Word Segmentation in Sanskrit Using Path Constrained Random Walks

Amrith Krishna, Bishal Santra*, Pavankumar Satuluri,
Sasi Prasanth Bandaru[#], Bhumi Faldu, Yajuvendra Singh[##] and Pawan Goyal
*Dept. of Electronics & Electrical Communication Engineering,
[#]Dept. of Electrical Engineering, [##]Dept. of Mathematics,
Dept. of Computer Science & Engineering
Indian Institute of Technology, Kharagpur, WB, India
amrith@iitkgp.ac.in

Abstract

In Sanskrit, the phonemes at the word boundaries undergo changes to form new phonemes through a process called as *sandhi*. A fused sentence can be segmented into multiple possible segmentations. We propose a word segmentation approach that predicts the most semantically valid segmentation for a given sentence. We treat the problem as a query expansion problem and use the path-constrained random walks framework to predict the correct segments.

1 Introduction

Word segmentation is an essential step for most of the text processing tasks. In Sanskrit texts, it is very common that the phonemes at the word boundaries undergo changes to form new phonemes, through a process termed *sandhi*. It also makes the word boundary between the words that undergo *sandhi*, indistinguishable. Proximity between phonemes (*saṃhitā*) is the sole criteria for applying *sandhi*, thus phonetic transformation takes place between two consecutive words. This poses a big challenge in identifying individual words in a given sentence, as the same fused form can be segmented into many possible segmentations. Though, there has been considerable advancements in tackling word segmentation challenges faced in other languages including Chinese, Korean and Arabic, a direct application of those approaches is not possible in Sanskrit texts due to the phenomena of *sandhi*. Since the *sandhi* rules are well documented in the tradition, all the syntactically valid segmentations (splits) of a given sentence in Sanskrit can be enumerated. Sanskrit Heritage Reader provides a nice compact interface to show all the valid solutions (Huet and Goyal, 2013). The main challenge is to identify the most relevant solution from all the possible segmentations. The relevance of contextual information in identifying semantically relevant segments is a well-accepted fact (Hellwig, 2009). Mittal (2010), Natarajan and Charniak (2011) have successfully applied statistical methods for *sandhi* splitting in Sanskrit. But, both the approaches relied heavily on word co-occurrence features as their context with minimal usage of the morphological information. We, unlike the previous methods, propose a segmentation approach which effectively combines the morphological features in addition to the word co-occurrence features from a manually tagged corpus of more than 400,000 sentences (Hellwig, 2009). We treat the problem as a query expansion problem that iteratively selects a segment amongst the competing segmentations and then continues to select the next correct segment with the information from extended context. The algorithm terminates when no more candidates remain to be evaluated. We use the scalable "path-constrained random walks (PCRW)" (Lao and Cohen, 2010) framework with linguistically motivated Inductive Logic Programming (ILP) formulations for finding the correct segmentation. Using extensive experiments, we find that our approach outperforms the existing approaches by a significant margin.

2 Challenges in Sandhi Splitting

Processing of Sanskrit text poses challenges of its own due to the *sandhi* phenomena. For example, consider the phonemic change at the boundary for the case of *ramā* + *īśaḥ* → *rameśaḥ*. Here the phonemes *ā* at the end of the previous word and *ī* at the beginning of the subsequent word combine together to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 494–504, Osaka, Japan, December 11-17 2016.

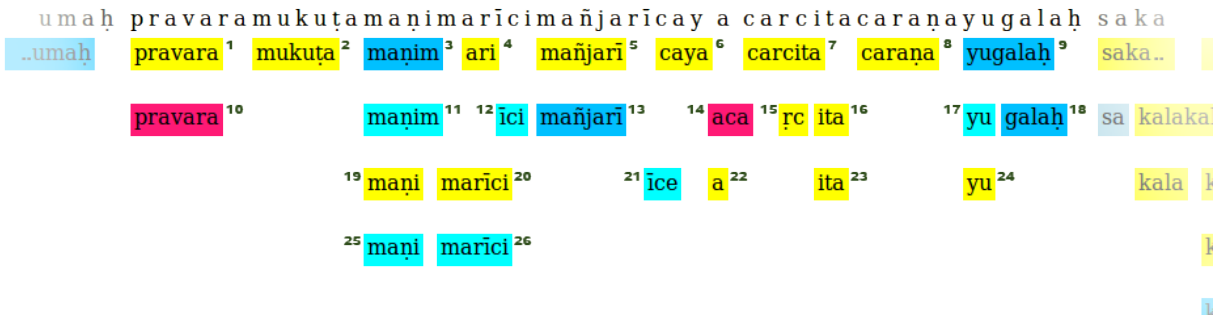


Figure 1: Possible segmentations for the compound "*pravaramukuṭamaimarīcimañjarīcayacarcitacaraṇa-yugalaḥ*" from the Sanskrit Heritage Reader. Each colour specifies a separate morphological class (Goyal and Huet, 2013; Huet and Goyal, 2013).

form the phoneme *e*. Such a transformation does not result in any change in morphological, syntactic or semantic properties of the individual words. However, there are phonetic variations resulting in the words to change their written forms.

