

# An ‘Ekalavya’ Approach to Learning Context Free Grammar Rules for Sanskrit Using Adaptor Grammar

Amrith Krishna, Bodhisattwa Prasad Majumder\*, Pawan Goyal

Dept. of Computer Science and Engineering, IIT Kharagpur, India

\*Walmart Labs, India

amrith@iitkgp.ac.in, bodhisattwapm2017@email.iimcal.ac.in,

pawang@cse.iitkgp.ernet.in

## Abstract

1 This work presents the use of Adaptor Grammar, a non-parametric Bayesian approach  
2 for learning (Probabilistic) Context Free Grammar productions from data. In Adaptor  
3 Grammar, we provide the set of non-terminals followed by a skeletal grammar. The  
4 productions and the associated probability for the productions are automatically learnt  
5 by the system from the usages of words or sentences, i.e., the dataset. No prior linguistic  
6 knowledge, other than the skeletal grammar, is provided to the system. The system  
7 completely learns the grammar structure by observing the data, we call this approach an  
8 ‘Ekalavya’ approach. In this work, we discuss the effect of using Adaptor grammars for  
9 Sanskrit at word-level supervised tasks such as compound type identification and also  
10 in identifying source and derived words from corpora for derivational nouns. In both  
11 of the works, we show the use of sub-word patterns learnt using Adaptor grammar as  
12 effective features for the supervised task. We also present our novel approach of using  
13 Adaptor Grammars for handling Structured Prediction tasks in Sanskrit. We present the  
14 preliminary results for word reordering task in Sanskrit. We also outline our plan for the  
15 use of Adaptor grammars for Dependency Parsing and Poetry to Prose Conversion tasks.

## 16 1 Introduction

17 The recent trends in Natural Language Processing (NLP) community suggest an increased ap-  
18 plication of black-box statistical approaches such as deep learning. In fact, such systems are  
19 preferred as there has been increase in performance of several NLP tasks such as machine trans-  
20 lation, sentiment analysis, word sense disambiguation etc. (Manning, 2016). In fact, MIT  
21 Technology review reported the following regarding Noam Chomsky’s opinion about the ex-  
22 tensive use of ‘purely statistical methods’ in AI. The report says that “derided researchers in  
23 machine learning who use purely statistical methods to produce behaviour that mimics some-  
24 thing in the world, but who don’t try to understand the meaning of that behaviour.” (Cass,  
25 2011).

26 Chomsky quotes, “It’s true there’s been a lot of work on trying to apply statistical models to  
27 various linguistic problems. I think there have been some successes, but a lot of failures. There  
28 is a notion of success ... which I think is novel in the history of science. It interprets success  
29 as approximating un-analysed data.” (Pinker, 2011). Norvig (2011), in his reply to Chomsky  
30 comes in defence of statistical approaches used in the community. Norvig lays emphasis on the  
31 engineering aspects of the problems that the community deals with and the performance gains  
32 achieved in using such approaches. He rightly attributes that, while the generative aspects of  
33 a language can be deterministic, the analysis parts can be ambiguous. Hence, the use of prob-  
34 abilistic approaches is often necessary for engineering success. Additionally, in real world, the  
35 language use is not free from usage errors. The deviation from the linguistic rules in use can be  
36 seen as noise in the dataset, and those cases need to be handled separately. The use of statistical  
37 approaches provides a convenient means of achieving the same. In Manning (2016), the author  
38 quotes the work of Paul Smolensky, “Work by Paul Smolensky on how basically categorical

39 systems can emerge and be represented in a neural substrate (Smolensky and Legendre, 2006).  
40 Indeed, Paul Smolensky arguably went too far down the rabbit hole, devoting a large part of  
41 his career to developing a new categorical model of phonology, Optimality Theory (Prince and  
42 Smolensky, 1993).” This is an example where the linguistics and the statistical computational  
43 models had a successful synergy, fruitful for both the domains.

44 The Probabilistic Context Free Grammars (PCFGs) provide a convenient platform for ex-  
45 pressing linguistic structures with probabilistic prioritisation of the structures it accept. It has  
46 been shown that PCFGs can be learnt automatically using statistical approaches (Horning,  
47 1969). In this work, we look into Adaptor grammar (Johnson et al., 2007b), a non-parametric  
48 Bayesian approach for learning grammars from the observations, say, sentences or word usages  
49 in the language. When given a skeletal grammar along with the fixed set of non terminals,  
50 Adaptor grammar learns the right hand side of the productions and the probabilities associated  
51 with them. The grammar does so just by observing the dataset provided to it, and hence the  
52 name ‘Ekalavya’ approach.

53 The use of Adaptor grammars for linguistic tasks provides the following advantages for a  
54 learning task.

- 55 1. Adaptor grammars in effect output valid PCFGs, which in turn are context free grammars,  
56 and thus are valid for linguistic representations.
- 57 2. It helps to encode linguistic information which is already described in various formalisms  
58 via the skeletal grammars. Thus domain knowledge can effectively be used. The only  
59 restriction here might be that the expressive power of the grammar is limited to that of a  
60 Context Free Grammar.
- 61 3. By leveraging the power of statistics, we can obtain the likelihood of various possible parses,  
62 in case of structural ambiguity during analysis of a sentence.
- 63 4. While the proposed structures might not be as competitive in performance as with the  
64 black-box statistical approaches, the interpretability of such systems is a big plus. Grammar  
65 experts can look into the individual snippets where the rules are learnt and interpret what  
66 the system learns.

67 In Section 2, we discuss the preliminaries regarding Context Free Grammars, Probabilistic  
68 CFGs and Adaptor Grammar. In Section 3, we discuss the use of Adaptor grammars in various  
69 NLP tasks for different languages. We then describe the work performed in Sanskrit with Adap-  
70 tor grammars in Section 4. We then discussion future directions in Sanskrit tasks, specifically  
71 for multiple structured prediction tasks.

## 72 2 Preliminaries - CFG and Probabilistic CFG

73 Context Free Grammar was proposed by Noam Chomsky who initially termed it as phrase  
74 structure grammar. Formally, a Context Free Grammar  $\mathcal{G}$  is a 4-tuple  $(V, \Sigma, R, S)$ , where  $V$   
75 is a set of non-terminals,  $\Sigma$  is a finite set of terminals,  $R$  is the set of productions from  $V$  to  
76  $(V \cup \Sigma)^*$ , where  $*$  is the ‘Kleene Star’ operation.  $S$  is an element of  $V$  which is treated as the  
77 start symbol, which forms the root of the parse trees for every string accepted by the grammar.  
78 The language that can be generated by the Non-terminal  $X$  can be represented as  $\mathcal{L}_X$ . So, the  
79 language that can be generated by the grammar  $\mathcal{G}$  is  $\mathcal{L}_S$ .

