An 'Ekalavya' Approach to Learning Context Free Grammar Rules for Sanskrit Using Adaptor Grammar

Amrith Krishna, Bodhisattwa Prasad Majumder*, Anil Kumar Boga\$, Pawan Goyal
Dept. of Computer Science and Engineering, IIT Kharagpur, India
*Walmart Labs, India
\$Dept of Electronics and Electrical Communication Engineering, IIT Kharagpur, India
amrith@iitkgp.ac.in, bodhisattwapm2017@email.iimcal.ac.in, bogaanil.009@gmail.com, pawang@cse.iitkgp.ernet.in

Abstract

This work presents the use of Adaptor Grammar, a non-parametric Bayesian approach for learning (Probabilistic) Context Free Grammar productions from data. In Adaptor Grammar, we provide the set of non-terminals followed by a skeletal grammar that establishes the relations between the non-terminals in the grammar. The productions and the associated probability for the productions are automatically learnt by the system from the usages of words or sentences, i.e., the dataset. This facilitates the encoding of prior linguistic knowledge through the skeletal grammar and yet the tiresome task of finding the productions is delegated to the system. As the system completely learns the grammar structure by observing the data. We call this approach as the 'Ekalavya' approach. In this work, we discuss the effect of using Adaptor grammars for Sanskrit at word-level supervised tasks such as compound type identification and also in identifying source and derived words from corpora for derivational nouns. In both of the works, we show the use of sub-word patterns learnt using Adaptor grammar as effective features for their corresponding supervised tasks. We also present our novel approach of using Adaptor Grammars for handling Structured Prediction tasks in Sanskrit. We present the preliminary results for word reordering task in Sanskrit. We also outline our plan for the use of Adaptor grammars for Dependency Parsing and Poetry to Prose Conversion tasks.

1 Introduction

The recent trends in Natural Language Processing (NLP) community suggest an increased application of black-box statistical approaches such as deep learning. In fact, such systems are preferred as there has been increase in performance of several NLP tasks such as machine translation, sentiment analysis, word sense disambiguation etc. (Manning, 2016). In fact, MIT Technology review reported the following regarding Noam Chomsky's opinion about the extensive use of 'purely statistical methods' in AI. The report says that "derided researchers in machine learning who use purely statistical methods to produce behaviour that mimics something in the world, but who don't try to understand the meaning of that behaviour." (Cass, 2011).

Chomsky quotes, "It's true there's been a lot of work on trying to apply statistical models to various linguistic problems. I think there have been some successes, but a lot of failures. There is a notion of success ... which I think is novel in the history of science. It interprets success as approximating un-analysed data." (Pinker et al., 2011). Norvig (2011), in his reply to Chomsky comes in defence of statistical approaches used in the community. Norvig lays emphasis on the engineering aspects of the problems that the community deals with and the performance gains achieved in using such approaches. He rightly attributes that, while the generative aspects of a language can be deterministic, the analysis of a language construct can lead to ambiguity. As probabilistic models are tolerant to noise in the data, the use of such approaches is often necessary for engineering success. It is often the case that the speakers of a language deviates from the laid out linguistic rules in usage. This can be seen as noise in the dataset, and yet the system we intend to build should be tolerant to such issues as well. The use of statistical approaches provides a convenient means of achieving the same. But, the use of statistical approaches do not imply discarding of the linguistic knowledge that we possess. Manning (2016) quotes the work of Paul Smolensky, "Work by Paul Smolensky on how basically categorical systems can emerge and be represented in a neural substrate (Smolensky and Legendre, 2006). Indeed, Paul Smolensky arguably went too far down the rabbit hole, devoting a large part of his career to developing a new categorical model of phonology, Optimality Theory (Prince and Smolensky, 1993)." This is an example where the linguistics and the statistical computational models had a successful synergy, fruitful for both the domains.

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

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

1. Adaptor grammars in effect output valid PCFGs, which in turn are context free grammars, and thus are valid for linguistic representations.

2. It helps to encode linguistic information which is already described in various formalisms via the skeletal grammars. Thus domain knowledge can effectively be used. The only restriction here might be that the expressive power of the grammar is limited to that of a Context Free Grammar.

3. By leveraging the power of statistics, we can obtain the likelihood of various possible parses, in case of structural ambiguity during analysis of a sentence.