As already mentioned, finding a semantically valid segmentation for a given sentence is a challenging task, due to the large number of possible segmentations. For example, consider the sentence, "*tatra sakalārthikalpadrumaḥ pravaramukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ sakalakalāpāraṅgato'maraśaktirnāma rājā babhūva*" - [from the text *Pañcatantram (kathāmukham)*] translates to, "There was a king namely Amaraśakthi, who was the wish-granting tree (fulfiller of all the desires) to the whole group of seekers, and the pair of whose feet was covered with a stream of rays originating from the gems in wreaths of eminent noble kings, and who was proficient in all arts". The sentence of 9 words (3 of them being compound words) can be segmented into more than half a million possible syntactically valid segmented sentences[1].

It can also be observed that multiple semantically valid splits are possible for a given string. For example, the phrase *śveto dhāvati* may be decomposed in two different ways. Both, the splits, *śvetaḥ dhāvati* (The white [one, horse] runs) and *śvā itaḥ dhāvati* (The dog runs [towards] here) are valid and give two different interpretations of the same phrase. This re-emphasizes the need for the context of the whole sentence in determining the correct set of candidates. The Sanskrit grammar poses no restriction on words in undergoing sandhi, other than the proximity of the phonemes at word boundaries. Consider the statement *dadarśāvatarantamambarāddhiraṇyagarbhāṅgabhuvaṃ muniṃ hariḥ* (Lord Viṣṇu saw Brahma's son Nārada, descending from the sky). Here, in this statement, the words *ambarāt* (from the sky) and *hiraṇyagarbhāṅgabhuvaṃ* (Nārada, son of Lord Brahma) are not related semantically, barring this they undergo euphonic transformation and fuse together to form *ambarāddhiraṇyagarbhāṅgabhuvaṃ*.

In dealing with segmentation, there are scenarios where the compound words should not be split into the constituents. Exo-centric compound refers to an external entity and the compound should be considered as a whole. Otherwise, the statement in which the compound is used need not result in a meaningful sentence. For example, the word '*daśaratha*' refers to a person and it does not refer to any of its constituents '*daśa*' (ten) or '*ratha*' (chariot). But during the analysis, it is required to use the compound component information, or else there might not be sufficient distributional information about each segment. Compound formation in Sanskrit is highly productive in nature. In the sentence from *Pañcatantram* menioned earlier, the compound,"*pravaramukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ*" (The pair of whose feet was covered with a stream of rays originating from the gems in wreaths of eminent noble kings,) is an Exo-centric (*Bahuvrīhi*) compound which consists of 9 constituents. Figure 1 shows the possible segmentation for the mentioned compound. The correct segmentation is the collection of words numbered 1-2-19-20-5-6-7-8-9. In such cases, it may be difficult to obtain distributional information about the compound if we analyse it without decomposing the compound into its components. But, after the analysis, if we do not join the compound components back to the original compound, then it alters the meaning of the sentence.

