit Heritage Reader. For the word segmentation task, we assume that the analysis from Sanskrit Heritage Reader is available for an input sentence, and we reduce our task of word segmentation to finding the proper sequence of segments from the set of all candidate segments as output by the Heritage Reader. We define the word segmentation task formally, with the Heritage Reader being the cornerstone for the entire task. The task is described based on the definitions taken from Goyal and Huet (2016). Sanskrit Heritage Reader is a lexicon driven system and the system is built based on Finite State Methods. The system can be defined as a lexical juncture system.

A *lexical juncture system* on a finite alphabet Σ is composed of a finite set of words $L \subseteq \Sigma^*$ and a finite set R of rewrite rules of the form $u|v \rightarrow f/x\_\_$ (Kaplan and Kay, 1994) , with $x, v, f \in \Sigma^*$ and $u \in \Sigma^+$. In this formalization, Σ is the set of phonemes, R is the set of sandhi rules, and L is the vocabulary as a set of lexical items. We define, $z_i \in L$ as a 3-tuple (l, m, w), where l denotes the lemma of the word, m denotes the morphological class of the word and w denotes the inflected word form for the lemma l with morphological class m. Given a sentence s, a sandhi analysis for s, S_i can be seen as a sequence $\langle z_1, \sigma_1, k_1\rangle; ...\langle z_p, \sigma_p, k_p\rangle$. Here, $\langle z_j, \sigma_j, k_j\rangle$ is a segment with $z_j \in L$, $k_j \in \mathbb{N}$ denotes the position at which the word w_j begins in the sentence s and $\sigma_j = [x_j]u_j|v_j \rightarrow w_j \in R$ for $(1 \leq j \leq p)$, $v_p = \epsilon$ and $v_j = \epsilon$ for $j < p$ only if $\sigma_j = o$, subject to the matching conditions: $z_j = v_{j-1}y_jx_ju_j$

for some $y_j \in \Sigma^*$ for all $(1 \leq j \leq p)$, where by convention $v_0 = \epsilon$. Finally $s = s_1 ... s_p$ with $s_j = y_j x_j w_j$ for $(1 \leq j \leq p)$. ϵ denotes the empty word.

But for s, there can be multiple possible segmentation analyses. Let $\mathcal{S}$ be the set of all such possible analyses for s. We find a shared forest representation of all such segmentation analyses

$$D(\mathcal{S}) = \bigcup_{S_i \in \mathcal{S}} S_i$$

.

A segment $\langle z_j, \sigma_j, k_j \rangle \in D(\mathcal{S})$, iff $\langle z_j, \sigma_j, k_j \rangle$ exists in at least one S_i.

We define our word segmentation task formally as follows. For a given sentence s with the set of segmentation analyses $D(\mathcal{S})$, we need to identify a set of segments $\langle z_j, \sigma_j, k_j \rangle \in D(\mathcal{S})$, such that this sequence of segments corresponds to the ground truth analysis (segmentation) S_{GT}. Figure 2b shows the ground truth segments for the sentence '*niśāḥ śaśāṅkakṣhatanīlarājayaḥ kvacidvicitam jalayantramandiram*'

4 Dataset

In order to alleviate the time and effort that is often required to spend for preprocessing of Sanskrit data, we release a dataset of 115,000 Sanskrit sentences with all the candidate segments as mentioned in Section 3. Since the sentence constructs in Sanskrit follow a free word order, the proximity between two words cannot be used as an indicator for semantic and syntactic compatibility between both the words. So, we assume that all the words that co-occur in a sentence, are equally prone to influence one another. In order to reflect this in our dataset, we assume our candidate segment representation as a Graph $G(V, E)$. For the graph G, a node $v \in V$ is a unique segment $\langle z_j, \sigma_j, k_j \rangle$. There exists an edge $e \in E$ between every pair of vertices $v_i, v_j \in V$, provided v_i, v_j are not '*conflicting*' with each other. Two nodes are '*conflicting*' if they have an overlap in the position relative to the sentence and the overlapped portion does not adhere to any of the rules that follow *sandhi*.