4. While the proposed structures might not be as competitive in performance as with the black-box statistical approaches such as the deep learning approaches, the interpretability of the Adaptor grammar based systems is a big plus. Grammar experts can look into the individual production rules learnt by the system. This frees the experts from coming up with the rules in the first place. Additionally by looking into the production rules, understandable to any domain expert with the knowledge of context free grammars, it can be validated whether the system has learnt patterns that are relevant to the task or not.

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

2 Preliminaries - CFG and Probabilistic CFG

Context Free Grammar was proposed by Noam Chomsky who initially termed it as phrase structure grammar. Formally, a Context Free Grammar $\mathcal{G}$ is a 4-tuple (V, Σ, R, S), where V is a set of non-terminals, Σ is a finite set of terminals, R is the set of productions from V to $(V \cup \Sigma)^*$, where $*$ is the 'Kleene Star' operation. S is an element of V which is treated as the start symbol, which forms the root of the parse trees for every string accepted by the grammar.

The language that can be generated by the Non-terminal X can be represented as $\mathcal{L}_X$. So, the language that can be generated by the grammar $\mathcal{G}$ is $\mathcal{L}_S$.

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}
Σ = {Word1', Word2', 'Word3', 'Word4, Word5'}

Figure 1: An example of a Context Free Grammar

The productions in Context Free Grammars are often handcrafted by linguistic experts. it is common to have large CFGs for many of the real life NLP tasks. It is common that a given string can have multiple possible parses for the given grammar. This is due to the fact that a Context Free Grammar contains all possible choices that can be produced from a given Non-terminal (O'Donnell, 2015). The grammar neither provide a deterministic parse nor prioritises the parses. This leads to structural ambiguity in the grammar. Probabilistic Context Free Grammars (PCFGs) have been introduced to weigh the probable trees when the ambiguity arises, and thus provide a means for prioritising the desired rules. A PCFG is a 5-tuple $(V, \Sigma, R, S, \theta)$, where θ, denotes a vector of real numbers in the range of $[0, 1]$ indexed by productions of R, subject to

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

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}
Σ = {Word1', Word2', 'Word3', 'Word4, Word5'}

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

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

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

1. The decision about expanding a non-terminal depends only on the non-terminal and the given distribution for that non-terminal. No other assumptions can be made.

2. Following from the first assumption, a generated expression is independent of other expressions.

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

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

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

where,

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

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

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

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

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

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

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

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 another 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

S → HB | BC
H → BC
@B → < -- Productions to be learnt by Adaptor -->
@C → < -- Productions to be learnt by Adaptor -->

V = {S,H,B,C}
Σ = {Alphabet Set}

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

$TD_X(G_{Y_1}, G_{Y_2} ....., G_{Y_m})$ is a distribution over all the trees rooted in the non-terminal X

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

$$G_X \sim C_X(H_X)$$

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 also demonstrates the effect of *syllabification* on word segmentation task using PCFGs (Johnson, 2008b). Johnson further motivates the usability of the aforementioned unsupervised approaches for word segmentation and 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 (Blei et al., 2017) 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 can be thought of as an adaptor grammar over two languages. Zhai et al. (2016) has presented a work, where the authors attempted to identify relevant suggestive keywords to a typed query so as to improve the results for search in an e-commerce site. The authors previously presented a new variational inference approach through a hybrid of Markov chain Monte Carlo and variational inference. It has been reported that the hybrid scheme has improved scalability without compromising the performance on typical common tasks of grammar induction.

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

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

As an extension to the standard Adaptor Grammar, O'Donnell (2015) presented Fragment Grammars which was built as a generalization of Adaptor Grammars. They generalise Adaptor Grammars by scoping the productivity and abstraction to occur at any points within individual stored structures. The specific model has adopted 'stochastic memoization' as an efficient substructure storing mechanism from the Adaptor grammar framework. It further memoizes partial internal computations via lazy evaluation version of the original storage mechanism given by Adaptor Grammar.