[1]Using the Sanskrit Heritage Reader available at http://sanskrit.inria.fr/

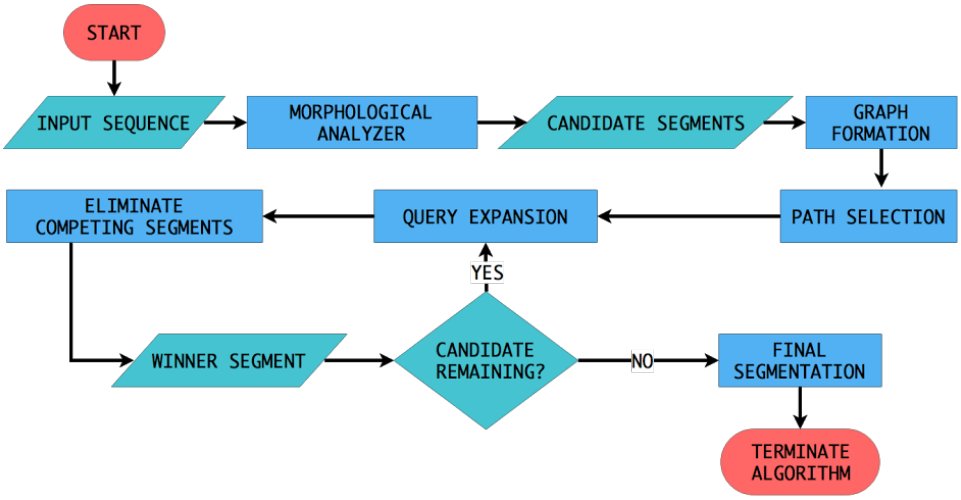


Figure 2: Flowchart for the Proposed Approach

3 Method

We treat the word segmentation problem as a query expansion problem. An input sentence is passed through the Sanskrit Heritage Reader[2]. The Sanskrit Heritage Reader provides all the possible segmentations for the input sentence along with the morphological information for each of the segments, similar to those in Figure 1. The output segments will henceforth be referred to as candidates. We now conceptualise a graph with these candidates as the nodes in the graph, in which we perform path-constrained random walks (PCRW) (Lao and Cohen, 2010) using a set of pre-defined path types. We start with the most promising candidate(s) as our initial query node and perform PCRW to obtain a winner node among the candidates. We add the winner to the extended context and repeat the process until no more candidates remain to be evaluated. We eliminate all the candidates conflicting with the winner node whenever a winner node is selected. The conflicting nodes are eliminated using positional information and *Sandhi rules*. Figure 2 provides the flowchart for the entire process.

3.1 Graph Formation

With the segments from the morphological analyser, we form a weighted multi-digraph $G(V, E, W)$, such that each node in the vertex set V represents a candidate. Every node has three attributes, the word-form, the lemma and the POS tag (morphological information) of the given segment. In this work, POS denotes the morphological information that each word-form possesses based on its usage in the sentences as directly obtained from the Sanskrit Heritage Reader. For nouns, the gender, the plurality and the grammatical case of the word-forms are obtained. For verbs the tense, number and person are the relevant morphological information that we obtain. We now construct multiple edges between all the candidate nodes except those which are conflicting with each other. We decide whether two nodes are conflicting using the position information and *sandhi* analysis. Consider two segments (k, z) and (k', z') in a given sentence. Here k and k' are the starting position (offsets) of these segments relative to the sentence, z and z' are the length of the segments. We say that (k, z) and (k', z') *conflict* if $k \leq k' < k + |z| - 1$ or $k' \leq k < k' + |z'| - 1$ (Huet and Goyal, 2013).

For every non-conflicting pair of nodes, we form edges with varying edge weights where the edge weights are decided by the possible combinations of attribute pairs at these nodes. Since each node has 3 attributes we form a total of 9 directed edges (in E) between every pair. The edge weights (in W) are defined as the co-occurrence probability of the attribute value at the target node given the attribute value at the source node. The probability values can be computed from a suitable corpus.

[2]http://sanskrit.inria.fr/DICO/reader.fr.html

3.2 Language Model

We represent our corpus, from which we obtain the distributional information for our task, as a Graph $G_2(V_2, E_2, W_2)$. We add every unique lemma, POS tag and word-form that occur in the corpus as a vertex in the graph. So, we define V_2, the vertex set as the union of the vocabulary of each of the three attributes. Now we form directed edges (in E_2) between every pair of nodes that co-occur in a sentence in the corpus. We calculate the edge weights (in W_2) using the expression 1.

$$P_{co}(j|i) = \frac{count(j,i)}{count(i)} \tag{1}$$

Here $count(i,j)$ denotes the total number of sentences in the corpus in which the nodes i and j co-occur. $count(i)$ is the count of documents in which node i occurs. For example, in order to compute the edge-weight for an edge from a POS node 'gen.m.sg.' to a lemma node '*hari*', we calculate $P(hari|gen.m.sg.)$, i.e, conditional probability of the lemma '*hari*' co-occurring with any word having the POS tag 'gen.m.sg.' in the same sentence. Similarly, we find edge weights from every possible pair of nodes that co-occur in the given corpus.

3.3 Path Selection using PCRW

With PCRW, the framework supports forming paths of any arbitrary length and also to formulate the constraints for path formations. The constraints we impose for path formation are termed as path types. We define various linguistically motivated path-types as Inductive Logic Programming (ILP) formulations. The path types are defined in Table 1. We form all possible paths in the graph G which satisfy the constraints defined in various path-types. In our formulation of the path types, a single edge path-type is defined as shown in expression 2.

$$Attribute^{constraint}_{node\ id} \xrightarrow{P_{co}} Attribute^{constraint}_{node\ id} \tag{2}$$

$$POS^{Noun}_{i} \xrightarrow{P(j|i)} Lemma^{Verb}_{j} \tag{3}$$

$$Constraint = \{\mathcal{P}(Noun), \mathcal{P}(Verb), \mathcal{P}(Compound), \mathcal{P}(Indeclinable), *\} \tag{4}$$