80 The productions in Context Free Grammars are often handcrafted by linguistic experts. it  
81 is common to have large CFGs for many of the real life NLP tasks. It is common that a given  
82 string can have multiple possible parses for the given grammar. This is due to the fact that a  
83 Context Free Grammar contains all possible choices that can be produced from a given Non-  
84 terminal (O’Donnell, 2015). The grammar neither provide a deterministic parse nor prioritises  
85 the parses. This leads to structural ambiguity in the grammar. Probabilistic Context Free

S  $\longrightarrow$  HB | BC  
 H  $\longrightarrow$  'Word1' | BD  
 B  $\longrightarrow$  'Word2' | 'Word3' | 'Word4'  
 C  $\longrightarrow$  'Word3' | 'Word5'  
 D  $\longrightarrow$  'Word5' | 'Word2'

$V = \{S, H, B, C, D\}$   
 $\Sigma = \{\text{Word1}', \text{Word2}', \text{Word3}', \text{Word4}', \text{Word5}'\}$

Figure 1: An example of a Context Free Grammar

86 Grammars (PCFGs) have been introduced to weigh the probable trees when the ambiguity arises,  
 87 and thus provide a means for prioritising the desired rules. A PCFG is a 5-tuple  $(V, \Sigma, R, S, \theta)$ ,  
 88 where  $\theta$ , denotes a vector of real numbers in the range of  $[0, 1]$  indexed by productions of  $R$ ,  
 89 subject to

$$\sum_{X \rightarrow \beta \in R_X} \theta_{X \rightarrow \beta} = 1$$

S  $\longrightarrow$  HB **0.3** | BC **0.7**  
 H  $\longrightarrow$  'Word1' **0.8** | BD **0.2**  
 B  $\longrightarrow$  'Word2' **0.1** | 'Word3' **0.4** | 'Word4' **0.5**  
 C  $\longrightarrow$  'Word3' **0.9** | 'Word5' **0.1**  
 D  $\longrightarrow$  'Word5' **0.55** | 'Word2' **0.45**

$V = \{S, H, B, C, D\}$   
 $\Sigma = \{\text{Word1}', \text{Word2}', \text{Word3}', \text{Word4}', \text{Word5}'\}$

Figure 2: Example of a Probabilistic Context Free Grammar corresponding to CFG shown in Figure 1

90 The probabilities associated with all the productions of a given non terminal should add upto  
 91 1. The probability of a given tree is nothing but the product of the rules which are used to  
 92 construct the tree. A given vector  $\theta_X$  denotes the parameters of a multinomial distribution that  
 93 have the non terminal  $X$  on their left hand side (LHS) (O'Donnell, 2015) .

94 Note that PCFGs make two strong conditional independence assumptions (O'Donnell, 2015):

- 95 1. The decision about expanding a non-terminal depends only on the non-terminal and the  
 96 given distribution for that non-terminal. No other assumptions can be made.
- 97 2. Following from the first assumption, a generated expression is independent of other expres-  
 98 sions.

99 There are numerous techniques suggested for estimation of weights for the productions in  
 100 PCFG. The Inside Outside algorithm is a maximum likelihood estimation approach based on  
 101 the unsupervised Expectation maximisation parameter estimation method. Summarily, the  
 102 algorithm starts by initialising the parameters with a random set of values and then iteratively  
 103 modifies the parameter values such that the likelihood of the training corpus is increased. The  
 104 process continues until the parameter values converge, i.e., no more improvement of the likelihood  
 105 over the corpus is possible.

106 Another way of estimating parameters is through Bayesian Inference approach (Johnson et  
 107 al., 2007a). Given a corpus of strings  $\mathbf{s} = s_1, s_2, \dots, s_n$ , we assume a CFG  $\mathcal{G}$  generates all the  
 108 strings in the corpus. We take the dataset  $\mathbf{s}$  and infer the parameters  $\theta$  using Bayes' theorem

$$P(\theta|\mathbf{s}) \propto P_{\mathcal{G}}(\mathbf{s}|\theta)P(\theta)$$

where,

$$P_{\mathcal{G}}(\mathbf{s}|\theta) = \prod_{i=1}^n P_{\mathcal{G}}(s_i|\theta)$$

109 Now, the joint posterior distribution for the set of possible trees  $\mathbf{t}$  and the parameters  $\theta$  can  
 110 be obtained by

$$P(\mathbf{t}, \theta|\mathbf{s}) \propto P(\mathbf{s}|\mathbf{t})P(\mathbf{t}|\theta)P(\theta) = \left(\prod_{i=1}^n P(s_i|t_i)P(t_i|\theta)\right)P(\theta)$$

111 To calculate the posterior distribution, we assume that the parameters in  $\theta$  are drawn from  
 112 a known distribution termed as the prior. We assume that each non terminal in the grammar  
 113 has a given distribution which need not be same for all. For a non terminal, the multinomial  
 114 distribution is indexed by the respective productions and since we use Dirichlet prior over here,  
 115 each production probability  $\theta_{X \rightarrow \beta}$  has a corresponding Dirichlet parameter  $\alpha_{X \rightarrow \beta}$ . Now, either  
 116 through Markov Chain Monte Carlo Sampling approaches (Johnson et al., 2007a) or through  
 117 variational inference or a hybrid approach, the parameters are learnt (Zhai et al., 2014).

118 However, this approach as well does not deal with the real bottleneck, which is to come up  
 119 with relevant rules which can solve a task for a given corpus. For large datasets, the CFGs  
 120 should have large set of rules and it is often cumbersome to come up with rules by experts alone.  
 121 Non-Parametric Bayesian Approaches has been proposed as modifications for PCFGs. Roughly,  
 122 the Non-parametric Bayesian approaches can be seen as learning a single model that can adapt  
 123 its complexity to the data (Gershman and Blei, 2012). The term non-parametric does not imply  
 124 that there are no parameters associated with the learning algorithm, but rather it implies that  
 125 the number of parameters is not fixed, and increases with increase in data or observations.

126 The most general version of learning PCFGs goes by the name of Infinite HMM or Infinite  
 127 PCFG (Johnson, 2010). In infinite PCFG, say for the model described in Liang et al. (2007),  
 128 we are provided with a set of atomic categories and a combination of these categories as rules.  
 129 Now, depending on the data, the learning algorithm learns the productions and the number  
 130 of possible non-terminals along with the probabilities associated with them (Johnson, 2010).  
 131 Another variation that is popular with the Non-Parametric Grammar induction models is the  
 132 Adaptor grammar (Johnson et al., 2007b). Here, the number of non-terminals remains fixed and  
 133 is set manually. But, the production rules and their corresponding probabilities are obtained by  
 134 inference. The productions are obtained for a subset of non-terminals which are ‘adapted’, and  
 135 it uses a skeletal grammar to obtain the linguistic structures.

136 An Adaptor Grammar is a 7-tuple  $\mathcal{G} = (V, \Sigma, R, S, \theta, A, C)$ . Here  $A \subseteq V$  denotes non-terminals  
 137 which are adapted, i.e., productions for the non terminals in  $A$  will automatically be learnt from  
 138 data.  $C$  is the Adaptor set, where  $C_X$  is a function that maps a distribution over trees  $\mathcal{T}_X$  to a  
 139 distribution over distributions over  $\mathcal{T}_X$  (Johnson, 2010).

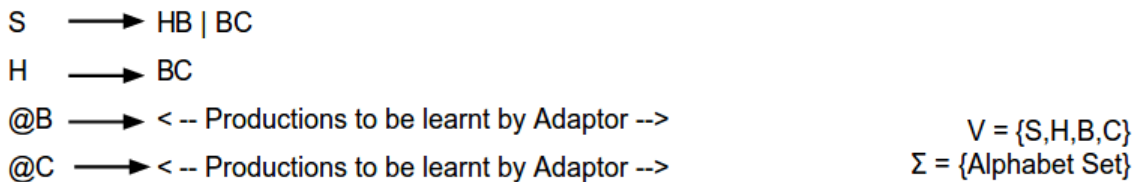


Figure 3: Example of an Adaptor Grammar. The non-terminals marked with an ‘@’ show that they are adapted. The productions will be learnt from data, where each production is a variable length permutation of subset of the elements in the alphabet set

The independence assumptions that exist for PCFGs are not anymore valid in the case of Adaptor Grammars (Zhai et al., 2014). Here the non-terminal  $X$  is defined in terms of an-

other distribution  $H_X$ . Now the adaptors for each of the non-terminal  $X$ ,  $C_X$ , can be based on Dirichlet Process or a generalisation of the same, termed as Pitman-Yor Process. Here  $TD_X(G_{Y_1}, G_{Y_2}, \dots, G_{Y_m})$  is a distribution over all the trees rooted in the non-terminal  $X$

$$H_X = \sum_{X \rightarrow Y_1 \dots Y_m \in R_X} \theta_{X \rightarrow Y_1 \dots Y_m} TD_X(G_{Y_1}, G_{Y_2}, \dots, G_{Y_m})$$

$$G_X \sim C_X(H_X)$$

### 3 Adaptor Grammar in Computational Linguistics

Adaptor Grammar has been widely used in multiple morphological and syntactic tasks for various languages. Adaptor Grammar has been initially shown for word segmentation task in English (Johnson et al., 2007b). A sentence with no explicit word boundaries were given as observations and the task was to predict the actual words in the sentence. The task is similar to tasks for variable length motif identification.