Given a sentence in s with n words, the sentence will have t breaks (spaces) between the characters such that $t < n$. If $t = n - 1$, all the words are segmented. Otherwise, there are at least two words which are joined together and will be in the *sandhied* form. We call such fused forms as chunks in our dataset. For example in Figure 2a, there are four chunks. Every node is a possible candidate in the segmentation task. Two nodes are said to be conflicting, if they cannot co-occur in the given sentence, i.e, if two nodes share a character position with respect to the input, and the shared portion does not result in a proper *sandhi* transformation then the nodes are said to be conflicting. In Figure 2a, the words '*ājayaḥ*' and '*jayaḥ*' share a common portion of the input and hence if one of them exists in the sentence, the other needs to be eliminated. Formally, consider two nodes, represented as (k, z) and (k', z') in a given chunk. Here k and k' are the starting position (offsets) of these nodes relative to the chunk, let $|z|$ and $|z'|$ be the length of the words which nodes represent. We say that (k, z) and (k', z') *conflict* if $k \leq k' < k + |z| - 1$ or $k' \leq k < k' + |z'| - 1$ (Goyal and Huet, 2016). If two nodes are not conflicting, then there is a possibility that the two may co-occur in the sentence and hence one can become the context in resolving the other. Hence we add the edge to all such possible nodes.

4.1 The Corpus

We use the Digital Corpus of Sanskrit (DCS) for obtaining the sentences and the ground truth splits for our segmentation dataset. The DCS consists of about 560,000 sentences tagged with lemma and morphological information for each of the words in the corpus. The corpus essentially consists of partial or fully digitised versions of about 225 manuscripts in Sanskrit. The manuscripts are written as prose, poetry or a mix of both. The manuscripts range from different domains including science, philosophy, religion, literature and poetry. The time period sof the various digitised works vary for more than 1000 years. So, in essence, the corpus is a representative sample of various writing styles, time period and domains of writing.

Our dataset is a subset of the mentioned corpus. Our dataset contains all the candidate space segments as output by the Sanskrit Heritage Reader for the 115,000 sentences. Since DCS and the Sanskrit Heritage Reader have differences in the design decisions they have made, it was not entirely trivial to match the entries in the DCS with the candidate segments provided by the Sanskrit Heritage Reader. We made sure that the entries in DCS and Heritage reader candidates are matched for 115,000 sentences, taking into consideration the lemma matches, morphological classes and the compound splits.

Since, we require our dataset to be structured for automated processing and also interpretable by humans, we decided to use the XML based graph specification GraphML for representing the candidate space segments.

GraphML - GraphML (Brandes et al., 2001) is an XML based file format, specifically designed for handling Graphs. We considered graphML as the standard for the representation of segments primarily due to the human readability and structured storage of information the format provides, when compared to other space efficient binary formats like pickles or other structured formats like JSON. Additionally, standard libraries across various programming languages exist that can read GraphML structures and if required convert them to other existing formats including the aforementioned ones. By sticking onto a general purpose data representation we make the programming efforts required to process the data easier. Standard graph processing libraries (Schult and Swart, 2008; Leskovec and Sosič, 2016) can be used to apply various network metrics, graph algorithms and use the format for visualisation (Bastian et al., 2009) of the data as well. We represent each sentence in the dataset as a separate GraphML file.

Graph Structure - Each node in the graph stores a set of attributes, which show useful information about the node. The different attributes that are stored in the node and their description are as follows.

1. **Word** - The final inflected form of the segmented word that can be a possible candidate. The word is the generated form of a lemma after the affixation.

2. **Lemma** - The root word of the given form as recognized by the Sanskrit Heritage Reader.

3. **Morphological Information** - The morphological information about a given word as provided by the Sanskrit Heritage Reader.

4. **Morphological Tags by DCS** - The field consists of integer values which is used in DCS to represent the morphological information. We add the redundant information as the mapping between DCS and the Heritage Reader are not one to one and hence to cross check the correctness, we are keeping the redundant information.

5. **Chunk Number** - Every contiguous stream of sounds in the input sentence is referred to as a chunk. A chunk contains at least one word or *pada*. The position information of each candidate segment is stored relative to the chunk's position.

6. **Word Position** - The field stores integer values which basically refers to the starting position of a given candidate segment relative to the chunk. The first word of every chunk has a position 0.

7. **Word Length** - The length of the generated word form, i.e., the field 'word'.

8. **Pre-verbs** - Certain words might be prefixed with pre-verbs. We store the pre-verbs separately as an additional attribute. But this is also counted when we calculate the 'word length'.