4 Adaptor Grammar for Sanskrit

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

4.1 Variable Length Character n-grams for compound type identification[1]

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

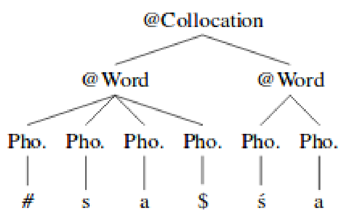
The system is developed as an ensemble based supervised classifier. We used Random Forests classifier with easy ensemble approach to handle the class imbalance problem persisting in the data. The classifier had majority of its labels in *Tatpuruṣa*. The presence of *Avyayībhāva* was the least. The classifier incorporated rich features from multiple sources. The rules from *Aṣṭādhyāyī* pertaining to compounds which are of conditional nature i.e. contains those containing selectional constraints were encoded as a feature. This was encoded by applying those selectional restrictions over the input compounds. Variable length character n-grams for each class of compounds were obtained from adaptor grammar. Each filtered production from the compound class specific grammar was used as a feature. We also incorporated noun pairs that follows the knoweldge structure in Amarakośa as mentioned in Nair and Kulkarni (2010). We used selected subset of relations from Nair and Kulkarni (2010).

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

We learn 3 separate grammars namely, G1, G2 and G3, with the same skeletal structure in Figure 4a, but with different data samples belonging to *Tatpuruṣa*, *Bahuvrīhi* and *Dvandva* respectively. We did not learn a grammar for *Avyayībhāva*, due to insufficient data samples for learning the patterns. We use a '$' marker to indicate the word boundary between the components, where the components were in sandhi split form. A '#' symbol was added to mark the beginning and ending of the first and the final components, respectively. We also learn a grammar G4, where the entire dataset is taken together along with additional 4000 random pair of words from the Digital Corpus of Sanskrit, where none of the words appeared as a compound component in the corpus. The co-occurrence or the absence of it was taken as the proxy for compatibility between the components. The skeletal grammar in Figure 4b has two adapted non-terminals, both marked by '@'. Also, the adapted non-terminal 'Word' is a non-terminal appearing as a production to the adapted non-terminal 'Collocation'. The '+' symbol indicates the notion of one one or more occurrence of 'Word', as used in regular expressions. This is not standard to use the notation in productions as per context free grammar. This is ideally achieved using recursive grammars in CFGs with additional non-terminals. But, in order to present a simpler representation of skeletal grammar we followed this scheme. In subsequent representations we will be using recursiveness instead of the '+' notation.

```
@Collocation -> Word+
@Word -> Phoneme+
Phoneme -> {Sanskrit Alphabet , $, #}
```

(a)



(b)

Figure 4: a) Skeletal grammar for the adaptor grammar b) Derivation tree for an instance of a production '#sa$ śa' for the non-terminal @Collocation

[1]The work has been done as part of the compound type identification work published in Krishna et al. (2016). Please refer to the aforementioned work for a detailed explanation of the concepts described here.

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

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

4.1.1 Experiments

Dataset - We obtained a labelled dataset of compounds and the decomposed pairs of components from the Sanskrit studies department, UoHyd[2]. The dataset contains more than 32000 unique compounds. The compounds were obtained from ancient digitised texts including *Śrīmad Bhagavat Gīta*, *Caraka saṃhitā* among others. The dataset contains the *sandhi* split components along with the compounds. With more than 75 % of the dataset containing *Tatpuruṣa* compounds, we down-sample the *Tatpuruṣa* compounds to a count of 4000, to match with the second highest class, *Bahuvrīhi*. We find that the *Avyayībhāva* compounds are severely under-represented in the data-set, with about 5 % of the *Bahuvrīhi* class. From the dataset, we filtered 9952 different data-points split into 7957 data points for training and the remaining as held-out dataset.

Result - To measure the impact of different types of features we incorporated, we train the classifier incrementally with different feature types. We report the results over the held-out data. At first we train the system with only *Aṣṭādhyāyī* rules and some additional hand-crafted rules. We find that the overall accuracy of the system is about 59.34%. Then we augmented the classifier by adding features from *Amarakoṣa*. We find that the overall accuracy of the system has increased to 63.81%. We then finally add the adaptor grammar based features which has increased the performance of the system to an accuracy of 74.98 %. The effect of adding adaptor grammar features were more visible for the improvement in performance of *Dvandva* and *Bahuvrīhi*. Notably, the precision for *Dvandva* and *Bahuvrīhi* increased by absolute values 0.15 and 0.06 respectively, when compared to the results before adding adaptor grammar based features. Table 1 presents the result of the system with the entire feature set per Compound class. The addition of adaptor grammar feature has resulted in an overall increase of the performance of the system from 63.81 % to 74.91 %. The patterns for adaptor grammar were learnt only using the data from training set and the heldout data was not used. This was done so as to

[2]http://sanskrit.uohyd.ac.in/scl/

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.

ensure no over-fitting of data takes place. Also, we filtered the productions which are less than a length of 3 and does not occur many times in the grammar.