Adaptor Grammars has been introduced by Johnson et al. (2007b) as a non-parametric Bayesian framework for performing inference of syntactic grammar of a language over parse trees. A PCFG (Probabilistic Context Free Grammar) and an adaptor function jointly defines an Adaptor grammar. The PCFG learns the grammar rules behind the data generation process and the adaptor function maps the probabilities of the generated parse trees to substantially larger values than of the same under the conditionally independent PCFG model

Adaptor grammars have been very effectively used in numerous NLP related tasks. Johnson (2010) has drawn connections between topic models and PCFGs and then proposed a model with combined insights from adaptor grammars and topic models. While LDA defines topics projecting documents to lower dimensional space, Adaptor grammar defines the distribution over trees. The author also projects a hybrid model to identify topical collocations using the power of PCFG encoded topic models. Adaptor grammars are also used in named entity structure learning. Zhai et al. (2016) has used adaptor grammars for identifying entities from shopping related queries in an unsupervised manner.

The word segmentation task is essentially identifying the individual words from a continuous sequence of characters. This is seen as a challenging task in computational cognitive science as well. Johnson (2008a) used Adaptor Grammar for word segmentation on the Bantu Language, ‘Sesotho’. Author specifically showed how the grammar with additional syllable structure yields better F-score for word segmentation task than the usual collocation grammar. Similar study has been carried out by Kumar et al. (2015). The authors present the mechanism to learn complex agglutinative morphology with specific examples of three of four Dravidian languages, Tamil, Malayalam and Kannada. Furthermore, authors specifically have stressed upon the task of dealing with *sandhi* using finite state transducers after producing morphological segment generation using Adaptor grammars. Adaptor grammar succeeds in leveraging the knowledge about the agglutinative nature of the Dravidian language, but refrains from modelling the specific morphotactic regularities of the particular language. Johnson further demonstrates the effect of *syllabification* on word segmentation task using PCFGs (Johnson, 2008b). Johnson further motivates the unsupervised way of performing word segmentation, grammar induction tasks by extracting the collocational dependencies between words (Johnson and Demuth, 2010).

Due to its nature of generalizability, Adaptor grammar has been used for a variety of tasks. Hardisty et al. (2010) achieves state-of-the-art accuracy in perspective classification using adaptive naïve bayes model – the adaptor grammar based non-parametric Bayesian model. Besides this, adaptor grammar has been proven to be effective in grammar induction (Cohen et al., 2010). Grammar induction is an unsupervised syntax learning task. Authors achieved considerable results along with the finding that the variational inference algorithm can be extended to the logistic normal prior instead of the Dirichlet prior. Neubig et al. (2011) proposed an unsupervised model for phrase alignment and extraction where they claimed that their method

183 can be thought of as an adaptor grammar over two languages. Zhai et al. (2016) has presented  
184 a work, where the authors attempted to identify relevant suggestive keywords to a typed query  
185 so as to improve the results for search in an e-commerce site. The authors previously presented  
186 a new variational inference approach through a hybrid of Markov chain Monte Carlo and varia-  
187 tional inference. It has been reported that the hybrid scheme has improved scalability without  
188 compromising the performance on typical common tasks of grammar induction.

189 Botha and Blunsom (2013) presented a new probabilistic model which extends Adaptor gram-  
190 mar to make it learn word segmentation and morpheme lexicons in an unsupervised manner.  
191 Stem derivation in Semitic languages such as Arabic achieves better performance using this  
192 mildly context-sensitive grammar formalism. Again, Eskander et al. (2016) recently investigated  
193 with Adaptor Grammars for unsupervised morphological segmentation to establish a claim of  
194 language-independence. Keeping aside other baselines such as morphological knowledge input  
195 from external sources and other cascaded architectures, adaptor grammar proved to be outper-  
196 forming in majority of the cases.

197 Another use of Adaptor grammar has been seen in identification of native language (Wong  
198 et al., 2012). Authors used adaptor grammar in identifying n-gram collocations of arbitrary  
199 length over a mix of Parts of Speech tags and words to feed them as feature in the classifier. By  
200 modelling the task with syntactic language models, authors showed that extracted collocations  
201 efficiently represent the native language. Besides grammar induction, Huang et al. (2011) fur-  
202 ther uses Adaptor grammar for machine transliteration. The PCFG framework helps to learn  
203 syllable equivalent in both languages and hence aids to the automatic phonetic translation. Fur-  
204 thermore, Feldman et al. (2013) recently explored a Bayesian model to understand how feedback  
205 from segmented words can alter the phonetic category learning of infants due to access of the  
206 knowledge of joint occurrence of word-pairs.

## 207 4 Adaptor Grammar for Sanskrit

208 Adaptor Grammar have also been used for Sanskrit as well, mainly as a means of obtaining  
209 variable length character n-grams to be used as features for classification tasks. Below, we  
210 describe two different applications, compound type identification, as well as identifying the  
211 *Taddhita* suffix for derivational nouns.

### 212 4.1 Variable Length Character n-grams for compound type identification<sup>1</sup>

213 Krishna et al. (2016) used adaptor grammars for identifying patterns present in different types  
214 of compound words. The underlying task was, given a compound word in Sanskrit, identify  
215 the type of the compound. The problem was a multi-class classification problem. The classifier  
216 needed to classify a given compound into one of the four broad classes, namely, *Avyayībhāva*,  
217 *Dvandva*, *Bahuvrīhi*, *Tatpuruṣa*.

218 The system is developed as an ensemble based supervised classifier. We used Random Forests  
219 classifier with easy ensemble approach to handle the class imbalance problem persisting in the  
220 data. The classifier had majority of its labels in *Tatpuruṣa*. The presence of *Avyayībhāvaw* was the  
221 least. The classifier incorporated rich features, consisting of rules from *Aṣṭādhyāyī* pertaining to  
222 compounds, variable length character n-grams obtained from adaptor grammar and incorporat-  
223 ing noun pairs from *amarakośa* that follows a selected set of relations.

224 We capture semantic class specific linguistic regularities present in our dataset using variable  
225 length character n-grams and character n-gram collocations shared between compounds using  
226 adaptor grammars.

227 We learn 3 separate grammars namely, G1, G2 and G3, with the same skeletal structure in  
228 Figure 4a, but with different data samples belonging to *Tatpuruṣa*, *Bahuvrīhi* and *Dvandva* re-  
229 spectively. We did not learn a grammar for *Avyayībhāva*, due to insufficient data samples for

---

<sup>1</sup>The work has been done as part of the compound type identification work published in Krishna et al. (2016). The Authors wish to inform that there will be overlap in the content discussed here and the aforementioned work.