We define a node-type as $Attribute^{constraint}_{node\ id}$. Here $Attribute_{node\ id}$ denotes the attribute value of the node in G with the id of the node given as the subscript. The superscript is a constraint that specifies the requirements a node in G must satisfy to be in the path of the specified type. Expression 3 is a path-type of length one. Here, the source node in the expression is POS_i, i.e., it expects the POS tag value of node i. The source node in the path must be a noun and the target node must be a verb. In the sentence in Figure 1, the only eligible nodes to be the source node for this path type are nodes numbered 3,13,9 and 18. There are only two nodes which can be the target node which are 10 and 14. Hence, there are eight possible paths of the path-type given in expression 3. Now, consider one of the eight possible paths $nom.sg.f._{13}^{Noun} \xrightarrow{P_{co}(aca|nom.sg.f)} aca_{14}^{Verb}$. We obtain the edge weight as $P_{co}(aca|nom.sg.f)$ from W_2, where 'aca' and '$nom.sg.f$' are two nodes in G_2

Please note that the constraint 'Noun' is defined as the set of all the possible POS tags that a noun word-form can assume. Each POS tag in the Noun set conveys the case, gender and plurality of a word. We similarly define disjoint sets of POS tags for verbs, compounds and indeclinables. The set 'Constraint' is defined in the expression 4, where $\mathcal{P}$ denotes a power set. By defining 'Constraint' as a power set, we can group different POS tags as a single constraint to be used. For example, in $Lemma^{verb}_{i} \longrightarrow Lemma^{Noun-\{nom.sg.m,inst.sg.m.\}}_{j}$, the set $\{nom.sg.m, i.sg.m.\}$ is an element of $\mathcal{P}$, where 'nom' and 'i' signify nominative and instrumental cases respectively. The edge here starts with a verb node and looks for any noun node other than 'nom.sg.m.' and 'i.sg.m.' cases. By design, we do not allow POS tags of noun, verbs or compounds to be combined, as $\mathcal{P}$ does not contain such a combination as its element. We specify '$*$' as a wild-card constraint which allows any POS tag.

Sl. No.	Linguistic Proprety	Path types
1	General Relations	$Lemma_i^* \longrightarrow Lemma_j^*$
2		$POS_i^* \longrightarrow POS_j^*$
3	Compounds	$p(Lemma_i^{compound} \vert Lemma_j^{compound})$
4	Expectancy	$POS_i^{noun} \longrightarrow POS_j^{verb} \longrightarrow Lemma_k^{noun}$
5		$Lemma_i^{noun} \longrightarrow Lemma_j^{verb} \longrightarrow Lemma_k^{noun}$
6	Proximity Nouns	$Word_i^{noun} \longrightarrow Lemma_j^{noun}$, where,$POS_i = POS_j$
7		$Lemma_i^{noun} \longrightarrow Lemma_j^{noun}$, where,$POS_i \neq POS_j$
8	Proximity Verbs	$Lemma_i^{verb-\{absolutive\}} \longrightarrow Lemma_j^{absolutive}$
9		$Lemma_i^{verb-\{gerund\}} \longrightarrow Lemma_j^{gerund}$

Table 1: Path Types for path selection

Path-type starting from node i to node s is defined recursively as follows:

$$PathType_{POS_i}^{Lemma_s} : POS_i^{Noun} \xrightarrow{P(j_1|i)} Lemma_{j_1}^{Verb} \xrightarrow{P(j_2|j_1)} Lemma_{j_2}^{Verb} .... \xrightarrow{P(s|j_k)} Lemma_s^{Verb} \tag{5}$$

$$P(PathType_{POS_i}^{Lemma_s}) = P(j_1|i).P(s|j_k) \prod_{l=2...k} P(j_l|j_{l-1}) \tag{6}$$

In expressions 5 and 6, $P(j|i)$ denotes $P_{co}(Attribute_j|Attribute_i)$. The value of the path from node i to node s is defined as the product of all the weights (from W_2) of the edges in the path. The value so obtained can be thought as a random walk score, where a random walk traverses over graph G_2 starting at node $Attribute_i$ and terminating at $Attribute_s$ with all other nodes in the path as intermediate nodes.

Table 1 defines various path types that we use in our system. Though only a sample of path types is shown, it is implied that path types of other attributes with a similar structure are also formed. In Table 1, we also mention the linguistic properties that motivated us to formulate the path types.