4.2 Distinctive Patterns in Derivational Nouns in Taddhita[3]

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

Pattern based approaches often result in false positives. The edit distance, a popular distance metric to compare the characterise similarity of two given strings, between the source and derived words due to the patterns tends to vary from 1 to 6. For example, consider the word '*rāvaṇi*' derived from '*rāvaṇa*', where the edit distance between the words is just 1. But, '*Āśvalāyana*' derived from '*aśvala*' has an edit distance of 6. Also, the word 'kālaśa' is derived from the word 'kalaśa', but 'kāraṇa' is not derived from 'karaṇa'. Similarly 'stutya' is derived from 'stu' but using a kṛt affix. But, dakṣiṇā (South direction) is used to derive dākṣhiṇātya (Southern) with a taddhita affix. If we have to use vṛddhi as an indicator, which is the only difference between both the examples, then there are cases such as kāraka derived from kṛ for kṛt and aṣvaka is derived from aṣva using taddhita. All these instances show the level of ambiguity that can arise in deciding the pairs of source and derived words using taddhita. All the aforementioned examples show the need for knowledge of *Aṣṭādhyāyī*(or the knowledge of affixes), semantic relation between the word pairs or a combination of these to resolve the right set of word pairs.

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

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

For every distinct pattern in our candidate set, we first identify the word pairs and then create a graph with the given word pairs. A word pair is a node and edges are formed between nodes where they match different set of similarities. The first set of similarities are based on rules directly from *Aṣṭādhyāyī*, while the second set of node similarities were using character n-grams

[3]The work has been done as part of the Derivational noun word pair identification work published in Krishna et al. (2017). Please refer to the aforementioned work for a detailed explanation of the concepts described here.

using Adaptor grammars. Once the similarities were found, we apply the Modified Adsorption approach (Talukdar and Crammer, 2009) on the graph. The modified adsorption is a semi supervised label prorogation approach where labels are provided to a subset of nodes and then propagated to the remaining nodes based on the similarity it shares with other nodes.

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

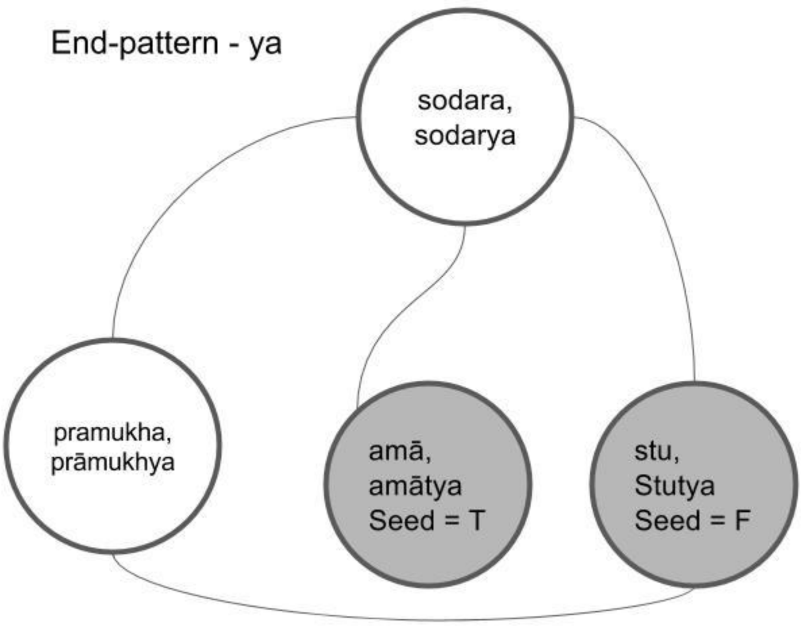


Figure 5: Graph structure for the group of words where derived words end in 'ya'. Nodes in grey denote seed nodes, where they are marked with their class label. The Nodes in white are unlabelled nodes.