230 learning the patterns. We use a '\$' marker to indicate the word boundary between the com-  
 231 ponents, where the components were in sandhi split form. A '#' symbol was added to mark  
 232 the beginning and ending of the first and the final components, respectively. We also learn a  
 233 grammar G4, where the entire dataset is taken together along with additional 4000 random pair  
 234 of words from the Digital Corpus of Sanskrit, where none of the words appeared as a compound  
 235 component in the corpus. The co-occurrence or the absence of it was taken as the proxy for  
 236 compatibility between the components. The skeletal grammar in Figure 4b has two adapted  
 237 non-terminals, both marked by '@'. Also, the adapted non-terminal 'Word' is a non-terminal  
 238 appearing as a production to the adapted non-terminal 'Collocation'. The '+' symbol indicates  
 239 the notion of one or more occurrence of 'Word', as used in regular expressions. This is  
 240 not standard to use the notation in productions as per context free grammar. This is ideally  
 241 achieved using recursive grammars in CFGs with additional non-terminals. But, in order to  
 242 present a simpler representation of skeletal grammar we followed this scheme. In subsequent  
 243 representations we will be using recursiveness instead of the '+' notation.

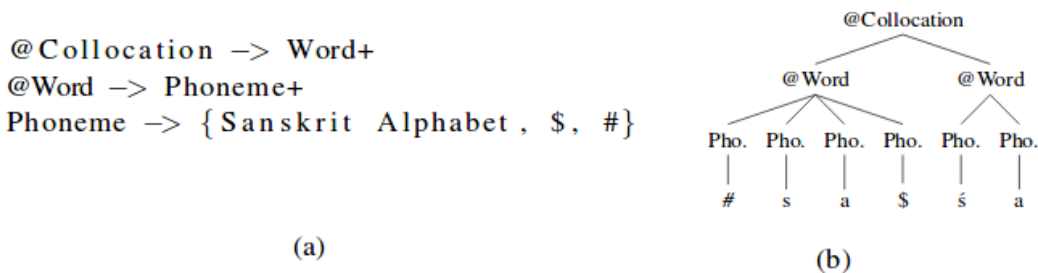


Figure 4: a) Skeletal grammar for the adaptor grammar (Johnson et al., 2007b). b) Derivation tree for an instance of a production '#sa\$śa' for the non-terminal @Collocation

244 Every production in the learned grammars has a probability to be invoked, where likelihood  
 245 of all the productions of a non-terminal sums to one. To obtain discriminative productions from  
 246 G1, G2 and G3, we find conditional entropy of the productions with that of G4 and filter only  
 247 those productions above a threshold. We also consider all the unique productions in each of  
 248 the Grammars in G1 to G3. We further restrict the productions based on the frequency of the  
 249 production in the data and the length of the sub-string produced by the production, both of  
 250 them were kept at the value of three.

251 We show an instance of one such production for a variable length character n-gram collocation.  
 252 Here, for the adapted non-terminal @Collocation, we find that one of the production finally  
 253 derives '#sa\$śa', which actually is derived as two @Word derivations as shown in the Figure  
 254 4b. We use this as a regular expression, which captures some properties that need to satisfied  
 255 by the concatenated components. The particular production mandates that the first component  
 256 must be exactly *sa*, as it is sandwiched between the symbols # and \$. Now, since *śa* occurs after  
 257 the previous substring which contains \$ the boundary for both the components, *śa* should belong  
 258 to the second component. Now, since as per the grammar both the substrings are independent  
 259 @word productions, we relax the constraint that both the substrings should occur immediately  
 260 one after the other. We treat the same as a regular expression, such that *śa* should occur after  
 261 *sa*, and any number of characters can come in between both the substrings. For this particular  
 262 pattern, we had 22 compounds, all of those belonging to *Bahuvrīhi*, which satisfied the criteria.  
 263 Now, compounds where first component is '*sa*' are mostly *Bahuvrīhi* compounds, and this  
 264 is obvious to Sanskrit linguists. But here, the system was not provided with any such prior  
 265 information or possible patterns. The system learnt the pattern from the data. Incidentally, our  
 266 dataset consisted of a few compound samples belonging to different classes as well where the  
 267 first component was '*sa*'.

Class	P	R	F
A	0.92	0.43	0.58
B	0.85	0.74	0.79
D	0.69	0.39	0.49
T	0.68	0.88	0.77

Table 1: Classwise performance of the Random Forests Classifier.

#### 268 4.1.1 Experiments

269 **Dataset** - We obtained a labelled dataset of compounds and the decomposed pairs of compo-  
270 nents from the Sanskrit studies department, UoHyd<sup>2</sup>. The dataset contains more than 32000  
271 unique compounds. The compounds were obtained from ancient digitised texts including *Śrī-*  
272 *mad Bhagavat Gīta*, *Caraka saṃhitā* among others. The dataset contains the *sandhi* split  
273 components along with the compounds. With more than 75 % of the dataset containing *Tat-*  
274 *puruṣa* compounds, we down-sample the *Tatpuruṣa* compounds to a count of 4000, to match  
275 with the second highest class, *Bahuvrīhi*. We find that the *Avyayībhāva* compounds are severely  
276 under-represented in the data-set, with about 5 % of the *Bahuvrīhi* class. From the dataset, we  
277 filtered 9952 different data-points split into 7957 data points for training and the remaining as  
278 held-out dataset.

279 **Result** - To measure the impact of different types of features we incorporated, we train the  
280 classifier incrementally with different feature types. We report the results over the held-out  
281 data. At first we train the system with only *Aṣṭādhyāyī* rules and some additional hand-crafted  
282 rules. We find that the overall accuracy of the system is about 59.34%. Then we augmented  
283 the classifier by adding features from *Amarakoṣa*. We find that the overall accuracy of the  
284 system has increased to 63.81%. We then finally add the adaptor grammar based features which  
285 has increased the performance of the system to an accuracy of 74.98 %. The effect of adding  
286 adaptor grammar features were more visible for the improvement in performance of *Dvandva*  
287 and *Bahuvrīhi*. Notably, the precision for *Dvandva* and *Bahuvrīhi* increased by absolute values  
288 0.15 and 0.06 respectively, when compared to the results before adding adaptor grammar based  
289 features. Table 1 presents the result of the system with the entire feature set per Compound class.  
290 The addition of adaptor grammar feature has resulted in an overall increase of the performance  
291 of the system from 63.81 % to 74.91 %. The patterns for adaptor grammar were learnt only  
292 using the data from training set and the heldout data was not used. This was done so as to  
293 ensure no over-fitting of data takes place. Also, we filtered the productions which are less than  
294 a length of 3 and does not occur many times in the grammar.

#### 295 4.2 Distinctive Patterns in Derivational Nouns in Taddhita<sup>3</sup>

296 Derivational nouns are a means of vocabulary expansion in a language. A new word is created  
297 in a language where an existing word is modified by an affix. Taddhita is a category of such  
298 derivational affixes which are used to derive a *prātipadika* from another *prātipadika*. The chal-  
299 lenge here is to identify Taddhita *prātipadikas* from corpora in Sanskrit and also to identify  
300 their source words.

301 Pattern based approaches often result in false positives. The edit distance between the source  
302 and derived words due to the patterns tends to vary from 1 to 6. For example, consider the  
303 word ‘*rāvāṇi*’ derived from ‘*rāvāṇa*’, where the edit distance between the words is just 1. But,  
304 ‘*Āśvalāyana*’ derived from ‘*aśvala*’ has an edit distance of 6. Also, the word ‘*kālaśa*’ is derived  
305 from the word ‘*kalaśa*’, but ‘*kāraṇa*’ is not derived from ‘*kaṛaṇa*’. Similarly ‘*stutya*’ is derived  
306 from ‘*stu*’ but using a *kṛt* affix. But, *dakṣiṇā* (South direction) is used to derive *dākṣiṇātya*

<sup>2</sup><http://sanskrit.uohyd.ac.in/scl/>

<sup>3</sup>The work has been done as part of the Derivational noun word pair identification work published in Krishna et al. (2017). The Authors wish to inform that there will be overlap in the content discussed here and the aforementioned work.