General Relations: The general relations use the '$*$' constraint implying any node can form path of this type. Here we restrict ourselves to the path types which have same attribute types at both the ends of the path (edge). We also use Kneser-Ney smoothing (Kneser and Ney, 1995) for word-form to word-form co-occurrence and lemma to lemma co-occurrence values, though we do not do the same for POS to POS co-occurrence. Smoothing is not provided for POS to POS as we find that all the possible co-occurrences are already captured in the language model.

Compounds: We find bigram probabilities with various compound components since the compounds strictly follow a sequential order in their formation.

Expectancy of a verb: In Sanskrit, there always exists a modifier-modified relation between the words and the verb in the sentence, known as *kāraka* roles or semantic roles. Expectancy plays a vital role in establishing these modifier-modified relations in a sentence. Every verb expects different semantic roles in its sentence usages, which are marked with different syntactic markers in Sanskrit. The path types 4 and 5 try to capture this notion. For example, consider the sentence *rāmaḥ brāhmaṇāya gāṃ dadāti* (Ram gives a cow to the Brahmin). Here the noun words *rāmaḥ* and *brāhmaṇāya* are in different POS tags serving the roles of subject and beneficiary. Though we do not need to know the exact roles, but with the path *rāma⟶dadāti⟶brāhmaṇāya*, we attempt to capture the probability of both the words to co-occur with the verb (both the noun words need not co-occur in the same sentence in the corpus. Refer expression 6), while both being in different POS tags. The path can be an approximation for expectancy.

Proximity - Kulkarni et al. (2015) defined Proximity in Sanskrit as 'the representation of word meanings without any intervention'. But in Sanskrit poetry, this may not be strictly followed as the poet needs to maintain the meter of the verses. Kulkarni et al. (2015) conclude that there is a violation of proximity in case of *viśeṣaṇa-viśeṣya*, i.e., the modifier-modified relation between two nouns, both in case of prose as well as poetry In such situations, morphological markers are one of the ways to identify related words. Paths 6 - 9 attempt to capture this notion. In path 6, we find the co-occurrence probabilities of

two nodes when both appear in the same POS. Path 6 tries to capture the relation of *viśeṣaṇa-viśeṣya* between two nouns. Path 7 exactly finds the inverse where we find the co-occurrence probabilities of two nodes when they are not in the same POS. Path 8 is different from Expectancy path-types as there is no verb in between the nodes.

3.4 Query Expansion with PCRW

In graph G, we form paths of all possible types starting from query node and terminating at any candidate node. No two nodes in a path should be conflicting with each other. For Path types 1, 2 (General relations) and 6 in Table 1, we perform random walks with restarts over graph G to capture the structural properties of the graph. All the three paths are of length 1, i.e. they are edges. Here we relax the criteria for path formation and we form edges with all the possible node pairs in G other than the conflicting nodes. The edge weights are obtained again from W_2. With other path-types, we do not perform random walks over G, but as mentioned the path scores can be seen as random walk traversal over the graph G_2. This effectively helps us to leverage the structural properties of the content graph structure G and the graph G_2, which ideally represents the distributional properties in the language (corpus). We then do a weighted sum of scores from all the paths and the winner node is selected from the combined scores. We eliminate the conflicting nodes with winner nodes and then use the extended context to select the subsequent winner. We choose the initial query node by identifying the longest candidate from the set of candidates. The heuristic is inspired by the maximum matching approach used by Chen and Liu (1992). In case of a tie, we select the candidate with the minimum number of conflicts.

3.5 Supervised Parameter Estimation for Path-Types

Every path type can be considered as a feature and the random walk score at each instance for a pair of (query, source) node is a feature-value. We need to estimate the relative importance of each path type. Our input is a pair of words, the source word and target words. We generate all possible positive instances and down-sample the negative training instances from a training set of 5000 sentences. The label to each instance is a +1 or 0 depending on whether or not the pair of words co-occur in a sentence in the corpus. We use logistic regression with L-BFGS (Andrew and Gao, 2007) optimisation procedure to handle over-fitting. We follow the same optimisation procedure as followed in Lao and Cohen (2010). The authors argue that this approach is superior to the alternative one-weight-per-edge-label approach.

4 Experiments

4.1 Comparison with the Existing Approaches

We compare our system's performance with the existing approaches for *sandhi* splitting by presenting the results of our system on a data-set of 2148 strings, henceforth to be referred to as 'Test 2148', which was used by Natarajan and Charniak (2011) in their work. Table 2 presents the results from 4 different systems. OT2 and OT3 are the best performing variants from Mittal (2010) based on recall and precision respectively. NC4 is the best performing version of the algorithm proposed by Natarajan and Charniak (2011). It is a supervised Bayesian word segmentation approach which employs Gibbs sampling to sample from the posterior distribution of a training set of 25000 split strings. We tested our system on the test data with our pre-trained model and we do not use the training data used in Natarajan and Charniak (2011). Precision is the proportion of correct predictions amongst all the predictions made by the systems. Recall is the proportion of correct predictions which matched with the ground truth to the total number of segments in the ground truth.