Character n-grams similarity by Adaptor Grammar - Pāṇini had an obligation to maintain brevity, as his grammar treatise was supposed to be memorised and recited orally by humans (Kiparsky, 1994). In *Aṣṭādhyāyī*, Pāṇini uses character sub-strings of varying lengths as conditional rules for checking the suitability of application of an affix. We examine if there are more such regularities in the form of variable length character n-grams that can be observed from the data, as brevity is not a concern for us. Also, we assume this would compensate for the loss of some of the information which Pāṇini originally encoded using pragmatic rules. In order to identify the regularities in pattern in the words, we use Adaptor grammar.

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

$Word \rightarrow Stem\ Suffix$
$Word \rightarrow Stem$
$Stem \rightarrow Chars$
$Suffix \rightarrow a|ya|.....|Ayana$

Listing 1: Skeletal CFG for the Adaptor grammar

The set $\mathcal{A}_2$ captures all the variable length character n-grams learnt as the productions by the grammar along with the probability score associated with the production. We form an edge between two nodes in G_{i2}, if there exists an entry in $\mathcal{A}_2$, which are present in both the nodes. We sum the probability value associated with all such character n-grams common to the pair 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 find the sigmoid of the value so obtained to assign the weight to the edge. The expression for calculating $\tau_{j,k}$ in the equation given below uses the Iverson bracket (Knuth, 1992) to show the conditional sum operation. The equation essentially makes sure that the probabilities associated with only those character n-grams gets summed, which are present in both the nodes. We define the edge score $\tau_{j,k}$, weight set W_{i2} and Edge set E_{i2} as follows.

$$\tau_{j,k} = \sum_{l=1}^{|\mathcal{A}_2|} a_{k_2,l}[a_{k_2,l} = a_{j_2,l}]$$

$$E_{i2}^{v_k,v_j} = \begin{cases} 1 & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases}$$

$$W_{i2}^{v_k,v_j} = \begin{cases} \sigma(\tau_{j,k}) & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases}$$

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

4.2.1 Experiments

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

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

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

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.

Table 2 shows the results for our system. We compare the performance of 5 different patterns, selected based on the number of candidate word pairs available for the pattern. The system proposed in the work MAD_i performs the best for all the 5 patterns. Interestingly, $MADB2_i$ is the second best-performing system in all the cases. The system uses 3 kind of similarity measures in a sequential pipeline of which adaptor grammar comes as the second feature set. To understand the impact of adding adaptor grammar based features, we can compare the results with that of $MADB1_i$. The system shows the result for each of the pattern before using adaptor grammar based features.

A baseline using the label propagation algorithm was also used. The motive behind the label propagation baseline was to measure the effect of Modified adsorption on the task. In Label Propagation, we experimented with the parameter K with different values, $K \in \{10, 20, 30, 40, 50, 60\}$, and found that $K = 40$, provides the best results for 3 of the 5 end-patterns. The values for K are set by empirical observations. We find that for those 3 patterns

[4]http://kjc-sv013.kjc.uni-heidelberg.de/dcs/

[5]http://www.sanskrit-lexicon.uni-koeln.de/monier/

(‘a’,‘*in*’,‘*i*’), the entire vertex set has *vṛddhi* attribute set to the same value. For the other two (‘*ya*’,‘*aka*’), $K = 50$ gave the best results. Here, the vertex set has nodes where the *vṛddhi* attribute is set to either of the values. For a better insight towards this finding, the notion of the pattern that we use in the design of the system needs be elaborated. A pattern is effectively the substrings that remain in both the source word and derived word after removing the portions which are common in both. This pattern is the visible change that happens in the derivation of a word. To reduce the number of distinct patterns we did not consider the pattern changes that occur due to *vṛddhi* and *guṇa* as distinct patterns, rather we abstracted them out. Now, multiple affixes may lead to generation of the same set of patterns. In the case of pattern, rather end-pattern, (Krishna et al., 2017), ‘a’, the effect may be the result of application of one of the following affixes such as *aṇ añ* etc. Here, all the affixes of pattern ‘a’ leads to *vṛddhi*. But for the pattern ‘ya’, the affixes may or may not lead to a *vṛddhi*. We report the best result for each of the system in Table 2.