307 (Southern) with a taddhita affix. If we have to use vṛddhi as an indicator, which is the only  
308 difference between both the examples, then there are cases such as kāraka derived from kṛ  
309 for kṛt and aṣvaka is derived from aṣva using taddhita. All these instances show the level of  
310 ambiguity that can arise in deciding the pairs of source and derived words using taddhita. All  
311 the aforementioned examples show the need for knowledge of *Aṣṭādhyāyī* (or the knowledge of  
312 affixes), semantic relation between the word pairs or a combination of these to resolve the right  
313 set of word pairs.

314 The approach proposed in Krishna et al. (2017) first identifies a high recall low precision  
315 set of word pairs from multiple Sanskrit Corpora based on pattern similarities as exhibited by  
316 the 137 affixes in Taddhita. Once the patterns are obtained, we look for various similarities  
317 between the word pairs to group them together. We use rules from *Aṣṭādhyāyī* especially from  
318 Taddhita section. But since we could not incorporate rules of semantic and pragmatic nature, to  
319 compensate for the missing rules, we tried to identify patterns from the word pairs, specifically  
320 the source words, to be used. We use Adaptor Grammar for the purpose.

321 Currently, we do not identify the exact affix that leads to the derivation of the word. Also,  
322 since the affixes are distinguished not just by the visible pattern, but also by the ‘it’ markers,  
323 it is challenging to identify the exact affix. So, we group all those affixes that result in similar  
324 patterns into a single group. All the word pairs that follow the same pattern belongs to one  
325 group. To further increase the group size, we group all those entries that differ by vṛddhi and  
326 guṇa also into the same group. Such distinctions are not considered while forming a group.  
327 Effectively we only look into the pattern at the end of the ‘derived word’. We call all such  
328 collection of groups based on the patterns as our ‘candidate set’

329 For every distinct pattern in our candidate set, we first identify the word pairs and then create  
330 a graph with the given word pairs. A word pair is a node and edges are formed between nodes  
331 where they match different set of similarities. The first set of similarities are based on rules  
332 directly from *Aṣṭādhyāyī*, while the second set of node similarities were using character n-grams  
333 using Adaptor grammars. Once the similarities were found, we apply the Modified Adsorption  
334 approach (Talukdar and Crammer, 2009) on the graph. The modified adsorption is a semi  
335 supervised label prorogation approach where labels are provided to a subset of nodes and then  
336 propagated to the remaining nodes based on the similarity it shares with other nodes.

337 Figure 5 shows a sample construction of the graph for the word pairs, where words differ by a  
338 pattern ‘ya’. Here every pair obtained by pattern matching is a node. Now, Modified Adsorption  
339 is a semi supervised approach. So, we need limited number of labelled nodes. The nodes marked  
340 in grey are labelled nodes. They are called as seed nodes. The label here is just binary, i.e.  
341 a word pair can either be a true Taddhita pair or not. Now, edges are formed between the  
342 word pairs. Modified Adsorption provides a mean of designing the graph explicitly, while many  
343 of its predecessors relied more on nearest neighbour based approaches (Zhu and Ghahramani,  
344 2002). Also, the edges can be weighted based on the closeness between different nodes. Once  
345 the graph structure is defined, we perform the modified adsorption. in this approach, the labels  
346 from the seed nodes are propagated through the edges, such that the labels from seed nodes are  
347 propagated to other unlabelled nodes as well. The highly similar nodes should be given similar  
348 labels or else the optimisation function penalises any other label assignments. We use three  
349 different means of obtaining similarities between the nodes. The first such set of similarity is  
350 the rules in *Aṣṭādhyāyī* that the pair of nodes have a match with. The second set of similarity  
351 the sum of probabilities of productions from adaptor grammar, which are matched for a pair  
352 of nodes. The third is the word vector similarity between the source words in the node pairs.  
353 For a detailed working of system and a detailed explanation of each set of features please refer  
354 to Krishna et al. (2017). Here, we republish the working of the second set of features obtained  
355 using Adaptor grammar and the results of the model thereafter.

356 **Character n-grams similarity by Adaptor Grammar** - Pāṇini had an obligation to  
357 maintain brevity, as his grammar treatise was supposed to be memorised and recited orally by

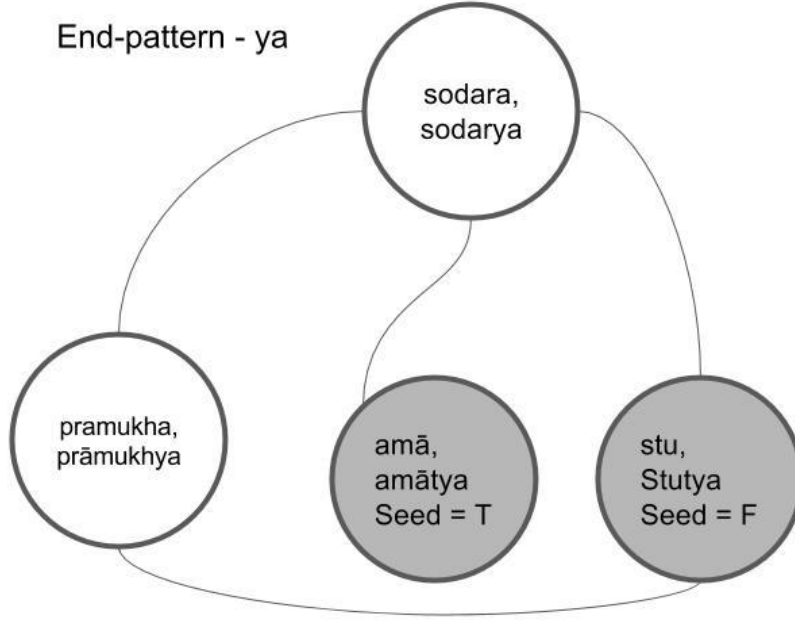


Figure 5: Graph structure for the group of words where derived words end in ‘ya’. Nodes in grey denote seed nodes, where they are marked with their class label. The Nodes in white are unlabelled nodes.

358 humans (Kiparsky, 1994). In *Aṣṭādhyāyī*, Pāṇini uses character sub-strings of varying lengths  
 359 as conditional rules for checking the suitability of application of an affix. We examine if there  
 360 are more such regularities in the form of variable length character n-grams that can be observed  
 361 from the data, as brevity is not a concern for us. Also, we assume this would compensate for  
 362 the loss of some of the information which Pāṇini originally encoded using pragmatic rules. In  
 363 order to identify the regularities in pattern in the words, we use Adaptor grammar.

364 In Listing 1, ‘Word’ and ‘Stem’ are non-terminals, which are adapted. The non-terminal  
 365 ‘Suffix’ consists of the set of various end-patterns. In this formalism, the grammar can only  
 366 capture sequential aspects in the words and hence attributes like *vṛddhi* that happen at the  
 367 internal of the word, non-sequential to rest of the modified pattern, need not be effectively  
 368 captured in the system.