Since the dataset does not provide any compound component information, we join the compound components from the prediction by applying the *sandhi* rules in order to obtain the compound before the evaluation. Our approach provides an overall improvement of 28.81% in F-score from the previously best performing system (NC4). A total of 1784 of 2148 (83.05%) were segmented with an F-score of one.

	Test 2148		
System	P	R	F
S	92.38	88.91	90.61
NC4	76.21	64.84	70.07
OT2	63.96	68.74	66.26
OT3	70.52	66.61	68.51

Table 2: Performance evaluation of our proposed system, S with the state of the art, NC4 (Natarajan and Charniak, 2011), OT2 and OT3 (Mittal, 2010).

	DCS 90K			Held-out Dataset		
System	P	R	F	P	R	F
B1	37.02	50.81	42.83	36.89	50.74	42.72
B2	42.25	58.62	49.115	42.04	58.45	48.91
B3	65.13	76.78	70.48	65.07	76.70	70.41
S	74.11	84.92	79.15	73.28	82.73	77.72

Table 3: Performance evaluation of our proposed system, S with the competing baselines B1, B2 and B3 over 100,000 sentences from DCS

4.2 Dataset

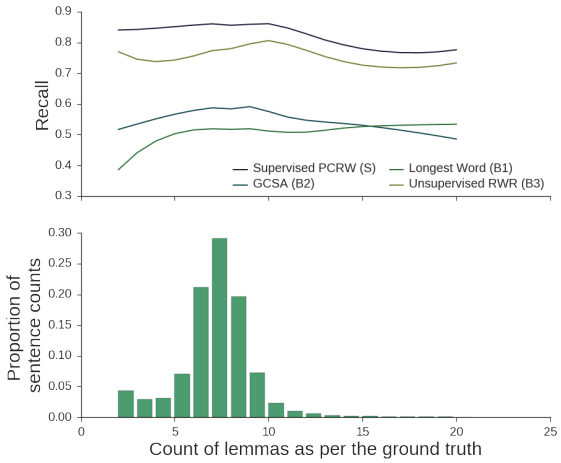
We use the Digital Corpus of Sanskrit (DCS), to build the language models and train our model. The corpus is a digitised collection of ancient works written in both prose and poetry styles with more than 430,000 segmented sentences with 66,000 unique lemmas. The corpus is a rich resource for the task as it contains over 3.2 million segmented tokens which are tagged with lemma and morphological information through expert supervision. We find only about 4067 Out Of Vocabulary (OOV) lemmas in the candidate space for which there was no co-occurrence evidence.

4.3 Baselines

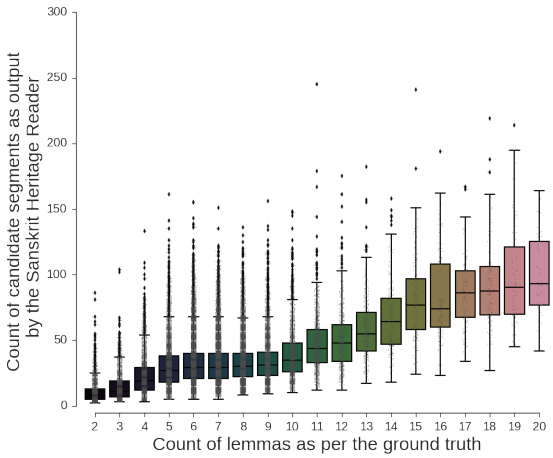
In Section 4.1 we showed the effectiveness of our model over the existing approaches. The systems described in Section 4.1, however, do not handle the context of an entire sentence. Additionally, the systems do not consider the morphological information in their training phases, and hence it will be unfair to use those systems for further analysis. We propose the following methods as our baselines. In all the systems we use the morphological analyser output as the input to predict the correct segmentation.
Longest Word selection (B1) - Inspired by the maximum matching approach from Chen and Liu (1992), we iteratively select the longest word available from the given set of possible segmentations. At each step of the iteration, we eliminate the candidates conflicting with the current winner. This method does not use any morphological information as such.
Greedy candidate selection approach (B2) - Here we consider the morphological analyser output to be a tree, and we perform a greedy selection that maximises the overall likelihood of the selection. In this approach, we combine the co-occurrence probabilities mentioned in paths 1,2 and 6 in Table 1.
Unsupervised RWR - General Relations (B3) - In this approach, we form the graph G for the sentence, similar to as mentioned in Section 3. However, we only use path-types 1,2 and 6 of Table 1. We perform Random Walks with Restarts (RWR) over the graph G and then average the scores from different random walk runs to obtain a winner node.
Supervised PCRW (S) - This is our proposed system, as defined in Section 3.