Currently, in the graph we keep the morphological information provided by DCS corpus as well as the Heritage Reader. Though the information is redundant, it acts as a means of validating our mapping scheme. With the aforementioned node attributes we have one edge attribute in the graph. In the graph, we store the information about whether two nodes are conflicting or not by forming edges between them and labeling them with different values. All the edges marked with attribute value '1' are non-conflicting nodes and hence can potentially co-occur. All those node pairs with edges marked with '2' are those nodes which are conflicting with each other, in other words, at most only one among the pair will exist in the final solution. It can be observed that in some files the edge type attribute has a value of '-1'. This implies that the edge is between two nodes where one of the nodes provides supplementary information about the other and hence will not be part of the segmentation. For example, consider the word '*madya*', which is a noun in vocative case with its lemma as '*madya*'. But as an additional information, it can be observed that the word is derived from the root '*mad*' which is in causative periphrastic future tense. Now, '*mad*' is not directly part of the input sentence but is supplementing the node '*madya*'. So the edge between the nodes '*mad*' and '*madya*' will have a label -1. Additionally the position attribute of '*mad*' node will also have the value -1, implying that it is not a node in the candidate space, but is supplementary to one of the candidate nodes.

5 Discussion

In this section, we detail the various linguistic properties that we leverage for mapping the output provided by the Sanskrit Heritage Reader with the information from DCS.

5.1 Compounds and Named Entities

We find that some of the compounds, especially the exo-centric compounds (and few endocentric compounds), are treated as a single word unit in DCS. The component information for the compounds are not provided. This is probably done in order to make the analysis semantically correct, as the resulting compound might be a named entity referring to either the name of a person or the name of a place. For example, the word '*daśaratha*' is a compound word, so is '*rāmalakṣmaṇabharataśatrughnāḥ*' (Krishna et al., 2016a). '*Daśaratha*' literally refers to any person who has ten chariots. But, when used in the sentence '*rāmasya pitā daśarathaḥ asti*' (Daśaratha is the father of rāma), *daśarathaḥ* does not refer to any person who has ten chariots but to a specific person. In such circumstances, it is often desired not to split compounds and represent it in terms of the components, as it is semantically not correct. In DCS, named entities like '*daśaratha*' are not decomposed into their components, but are represented as single nouns. But, '*rāmalakṣmaṇabharataśatrughnāḥ*' is a conjunctive compound containing names of four different people. DCS decomposes the compound into its components. The Heritage Reader, which does not consider the semantic context, gives the component information for all of the compounds.

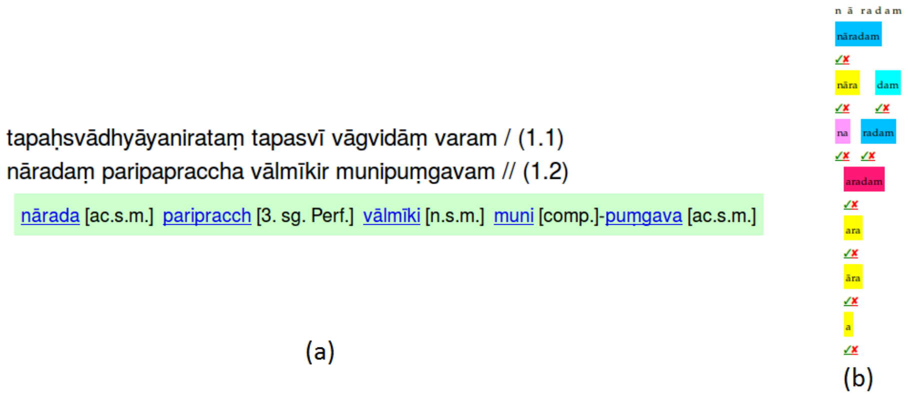
To provide mapping between both the resources in such cases, we replace the components suggested by the heritage reader with a *sandhied* versions of the compound components. The *sandhied* version is used as the candidate segment, so as to match it with the scheme of the DCS. But multiple component combinations may lead to the same *sandhied* versions. For example, DCS treats the word *nāradam* as a single unit which is desirable, on the other hand Heritage Reader produces different splits for the same word. Figure 3a shows the treatment of the word *nāradam* in DCS, while Figure 3b shows the analysis of the same word by the Heritage Reader. Here, all the possible component combinations that lead to the correct *sandhied* form are added as nodes in the graph but as supplementary nodes with an edge type value of -1. The supplementary information need not be of direct impact for the word segmentation task, and hence can be ignored for the task. But it will be beneficial when it comes to syntactic parsing tasks like dependency parsing or other semantic tasks.