369  $Word \rightarrow Stem Suffix$

370  $Word \rightarrow Stem$

371  $Stem \rightarrow Chars$

372  $Suffix \rightarrow a|ya|....|Ayana$

373 Listing 1: Skeletal CFG for the Adaptor grammar

374 The set  $\mathcal{A}_2$  captures all the variable length character n-grams learnt as the productions by  
 375 the grammar along with the probability score associated with the production. We form an edge  
 376 between two nodes in  $G_{i2}$ , if there exists an entry in  $\mathcal{A}_2$ , which are present in both the nodes.  
 377 We sum the probability value associated with all such character n-grams common to the pair  
 378 of nodes  $v_j, v_k \in V_i$ , and calculate the edge score  $\tau_{j,k}$ . If the edge score is greater than zero, we  
 379 find the sigmoid of the value so obtained to assign the weight to the edge. Equation ?? uses the  
 380 Iverson bracket (Knuth, 1992) to show the conditional sum operation. The equation essentially  
 381 makes sure that the probabilities associated with only those character n-grams gets summed,  
 382 which are present in both the nodes. We define the edge score  $\tau_{j,k}$ , weight set  $W_{i2}$  and Edge set

383  $E_{i2}$  as follows.

$$\tau_{j,k} = \sum_{l=1}^{|\mathcal{A}_2|} a_{k_2,l} [a_{k_2,l} = a_{j_2,l}]$$
$$E_{i2}^{v_k,v_j} = \begin{cases} 1 & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases}$$
$$W_{i2}^{v_k,v_j} = \begin{cases} \sigma(\tau_{j,k}) & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases}$$

384 As mentioned, we use the label distribution per node obtained from phase 1 as the seed labels  
385 in this setting.

#### 386 4.2.1 Experiments

387 As we mentioned, we use three different set of similarity sets for weighting the edges. But, in  
388 MAD we cannot provide different set of similarity functions together. While a weighted average  
389 of the similarities is one option, we chose to do a different approach altogether. We will apply  
390 the similarity weights sequentially on the graph. Here, we gain an additional advantage for this  
391 approach. In Modified Adsorption, we need to provide seed labels, which are labels for some of  
392 the nodes. In reality, the seed nodes do not have a binary assignment of the labels, rather a  
393 distribution of the labels (Talukdar and Crammer, 2009). So after the run of each similarity set,  
394 we get a label distribution for each of the node in the graph. This label distribution is used as a  
395 seed nodes in the subsequent run of the modified adsorption. The seed nodes also gets modified  
396 during the run of the algorithm.

397 **Dataset** - We use multiple lexicons and corpora to obtain our vocabulary  $\mathcal{C}$ . We use In-  
398 doWordNet (Kulkarni et al., 2010), the Digital Corpus of Sanskrit<sup>4</sup>, a digitised version of the  
399 Monier Williams<sup>5</sup> Sanskrit-English dictionary, a digitised version of the Apte Sanskrit-Sanskrit  
400 Dictionary (Goyal et al., 2012) and we also utilise the lexicon employed in the Sanskrit Heritage  
401 Engine (Goyal and Huet, 2016). We obtained close to 170,000 unique word lemmas from the  
402 combined resources.

403 **Results** - In Krishna et al. (2017), we report results from 11 of the patterns from a total  
404 of more than 80 patterns we initially obtained. The lack of evidence was the reason why we  
405 not attempted for others. here, we only show results for 5 of the patterns, which were selected  
406 based on the size of evidence from the corpora we obtain. Since we use each of the similarity  
407 set sequentially, we have outputs at each of the phase of the sequences. The result of the  
408 system after incorporating *Aṣṭādhyāyī* rules is *MADB1*, while that after incorporating Adaptor  
409 grammar ngrams is *MADB2* and the final result after the word vector similarity is *MAD*.  
410 Now, since we have 5 different patterns, we have an index  $i$  sub-scripted to the systems to  
411 denote the corresponding patterns. We additionally use a baseline called as Label Propagation  
412 (LP), based on the algorithm by Zhu and Ghahramani (2002). We can find that the systems  
413 which incorporates adaptor grammar are the *MAD* and *MADB2*. Both the systems are the  
414 best and second best performing systems respectively.

415 Table 2 shows the results for our system. We compare the performance of 5 different patterns,  
416 selected based on the number of candidate word pairs available for the pattern. The system  
417 proposed in the work *MAD<sub>i</sub>* performs the best for all the 5 patterns. Interestingly, *MADB2<sub>i</sub>*  
418 is the second best-performing system in all the cases. The system uses 3 kind of similarity  
419 measures in a sequential pipeline of which adaptor grammar comes as the second feature set. To  
420 understand the impact of adding adaptor grammar based features, we can compare the results  
421 with that of *MADB1<sub>i</sub>*. The system shows the result for each of the pattern before using adaptor  
422 grammar based features.

<sup>4</sup><http://kjc-sv013.kjc.uni-heidelberg.de/dcs/>

<sup>5</sup><http://www.sanskrit-lexicon.uni-koeln.de/monier/>

Pattern	System	P	R	A
a	MAD	0.72	0.77	73.86
	MADB2	0.68	0.68	68.18
	MADB1	0.49	0.52	48.86
	LP	0.55	0.59	55.68
aka	MAD	0.77	0.67	73.33
	MADB2	0.71	0.67	70
	MADB1	0.43	0.4	43.33
	LP	0.75	0.6	70
in	MAD	0.74	0.82	76.47
	MADB2	0.67	0.70	67.65
	MADB1	0.51	0.56	51.47
	LP	0.63	0.65	63.23
ya	MAD	0.7	0.72	70.31
	MADB2	0.61	0.62	60.94
	MADB1	0.53	0.59	53.12
	LP	0.56	0.63	56.25
i	MAD	0.55	0.52	54.76
	MADB2	0.44	0.38	45.24
	MADB1	0.3	0.29	30.95
	LP	0.37	0.33	38.09

Table 2: Comparative performance of the four competing models.

423 A baseline using the label propagation algorithm was also used. The motive behind the  
424 label propagation baseline was to measure the effect of Modified adsorption on the task.  
425 In Label Propagation, we experimented with the parameter  $K$  with different values,  $K \in$   
426  $\{10, 20, 30, 40, 50, 60\}$ , and found that  $K = 40$ , provides the best results for 3 of the 5 end-  
427 patterns. We find that for those 3 patterns ( $'a'$ ,  $'in'$ ,  $'i'$ ), the entire vertex set has  $v_{rddhi}$  attribute  
428 set to the same value. For the other two ( $'ya'$ ,  $'aka'$ ),  $K = 50$  gave the best results. Here, the  
429 vertex set has nodes where the  $v_{rddhi}$  attribute is set to either of the values. We report the best  
430 result for each of the system in Table 2.

## 431 5 Inference of Syntactic Structure in Sanskrit

432 In this section, we are reporting an ongoing work, where we investigate the effectiveness of using  
433 Adaptor grammar for inference of syntactic structures in Sanskrit. We experiment the effect of  
434 Adaptor Grammar in capturing the 'natural order' or the word order followed in prose. For this  
435 task, we use a dataset of Sanskrit sentences which are in prose order. The dataset consists of  
436 2000 sentences from Pañcākhyanāka and more than 600 sentences from Mahābhārata . For this  
437 experiment, we only consider the morphological classes of the words involved in the sentences.  
438 Currently we use the morphological tags as used in the Sanskrit Library<sup>6</sup>. We keep 500 of the  
439 sentences for testing and the remaining 2000 are used for identifying the patterns. Some of the  
440 constructs had one or two words, which we ignore for the experiment.

441 We learn the necessary productions in a grammar and then evaluate the grammar on the  
442 500 test sentences. We calculate the likelihood of generating each of the sentence. In order  
443 to test the likelihood of the correct sentence, we also generate all possible permutations of the  
444 morphological tags in each of the test sentences. For sentences of length  $> 5$ , we break them  
445 into sub-sequences of 5 and find the permutations of the sub-sequences and concatenate them  
446 again. This is used as a means of sampling the possible combinations as the explicit enumeration  
447 of all the permutations are computationally costly. From the generated candidate set we find  
448 the likelihood of the ground truth sentence and rank them. We report our results based on two  
449 measures.

- 450 1. **Edit Distance (ED)** - The edit distance of the top ranked sentence among the candidate  
451 set for a given sentence with that of the ground truth. Edit distance is roughly described  
452 as the minimum number of operations required to convert one string to another based on

<sup>6</sup><http://sanskritlibrary.org/helpmorphids.html>

453 a fixed set of operations with predefined costs. We use the standard Levenshtein distance,  
 454 where the three operations are ‘insert’, ‘delete’ and ‘substitution’. All the 3 operations have  
 455 a cost of 1. We compare the ground truth sentence with the predicted sentence that has the  
 456 highest likelihood to obtain the measure. The lesser the edit distance is better the result.