4.4 Results

We consider 100,000 sentences of varying length (not used for training in Section 3.5) from the DCS corpus. From the 100,000 sentences, 90,000 of the sentences, henceforth to be referred to as "DCS 90K", were used for calculating the co-occurrence values and we keep the remaining 10,000 sentences as held-out data which was not used at any point in our framework. Figure 3b shows the box plot for the number of sentences based on the frequency distribution of lemmas in ground truth against the count of possible candidates as given by the Sanskrit Heritage Reader for all the 100,000 files. From the plot, it is evident that the count of candidates increases manifold with the increase in lemmas. We eliminate all the sentences with exactly one lemma from the dataset which amounts to less than 1% of the total dataset. Table 3 compares the performance of each of the system in terms of precision, recall and F-Score. Our final system S performs the best with an increase of 13.79% and 10.60% in precision and recall respectively from our strong baseline B3 for the DCS 90K dataset. For the held-out dataset, the results remain more or less consistent with those obtained over DCS 90K for all the systems. Our



(a) Recall Vs. GT Frequency distribution



(b) Candidate Vs. GT Frequency distribution

Figure 3: Analysis of systems based on frequency distribution of ground truth (GT) lemmas

system S has an F-Score of 76.58% which is slightly less than the F-score attained on the original dataset, which was 79.15%. Our system could predict all the correct segmentations for as many as 38.64% of the sentences against the 19.57% fully correct predictions of B3.

Variation in Recall based on length of lemmas - The line graphs in the Figure 3a illustrate the recall of each system over the frequency distribution on the 'DCS 90K' based on the number of lemmas per sentence in the ground truth. The x-axis represents the count of lemmas in a sentence, and the y-axis represents average recall. The bar chart represents the proportion of sentences with respect to the count of lemmas in ground truth. The average length of sentences in our dataset is 6.87. It can be observed that our system works better for the sentences of all the lengths.

5 Discussion

A close inspection of the Tables 2 and 3, reveals that for all the systems, precision is higher than recall for the first dataset, and vice-versa in the DCS. On our manual inspection of the dataset, it is found that the DCS being manually tagged, relies on the context to decide whether to decompose a compound into its components. For example the word '*daśaratha*' is a compound word, so is *kṛṣṇārjunanakulasahadevāḥ*. In DCS, '*daśaratha*' being an exo-centric compound, it will not be decomposed into its components. But '*kṛṣṇārjunanakulasahadevāḥ*' is a copulative compound containing names of four different people. DCS decomposes the compound into its components. But, our morphological analyzer decomposes all the compounds into its components as it only considers the syntactic conditions. Unlike the case with first dataset, DCS does not contain segmented word-forms but only segmented lemmas. Hence joining of components is sometimes not possible as grammar cases for the nouns is not available in the system to perform *sandhi*. Due to this issue we find that multiple components which are predicted are treated as separate lemmas. This results in lower precision for the system, especially when 75% of the vocabulary in DCS consists of compounds.

Another intriguing challenge is with the case of compounds where it can be split with different interpretations such that, two possible segmentations result into being the exact negation of each other. Even from the semantic context, it will be difficult to consider whether the negation is required or not. There are a considerable number of sentences in which such issues occur. For example, the string *kurvannāpnoti kilbiṣam* may be decomposed in two different ways '*kurvan āpnoti kilbiṣam*' (While doing, you will accumulate sin) and '*kurvan na āpnoti kilbiṣam*' (While doing, you will not accumulate any sin) Both of them have completely opposite meanings.

6 Related Work

During the last couple of decades, computational analysis of Sanskrit and its grammar has gained considerable attention from the computational linguistics community. Hyman (2008) observes that the Pāṇinian *sūtras* for external *sandhi* can be modeled using a finite state grammar. Huet (2009) developed a tool for segmentation in Sanskrit by building an efficient Finite State Automata (FSA). With efforts from Huet, an automated analyser for exhaustive syntactically valid analysis of a Sanskrit sentence, called as the 'Sanskrit Heritage Reader' is available (Goyal et al., 2012; Goyal and Huet, 2013). However, the system provides all the possible syntactically valid segmentation and it requires human assistance to choose the relevant segmentations so as to form the semantically correct sentence. We use the 'Sanskrit Heritage Reader' in our pipeline to obtain the set of all possible segmentations.