5.2 Secondary Derivative Affixes

The secondary derivative affixes such as '*vat*' and '*tva*' are treated differently in both the systems. While, Sanskrit Heritage Reader remains faithful to the traditional means of analysis, by keeping the morpheme as part of the root word, DCS deviates from this representation. In DCS, the morphemes, i.e., the root word and the secondary derivative affix, are treated as separate words. For example, in the sentence '*śarātisarge śīghratvāt kālāntakayamopamaḥ*', consider the word '*śīghratvāt*'. The word is an ablative case neuter gender singular noun of '*śīghratva*'. Now, this can further be represented as the combination of the morphemes, '*śīghra*' and '*tva*', where the former is a noun and the latter is the secondary derivative affix. In heritage reader, this is represented as '*śīghratva*', implying the traditional analysis of the word into its morphemes. But, in DCS both the morphemes are represented as two individual lemma. Same applies with '*vat*' as well. We take care of this in our dataset.

5.3 Markers in Lemma

The inflections in compounds in Sanskrit are generally added to the final component of the compound (Krishna et al., 2016b). This implies that all the other components are devoid of the inflection and are in pure form. But, certain words in Sanskrit often have additional markers in them when the word is mentioned as its base form. But, when the base form is used as component, the markers are removed. There are entries in DCS where the form of the component is assumed to be the base form. For example, in the compound '*mahādeva*' (The great god or Lord *Śiva*), the compound has the components '*mahān*' (great) and '*deva* '(deity, god). But, the lemma or the base form for '*mahā*' is '*mahat*'. Heritage Reader follows this conven-



tapaḥsvādhyāyaniratam̩ tapasvī vāgvidām̩ varam / (1.1)
nāradam̩ paripapraccha vālmīkir munipum̩gavam // (1.2)

nārada [ac.s.m.] paripracch [3. sg. Perf.] vālmīki [n.s.m.] muni [comp.]-pum̩gava [ac.s.m.]

Figure 3: (a) - Morphological analysis of *nāradam* in the DCS. (b) - Morphological analysis of *nāradam* in the Sanskrit Heritage Reader

tion. But in DCS, '*mahān*' is considered as the lemma. We find such cases where similar issues are present and resolve them.

5.4 Phonetic Variations

In Sanskrit, '*m̩*', known as the *anusvāra*, when followed by one of the predefined set of phonemes from the phoneme list, can optionally undergo a transformation where it is replaced by a nasal (*anunāsika*) variation corresponding to the phone. This can be seen as an internal sandhi. DCS does not consider this internal transformation and the heritage reader does so. In effect there are five such different *anunāsika* variations possible corresponding to various places of articulation. In the case of *Śam̩kara*, the *m̩* is followed by '*k*'. The *anunāsika* for '*k*' is '*ṅ*'. Thus, *Śam̩kara* becomes *Śaṅkara*. Similarly, for *sam̩jaya* where *m̩* is preceded by '*j*', *sam̩jaya* gets transformed to '*sañjaya*'.

Phonetic Variations due to Pre-verbs - Linguistically, preverbs are treated as bound morphemes. But in DCS, there are numerous instances when a verb is prefixed with a preverb, the joint form is treated as a different lemma altogether. This distinction needs to be taken care of, as the Heritage Reader provides the original verbal root. Additionally, internal *sandhi* can take place when the preverb is prefixed to a verb. When obtaining the analysis for each of the morphemes, we need to split the morphemes by undoing the *sandhi* operation as well. But, in DCS whenever the preverb is not separated, the *sandhi* operation also remains embedded in the form. The internal *sandhi* also needs to be taken care of, in order to deal with the analysis from the heritage reader. For example in case of '*praṇam*', the word is an inflection of the verbal root '*nam*' with '*pra*' prefixed as a preverb. Here, due to internal *sandhi*, '*n*' transforms to '*ṇ*'. Similarly, '*s*' changes to '*ṣ*' in case of '*abhiṣic*', which is joint form for the morphemes '*abhi*' and '*sic*'.