457 **2. Mean Reciprocal Rank (MRR)** - Mean Reciprocal rank is the average of reciprocal  
 458 ranks for each of the queries. Here a test sentence is treated as a query. The different  
 459 permutations are the retrieved results for the query. So from the ranked retrieved list, we  
 460 find the inverse of the rank of the gold standard sentence. The better the MRR Score,  
 461 better the result.

$$\frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{rel_i}{rank_i}$$

462 We first attempt the same skeletal grammars as proposed by Johnson et al. (2007b) for  
 463 capturing the syntactic regularities. We used both the ‘unigram’ and ‘collocation’ grammar as  
 464 mentioned in the work. Figures 6 and 7 show the first two grammars that we have used for the  
 465 task.

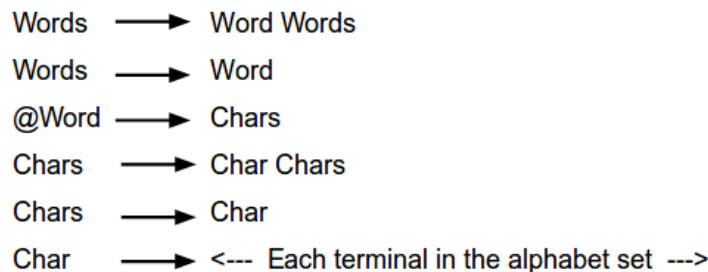


Figure 6: Unigram grammar as used in Johnson et al. (2007b)

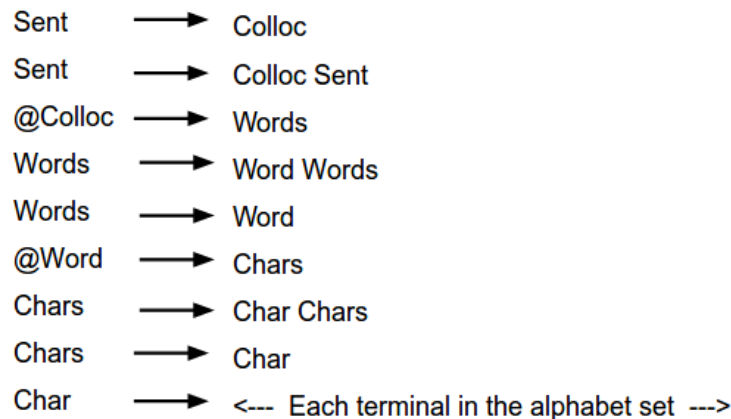


Figure 7: Collocation grammar as used in Johnson et al. (2007b)

466 With these grammars, we experimented with various hyper-parameter settings. Since both  
 467 the grammars are right recursive grammars, the length of the productions so learnt from the  
 468 grammar varied greatly. Though this is beneficial for identifying the word lengths, the associa-  
 469 tion with the morphological tags cannot be much longer. Secondly, the number of productions  
 470 to be learnt is a user defined hyper-parameter. We find that due to the possible varying length  
 471 size of strings and less number of observations, main morphological patterns that were learnt as  
 472 the productions were not repeated enough in the observations to be statistically significant.

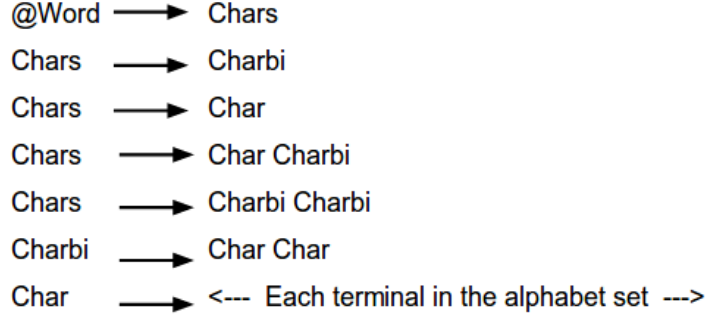


Figure 8: Modified grammar by eliminating the recursiveness in the Adapted nonterminal ‘@Word’.

473 We modified both the grammars to restrict the length of the productions to a maximum of  
 474 4 and limited the number of productions to be learnt. We show the modification done to the  
 475 adapted non-terminal ‘word’ in both the grammars. This restricts the number of productions  
 476 that ‘word’ can learn. The modified portion can be seen in Figure 8.

Grammar	MRR	ED
Unigram	0.2923	4.87
Collocation	0.3016	4.66
Modified Unigram	0.4025	3.21
Modified Collocation	0.5671	2.20

Table 3: Results for the word reordering task.

477 The results for all the four grammars are shown in Table 3. it can be seen that there is  
 478 considerable improvement in the Mean reciprocal rank and the edit distance measures for the  
 479 task with the restricted grammars. On our manual inspection of the patterns learnt from all the  
 480 grammars, it was observed that the initial skeletal grammars were essentially over-fitting the  
 481 training instances due to longer lengths. The modified grammars could reduce the Edit distance  
 482 to almost half and double the Mean Reciprocal Rank for the task.

483 For example, consider the sentence ‘tatra budhaḥ vrata caryā samāptau āgacchat (ā agacchat)’  
 484 from Mahābhārata. Consider the corresponding sequence of morphological tags as shown, ‘i  
 485 m1s iic f3s f7s i ipf[1]\_a3s’.<sup>7</sup> We filter out the ‘iic’ tags as the ‘iic’ tag stands for compound  
 486 component. It can be seen as part of the immediate next noun tag following it. We do not filter  
 487 out the ‘i’ tags as of now, where ‘i’, stands for the indeclinable. So in effect the tag sequence is  
 488 ‘i m1s f3s f7s i ipf[1]\_a3s’. The ‘Collocation’ Grammar had the following sequence as the most  
 489 likely output ‘i f7s i m1s f3s ipf[1]\_a3s’ with an edit distance of 4. In the ‘Modified Collocation’  
 490 Grammar the predicted sequence is ‘i m1s f3s i f7s ipf[1]\_a3s’. The edit distance of the sentence  
 491 is 2. Here, it can be seen that just 2 tags have swapped their position. The tags ‘i’ and ‘f7s’  
 492 have changed their positions, but are still at adjacent positions to each other. The fourth and  
 493 fifth words in the original sentence have changed to become the fifth and fourth words in the  
 494 predicted sentence.

495 The results shown here are preliminary in nature. What excites us the most is the provision  
 496 this framework provides to incorporate the syntactic knowledge which is explicitly defined in our  
 497 grammar formalisms. With this work, we plan to extend the work to two immediate tasks. First,  
 498 we plan to extend the word-reordering task to the poetry to prose conversion task. Currently, the  
 499 task is to convert a bag of words into its corresponding prose or the ‘natural order’. But we will  
 500 investigate the regularities involved in poetry apart from the aspects of meter and incorporate  
 501 the regularities to guide the grammar in picking up those patterns. We can also attempt to learn

<sup>7</sup>We follow the notations from Sanskrit Library - <http://sanskritlibrary.org/helpmorphids.html>

502 the conditional probabilities for the syntactic patterns in both poetry and prose. Second, we  
503 will be performing the Dependency parse analysis of given sentences at a morphological level. A  
504 dependency analysis of a sentence using Context free grammar, i.e., phrase structured grammars  
505 are not straightforward. Headden III et al. (2009) provide some insights into use of PCFGs and  
506 lexical evidence for unsupervised dependency parsing. Currently we will be working only on the  
507 projective dependency parsing. We will be relying on the Dependency Model with Valence to  
508 define our PCFG formalism for dependency parsing.

## 509 6 Conclusion

510 The primary goal of this work was to look into the applicability of the Adaptor Grammars, a  
511 non-parametric Bayesian approach for learning syntactic structures from observations. In this  
512 work, we introduced the basic concepts of the Adaptor grammars, various applications in which  
513 the grammar is used in NLP tasks. We provide detailed descriptions of how adaptor grammar is  
514 used in word level vocabulary expansion tasks in Sanskrit. The adaptor grammars were used as  
515 effective sub-word n-gram features for both Compound type identification and Derivational noun  
516 pair identification. We further showed the feasibility of using adaptor grammar for syntactic  
517 level analysis of sentences in Sanskrit. We will be further investigating the feasibility of using  
518 the Adaptor grammars for dependency parsing and poetry to prose conversion tasks at sentence  
519 level.

## 520 Acknowledgements

521 The authors acknowledge use of the morphologically tagged database of the Pañcākhyanaka  
522 and Mahābhārata produced under the direction of Professor Peter M. Scharf while laureate of  
523 a Blaise Pascal Research Chair at the Université Paris Diderot 2012–2013 and maintained by  
524 The Sanskrit Library.

## 525 References

- 526 Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non- concatenative morphology. In  
527 *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association  
528 for Computational Linguistics.
- 529 Stephen Cass, 2011. *Unthinking Machines, Artificial intelligence needs a reboot, say experts*.
- 530 Shay B Cohen, David M Blei, and Noah A Smith. 2010. Variational inference for adaptor grammars. In  
531 *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the*  
532 *Association for Computational Linguistics*, pages 564–572. Association for Computational Linguistics.
- 533 Ramy Eskander, Owen Rambow, and Tianchun Yang. 2016. Extending the use of adaptor grammars for  
534 unsupervised morphological segmentation of unseen languages. In *COLING*, pages 900–910.
- 535 Naomi H Feldman, Thomas L Griffiths, Sharon Goldwater, and James L Morgan. 2013. A role for the  
536 developing lexicon in phonetic category acquisition. *Psychological review*, 120(4):751.
- 537 Samuel J Gershman and David M Blei. 2012. A tutorial on bayesian nonparametric models. *Journal of*  
538 *Mathematical Psychology*, 56(1):1–12.
- 539 Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus anno-  
540 tation. *Journal of Language Modelling*, 4(2):145–182.
- 541 Pawan Goyal, Gérard P Huet, Amba P Kulkarni, Peter M Scharf, and Ralph Bunker. 2012. A distributed  
542 platform for sanskrit processing. In *COLING*, pages 1011–1028.
- 543 Eric A Hardisty, Jordan Boyd-Graber, and Philip Resnik. 2010. Modeling perspective using adaptor  
544 grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Process-*  
545 *ing*, pages 284–292. Association for Computational Linguistics.

- 546 William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised depen-  
547 dency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies:  
548 The 2009 Annual Conference of the North American Chapter of the Association for Computational  
549 Linguistics*, pages 101–109. Association for Computational Linguistics.
- 550 James Jay Horning. 1969. A study of grammatical inference. Technical report, STANFORD UNIV  
551 CALIF DEPT OF COMPUTER SCIENCE.
- 552 Yun Huang, Min Zhang, and Chew Lim Tan. 2011. Nonparametric bayesian machine transliteration  
553 with synchronous adaptor grammars. In *Proceedings of the 49th Annual Meeting of the Association  
554 for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 534–539.  
555 Association for Computational Linguistics.
- 556 Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic chinese word segmentation using  
557 adaptor grammars. In *Proceedings of the 23rd international conference on computational linguistics*,  
558 pages 528–536. Association for Computational Linguistics.
- 559 Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for pcfgs via markov  
560 chain monte carlo. In *Human Language Technologies 2007: The Conference of the North American  
561 Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages  
562 139–146.
- 563 Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007b. Adaptor grammars: A framework for  
564 specifying compositional nonparametric bayesian models. In *Advances in neural information processing  
565 systems*, pages 641–648.
- 566 Mark Johnson. 2008a. Unsupervised word segmentation for sesotho using adaptor grammars. In *Proceed-  
567 ings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*,  
568 pages 20–27. Association for Computational Linguistics.
- 569 Mark Johnson. 2008b. Using adaptor grammars to identify synergies in the unsupervised acquisition of  
570 linguistic structure. In *ACL*, pages 398–406.
- 571 Mark Johnson. 2010. Pcfgs, topic models, adaptor grammars and learning topical collocations and  
572 the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for  
573 Computational Linguistics*, pages 1148–1157. Association for Computational Linguistics.
- 574 Paul Kiparsky. 1994. Paninian linguistics. *The Encyclopedia of Language and Linguistics*, 6:2918–2923.
- 575 Donald E Knuth. 1992. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422.
- 576 Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016.  
577 Compound type identification in sanskrit: What roles do the corpus and grammar play? In *Proceedings  
578 of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*,  
579 pages 1–10, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- 580 Amrith Krishna, Pavankumar Satuluri, Harshavardhan Ponnada, Muneeb Ahmed, Gulab Arora, Kaus-  
581 tubh Hiware, and Pawan Goyal. 2017. A graph based semi-supervised approach for analysis of deriva-  
582 tional nouns in sanskrit. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for  
583 Natural Language Processing*, pages 66–75, Vancouver, Canada, August. Association for Computational  
584 Linguistics.
- 585 Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya.  
586 2010. Introducing sanskrit wordnet. In *Proceedings on the 5th Global Wordnet Conference (GWC  
587 2010)*, *Narosa, Mumbai*, pages 287–294.
- 588 Arun Kumar, Lluís Padró, and Antoni Oliver González. 2015. Joint bayesian morphology learning of  
589 dravidian languages. In *RICTA 2015: Proceedings of the Joint Workshop on Language Technology for  
590 Closely Related Languages, Varieties and Dialects: Hissan, Bulgaria: September 10, 2015: proceedings  
591 book*.
- 592 Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. 2007. The infinite pcfg using hierarchical  
593 dirichlet processes. In *EMNLP-CoNLL*, pages 688–697.
- 594 Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.



- 595 Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An  
596 unsupervised model for joint phrase alignment and extraction. In *Proceedings of the 49th Annual*  
597 *Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*,  
598 pages 632–641. Association for Computational Linguistics.
- 599 Peter Norvig, 2011. *On Chomsky and the Two Cultures of Statistical Learning*.
- 600 Timothy J O’Donnell. 2015. *Productivity and reuse in language: A theory of linguistic computation and*  
601 *storage*. MIT Press.
- 602 Steven Pinker, 2011. *Keynote Panel: The Golden Age — A Look at the Original Roots of Artificial*  
603 *Intelligence, Cognitive Science, and Neuroscience*.
- 604 Alan Prince and Paul Smolensky. 1993. *Optimality Theory: Constraint interaction in generative gram-*  
605 *mar*. John Wiley & Sons, the version published in 2008.
- 606 Paul Smolensky and Géraldine Legendre. 2006. *The harmonic mind: From neural computation to*  
607 *optimality-theoretic grammar (Cognitive architecture), Vol. 1*. MIT press.
- 608 Partha Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning.  
609 *Machine Learning and Knowledge Discovery in Databases*, pages 442–457.
- 610 Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. 2012. Exploring adaptor grammars for native  
611 language identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural*  
612 *Language Processing and Computational Natural Language Learning*, pages 699–709.
- 613 Ke Zhai, Jordan Boyd-Graber, and Shay B Cohen. 2014. Online adaptor grammars with hybrid inference.  
614 *Transactions of the Association for Computational Linguistics*, 2:465–476.
- 615 Ke Zhai, Zornitsa Kozareva, Yuening Hu, Qi Li, and Weiwei Guo. 2016. Query to knowledge: Unsu-  
616 pervised entity extraction from shopping queries using adaptor grammars. In *Proceedings of the 39th*  
617 *International ACM SIGIR conference on Research and Development in Information Retrieval*, pages  
618 255–264. ACM.
- 619 Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label  
620 propagation.