5 Inference of Syntactic Structure in Sanskrit

In this section, we are reporting an ongoing work, where we investigate the effectiveness of using Adaptor grammar for inference of syntactic structures in Sanskrit. We experiment the effect of Adaptor Grammar in capturing the ‘natural order’ or the word order followed in prose. For this task, we use a dataset of Sanskrit sentences which are in prose order. The dataset consists of 2000 sentences from Pañcākhyānaka and more than 600 sentences from Mahābhārata . For this experiment, we only consider the morphological classes of the words involved in the sentences. Currently we use the morphological tags as used in the Sanskrit Library[6]. We keep 500 of the sentences for testing and the remaining 2000 are used for identifying the patterns. Some of the constructs had one or two words, which we ignore for the experiment.

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

1. **Edit Distance (ED)** - The edit distance of the top ranked sentence among the candidate set for a given sentence with that of the ground truth. Edit distance is roughly described as the minimum number of operations required to convert one string to another based on a fixed set of operations with predefined costs. We use the standard Levenshtein distance (Levenshtein, 1966), where the three operations are ‘insert’, ‘delete’ and ‘substitution’. All the 3 operations have a cost of 1. We compare the ground truth sentence with the predicted sentence that has the highest likelihood to obtain the measure. The lesser the edit distance is better the result.

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

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

[6] http://sanskritlibrary.org/helpmorphids.html

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

```
Words  → Word Words
Words  → Word
@Word  → Chars
Chars  → Char Chars
Chars  → Char
Char   → <--- Each terminal in the alphabet set --->
```

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

```
Sent    → Colloc
Sent    → Colloc Sent
@Colloc → Words
Words   → Word Words
Words   → Word
@Word   → Chars
Chars   → Char Chars
Chars   → Char
Char    → <--- Each terminal in the alphabet set --->
```

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

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

```
@Word  → Chars
Chars  → Charbi
Chars  → Char
Chars  → Char Charbi
Chars  → Charbi Charbi
Charbi → Char Char
Char   → <--- Each terminal in the alphabet set --->
```

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

We modified both the grammars to restrict the length of the productions to a maximum of

4 and limited the number of productions to be learnt. We show the modification done to the adapted non-terminal ‘word’ in both the grammars. This restricts the number of productions 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.

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

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

The results shown here are preliminary in nature. What excites us the most is the provision this framework provides to incorporate the syntactic knowledge which is explicitly defined in our grammar formalisms. With this work, we plan to extend the work to two immediate tasks. First, we plan to extend the word-reordering task to the poetry to prose conversion task. Currently, the task is to convert a bag of words into its corresponding prose or the ‘natural order’. But we will investigate the regularities involved in poetry apart from the aspects of meter and incorporate the regularities to guide the grammar in picking up those patterns. We can also attempt to learn the conditional probabilities for the syntactic patterns in both poetry and prose. Second, we will be performing the Dependency parse analysis of given sentences at a morphological level. A dependency analysis of a sentence using Context free grammar, i.e., phrase structured grammars are not straightforward. Goyal and Kulkarni (2014) presents a scheme for converting Sanskrit constructs in constituency parse structure to Dependency parse structure. Headden III et al. (2009) provide some insights into use of PCFGs and lexical evidence for unsupervised dependency parsing. Currently we will be working only on the projective dependency parsing. We will be relying on the Dependency Model with Valence to define our PCFG formalism for dependency parsing.

6 Conclusion

The primary goal of this work was to look into the applicability of the Adaptor Grammars, a non-parametric Bayesian approach for learning syntactic structures from observations. In this work, we introduced the basic concepts of the Adaptor grammars, various applications in which

[7]We follow the notations from Sanskrit Library - http://sanskritlibrary.org/helpmorphids.html

the grammar is used in NLP tasks. We provide detailed descriptions of how adaptor grammar is used in word level vocabulary expansion tasks in Sanskrit. The adaptor grammars were used as effective sub-word n-gram features for both Compound type identification and Derivational noun pair identification. We further showed the feasibility of using adaptor grammar for syntactic level analysis of sentences in Sanskrit. We plan to investigate the feasibility of using the Adaptor grammars for dependency parsing and poetry to prose conversion tasks at sentence level.

Acknowledgements

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

References

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted).

Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non- concatenative morphology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Stephen Cass, 2011. *Unthinking Machines, Artificial intelligence needs a reboot, say experts.*

Shay B Cohen, David M Blei, and Noah A Smith. 2010. Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572. Association for Computational Linguistics.

Ramy Eskander, Owen Rambow, and Tianchun Yang. 2016. Extending the use of adaptor grammars for unsupervised morphological segmentation of unseen languages. In *COLING*, pages 900–910.