Mittal (2010) has proposed an approach for automated Sanskrit segmentations by relying on the maximum a posteriori estimate from all possible *sandhi* splits for a given string. Natarajan and Charniak (2011) has proposed 'S3 - Statistical Sandhi Splitter', a Bayesian word segmentation approach which can handle *sandhi* formations. Hellwig (2015) proposed a neural network based approach that jointly solves the problem of compound splitting and sandhi resolution. Mochihashi et al. (2009), Johnson et al. (2006) developed non-parametric Bayesian approaches which have been applied on word segmentation tasks. The work uses a Dirichlet process model inspired from the model of Goldwater et al. (2006). Kuncham et al. (2015) have developed a language independent sandhi splitter for agglutinative languages using a strucutred prediction approach.

Word Segmentation challenges for in languages like Arabic, Japanese, Korean and Chinese are extensively studied. Though, in theses languages, identifying the proper word boundary is a challenge, none of these have to deal with the 'Sandhi' as such. In Chinese word segmentation, Xue (2003) presented a character tagging approach. Wang et al. (2014) proposed a method based on dual decomposition (Rush et al., 2010) to combine word-based and character based approaches in an efficient framework. Yao and Huang (2016) proposed a bi-directional RNN with long short-term memory (LSTM) units which makes use of character embeddings. In our approach, we use the PCRW framework proposed by Lao and Cohen (2010). Gao et al. (2013) tackled the problem of query expansion using PCRW for web-log queries. PCRW has been widely used in heterogeneous information networks. The framework has been effectively put to use in Never Ending Language Learning (NELL) (Mitchell et al., 2015; Lao et al., 2011).

7 Conclusion

In this paper we presented an approach to Sanskrit word segmentation using supervised PCRW. We treated the problem as a query expansion problem. In our approach, the input sentence is treated as a graph. The candidate segments form the nodes and there exists an edge between every pair of nodes except the conflicting ones. In Sanskrit, since there is no guarantee of proximity to be maintained between words, we presume that treating the input as a sequence as followed in models like linear-chain Conditional Random Fields (CRF) and Hidden Markov Models might be a serious limitation for the task. Also, with the rich feature space that we employ, inference in CRF Models might be a bottleneck for the model's performance (Doppa et al., 2014). We find that the inclusion of morphological information by means of linguistically motivated ILP path-types greatly increases the performance of the system as much as by 12.30% in F-Score. Our system also outperforms the existing best approach by 28.82% in F-Score. We also showed the effectiveness of our system in a dataset with a collection of both prose and poetry. Our approach is scalable and can be applied to larger datasets as well.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. The authors are grateful to Dr. Oliver Hellwig for providing the DCS Corpus and Dr. Gérard Huet for providing the Sanskrit Heritage Engine, to be installed at local systems, and helping with other queries related to Sanskrit Heritage Reader. They are also grateful to Mr. Unnikrishnan T A for his suggestions on the implementation of the framework.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM.

Keh-Jiann Chen and Shing-Huan Liu. 1992. Word identification for mandarin chinese sentences. In *Proceedings of the 14th conference on Computational linguistics-Volume 1*, pages 101–107. Association for Computational Linguistics.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Structured prediction via output space search. *Journal of Machine Learning Research*, 15:1317–1350.

Jianfeng Gao, Gu Xu, and Jinxi Xu. 2013. Query expansion using path-constrained random walks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 563–572. ACM.

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*, pages 673–680. Association for Computational Linguistics.

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, Gérard P Huet, Amba P Kulkarni, Peter M Scharf, and Ralph Bunker. 2012. A distributed platform for sanskrit processing. In *COLING*, pages 1011–1028.

Oliver Hellwig. 2009. Sanskrittagger: A stochastic lexical and POS tagger for sanskrit. In *Sanskrit Computational Linguistics*, pages 266–277. Springer.

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

Gérard Huet and Pawan Goyal. 2013. Design of a lean interface for Sanskrit corpus annotation. In *Proceedings of ICON 2013, the 10th International Conference on NLP*, pages 177–186.

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.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

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

Prathyusha Kuncham, Kovida Nelakuditi, Sneha Nallani, and Radhika Mamidi. 2015. Statistical sandhi splitter for agglutinative languages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 164–172. Springer.

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

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Vipul Mittal. 2010. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90. Association for Computational Linguistics.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics.

Abhiram Natarajan and Eugene Charniak. 2011. S3-statistical saṃdhi splitting. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 301–308. Association for Computational Linguistics.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Mengqiu Wang, Rob Voigt, and Christopher D. Manning. 2014. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, MD*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yushi Yao and Zheng Huang. 2016. Bi-directional lstm recurrent neural network for chinese word segmentation. *arXiv preprint arXiv:1602.04874*.