6 Conclusion

We release the dataset hoping to catalyse the research in computational processing of Sanskrit. Here, we abstract out language specific details that are often required in handling the data, and makes the data accessible to researchers who otherwise required to rely on linguistic experts in Sanskrit. A considerable amount of time and expertise is often required in pre-processing the data and aligning the ground truth with the output of the morphological analyser. This acts as a barrier to the entry point in the field of Sanskrit Computational Linguistics. We present the pre-processed data by removing all the inconsistencies that are often faced. We expect to eventually roll out the data for all the 560,000 files. Currently we release 115,000 files. Please note that the GraphML files store directly the output of the Heritage Reader for the input sentence in a structured format. The ground truth segmentation is directly taken from DCS, which is manually tagged, and is assumed to be correct. The mapping between the schemes used for defining the morphological classes in both the Sanskrit Heritage Reader and the DCS is not completely unambiguous. Currently, we provide the morphological class information for each word in both the schemes. Hence, the dataset will not be affected by the changes that may occur to the said mapping between the schemes. The current dataset may not be directly usable for dependency parsing, as the current sentences are not properly aligned to sentence boundaries, especially for sentences in poetry formats. Computational means of finding sentence boundaries is a research topic of its own (Hellwig, 2016). Summarily, we bring clarity with regards to the requirements for the word segmentation task and release a dataset for the experiments. We hope to not only increase the collective productivity of the community which would have otherwise been spent on preprocessing, but also believe that this endeavour makes benchmarking of different systems straightforward, as their performance can be tested on the same dataset.

Acknowledgements

The authors are thankful to the anonymous reviewers for their valuable feedback. The authors would like to thank Gérard Huet for providing the Sanskrit Heritage Engine, to be installed on local systems and Oliver Hellwig for contributing the DCS Corpus. We are grateful to Amba Kulkarni, Peter Scharf and again Gérard Huet for their comments and feedback regarding the design and development of this dataset. The authors would also like to acknowledge the efforts put by Bhumi Faldu in programming and maintaining the codebase with which the dataset was created.

References

Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. 2009. Gephi: an open source software for exploring and manipulating networks. *ICWSM* 8:361–362.

Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. 2001. Graphml progress report structural layer proposal. In *International Symposium on Graph Drawing*. Springer, pages 501–512.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 673–680.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd*. pages 130–171.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling* 4(2):145–182.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1011–1028. http://www.aclweb.org/anthology/C12-1062.

Oliver Hellwig. 2015. Using recurrent neural networks for joint compound splitting and sandhi resolution in sanskrit. In *4th Biennial Workshop on Less-Resourced Languages*.

Oliver Hellwig. 2016. Detecting sentence boundaries in sanskrit texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 288–297. http://aclweb.org/anthology/C16-1028.

Gérard Huet. 2009. Sanskrit Segmentation, South Asian Languages Analysis Roundtable xxviii, Denton, Texas. South Asian Languages Analysis Roundtable XXVIII.

Malcolm D. Hyman. 2008. From Paninian Sandhi to Finite State Calculus. In *Sanskrit Computational Linguistics*. pages 253–265.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20,3:331–378.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016a. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 494–504. http://aclweb.org/anthology/C16-1048.

Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016b. Compound type identification in sanskrit: What roles do the corpus and grammar play? In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1–10. http://aclweb.org/anthology/W16-3701.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How Free is free Word Order in Sanskrit. In *Sanskrit Syntax*. The Sanskrit Library, USA.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning* 81(1):53–67.

Jure Leskovec and Rok Sosič. 2016. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8(1):1.

Keshav S Melnad, Pawan Goyal, and Peter Scharf. 2015. Meter identification of sanskrit verse. In *Sanskrit Syntax*. pages 325–346.

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, pages 85–90. http://www.aclweb.org/anthology/P10-3015.

Abhiram Natarajan and Eugene Charniak. 2011. s^3 - statistical sandhi splitting. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 301–308. http://www.aclweb.org/anthology/I11-1034.

Peter M Scharf and Malcolm D Hyman. 2011. Linguistic issues in encoding sanskrit. *The Sanskrit Library* .

Daniel A Schult and P Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*. volume 2008, pages 11–16.