Naomi H Feldman, Thomas L Griffiths, Sharon Goldwater, and James L Morgan. 2013. A role for the developing lexicon in phonetic category acquisition. *Psychological review*, 120(4):751.

Samuel J Gershman and David M Blei. 2012. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12.

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

Pawan Goyal and Amba Kulkarni. 2014. Converting phrase structures to dependency structures in sanskrit. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1834–1843.

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.

Eric A Hardisty, Jordan Boyd-Graber, and Philip Resnik. 2010. Modeling perspective using adaptor grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 284–292. Association for Computational Linguistics.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109. Association for Computational Linguistics.

James Jay Horning. 1969. A study of grammatical inference. Technical report, STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE.

Yun Huang, Min Zhang, and Chew Lim Tan. 2011. Nonparametric bayesian machine transliteration with synchronous adaptor grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 534–539. Association for Computational Linguistics.

Mark Johnson and Katherine Demuth. 2010. Unsupervised phonemic chinese word segmentation using adaptor grammars. In *Proceedings of the 23rd international conference on computational linguistics*, pages 528–536. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for pcfgs via markov chain monte carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007b. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648.

Mark Johnson. 2008a. Unsupervised word segmentation for sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27. Association for Computational Linguistics.

Mark Johnson. 2008b. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *ACL*, pages 398–406.

Mark Johnson. 2010. Pcfgs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1157. Association for Computational Linguistics.

Paul Kiparsky. 1994. Paninian linguistics. *The Encyclopedia of Language and Linguistics*, 6:2918–2923.

Donald E Knuth. 1992. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422.

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

Amrith Krishna, Pavankumar Satuluri, Harshavardhan Ponnada, Muneeb Ahmed, Gulab Arora, Kaustubh Hiware, and Pawan Goyal. 2017. A graph based semi-supervised approach for analysis of derivational nouns in sanskrit. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 66–75, Vancouver, Canada, August. Association for Computational Linguistics.

Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya. 2010. Introducing sanskrit wordnet. In *Proceedings on the 5th Global Wordnet Conference (GWC 2010), Narosa, Mumbai*, pages 287–294.

Arun Kumar, Lluís Padró, and Antoni Oliver González. 2015. Joint bayesian morphology learning of dravidian languages. In *RICTA 2015: Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects: Hissan, Bulgaria: September 10, 2015: proceedings book*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Percy Liang, Slav Petrov, Michael I Jordan, and Dan Klein. 2007. The infinite pcfg using hierarchical dirichlet processes. In *EMNLP-CoNLL*, pages 688–697.

Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.

Sivaja S Nair and Amba Kulkarni. 2010. The knowledge structure in amarakosa. In *Sanskrit Computational Linguistics*, pages 173–189. Springer.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 632–641. Association for Computational Linguistics.

Peter Norvig, 2011. *On Chomsky and the Two Cultures of Statistical Learning*.

Timothy J O'Donnell. 2015. *Productivity and reuse in language: A theory of linguistic computation and storage*. MIT Press.

Steven Pinker, Emilio Bizzi, Sydney Brenner, Noam Chomsky, Marvin Minsky, and Barbara H. Partee, 2011. *Keynote Panel: The Golden Age — A Look at the Original Roots of Artificial Intelligence, Cognitive Science, and Neuroscience.*

Alan Prince and Paul Smolensky. 1993. *Optimality Theory: Constraint interaction in generative grammar.* John Wiley & Sons, the version published in 2008.

Paul Smolensky and Géraldine Legendre. 2006. *The harmonic mind: From neural computation to optimality-theoretic grammar (Cognitive architecture), Vol. 1.* MIT press.

Partha Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. *Machine Learning and Knowledge Discovery in Databases*, pages 442–457.

Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. 2012. Exploring adaptor grammars for native language identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 699–709.

Ke Zhai, Jordan Boyd-Graber, and Shay B Cohen. 2014. Online adaptor grammars with hybrid inference. *Transactions of the Association for Computational Linguistics*, 2:465–476.

Ke Zhai, Zornitsa Kozareva, Yuening Hu, Qi Li, and Weiwei Guo. 2016. Query to knowledge: Unsupervised entity extraction from shopping queries using adaptor grammars. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 255–264. ACM.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation.