ADDRESSING LANGUAGE SPECIFIC CHARACTERISTICS FOR DATA-DRIVEN MODELLING OF LEXICAL, SYNTACTIC AND PROSODIC TASKS IN SANSKRIT

Amrith Krishna

ADDRESSING LANGUAGE SPECIFIC CHARACTERISTICS FOR DATA-DRIVEN MODELLING OF LEXICAL, SYNTACTIC AND PROSODIC TASKS IN SANSKRIT

Thesis submitted to the
Indian Institute of Technology, Kharagpur
For award of the degree

of

Doctor of Philosophy

by

Amrith Krishna

Under the supervision of

Dr. Pawan Goyal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

October 2019

APPROVAL OF THE VIVA-VOCE BOARD

Date: / / 20

Certified that the thesis entitled **"Addressing Language Specific Characteristics for Data-Driven Modelling of Lexical, Syntactic and Prosodic Tasks in Sanskrit"** submitted by Amrith Krishna to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Member of DSC) (Member of DSC) (Member of DSC)

(Supervisor)

(External Examiner) (Chairman)

CERTIFICATE

This is to certify that the thesis entitled ***"Addressing Language Specific Characteristics for Data-Driven Modelling of Lexical, Syntactic and Prosodic Tasks in Sanskrit"****, submitted by Amrith Krishna to the Indian Institute of Technology, Kharagpur, for the partial fulfillment of the award of the degree of Doctor of Philosophy in Computer Science and Engineering, is a record of bona fide research work carried out by him under my supervision and guidance.*

The thesis in my opinion, is worthy of consideration for the award of the degree of Doctor of Philosophy in accordance with the regulations of the Institute. To the best of my knowledge, the results embodied in this thesis have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Pawan Goyal
Assistant Professor
Department of Computer
Science and Engineering,
IIT Kharagpur

Date:

DECLARATION

I certify that

a. the work contained in this thesis is original and has been done by me under the guidance of my supervisor.

b. the work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Amrith Krishna

ACKNOWLEDGMENTS

First and foremost, I would like to thank my Advisor, Pawan Goyal. Pawan is knowledgeable, considerate, easy-going, and he is a mentor with whom every conversation I had was cheerful and uplifting. I am immensely grateful to Pawan Goyal for his guidance and constant support throughout my PhD journey, which culminated in this thesis. I would like to thank the members of my DSC for their valuable comments and suggestions during my stay at Kharagpur. I am grateful to Animesh Mukherjee, who initiated me into research during my initial days at Kharagpur. I wish to thank Plaban Kumar Bhowmick and Bivas Mitra with whom I was fortunate to work with during my MTech and initial phase of PhD, respectively. My associations with both professors have helped me shape my perspective towards research. I thank Niloy Ganguly for readily agreeing to serve as the chairman of the DSC, and for being the "Dutch uncle" who provides helpful feedback that does not confine to matters of research alone.

I extend my gratitude to Amba Kulkarni, Gérard Huet, Peter Scharf and Oliver Hellwig without whose support and guidance, my interest in Sanskrit Computational Linguistics would have remained yet another unpursued dream. I feel deeply privileged to have learned from each of them. I am indebted to Pavankumar Satuluri, my go-to expert on all things linguistic. I thank him for tolerating my naive questions on Sanskrit grammar and patiently answering them. I would like to thank him for going above and beyond the call of duty to make sure I have understood every aspect of the question I have in mind. I am grateful to Bishal Santra, Bodhisattwa Prasad

Majumder, Vishnu Sharma, Ashim Gupta and other collaborators who selflessly contributed to the works discussed in this thesis. I extend my gratitude to Arnab Bhattacharya, Ganesh Ramakrishnan, Rogers Mathew, Rohit Saluja, Devaraj Adiga, Hrishikesh Terdalkar, Madhusoodan Pai, Sanjeev Panchal and Ganesh Iyer for their comments and feedback.

Personal Support I would like to thank my friends as well as the past and present members of CNeRG for providing support and encouragement along the way. Unnikrishnan deserves a big shout out for supporting me in my intellectual pursuits right from my undergraduate days and for the innumerable fun conversations we had. I thank Binny Mathew and Ashutosh Soni for all the help, coffee, outings, patience, conversations, laughter, silliness, encouragement and our never-ending debates at the campus. My thanks to Sandipan Sikdar for all the thought-provoking discussions on a wide range of topics ranging from Quantum mechanics to Behavioural Economics. I thank Abhik Jana for being the "I can relate to" person for every possible step in my PhD, and for helping me through those. I am grateful that Nikita Vijay, Jasabanta Patro and Nature Poddar stayed close to me through PhD despite my never ending rants about my research. I thank Dibyasree Paul for being there for me even at the nadir of my PhD and for making me realise why Kolkata is truly the "city of joy". I thank Satadal Sengupta, Rohit Verma, Madhumita Mallick and Sudipa Mondal for their companionship. Finally, I thank my family: my mother and sister for all the care, love and encouragement. "All that I am, or hope to be, I owe to my father"[1], and I dedicate the thesis to him.

[1]Original quote from Abraham Lincoln - https://www.goodreads.com/quotes/18542-all-that-i-am-or-ever-hope-to-be-i

Author's Biography

Amrith Krishna received his B.Tech degree in 2012 from Department of Computer Science and Engineering, Federal Institute of Science and Technology. He then received M.Tech Degree in 2015 from Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur. After that, in the same year, he started his PhD journey in Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur under the supervision of Dr Pawan Goyal.

Publications from the Thesis

The publications (including communicated) are listed in chronological order, with comments in parentheses.

1. **Amrith Krishna**, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, Pawan Goyal. 2019. A Structured Prediction Framework Using Energy Based Models for Sanskrit. Computational Linguistics, MIT Press (Communicated).

2. **Amrith Krishna**, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, Pawan Goyal. 2019. Poetry to Prose Conversion in Sanskrit as a Linearisation Task: A case for Low-Resource Languages. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers, ACL 2019). Florence, Italy, 1160–1166.

3. **Amrith Krishna**, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, Pawan Goyal. 2018. Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018). Brussels, Belgium, 2550-2561.

4. **Amrith Krishna**, Bodhisattwa P. Majumder, Rajesh Bhat, Pawan Goyal. 2018. Upcycle Your OCR: Reusing OCRs for Post-OCR Text Correction in Romanised Sanskrit. In Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018). Brussels, Belgium, 345-355.

5. Vikas Reddy*, **Amrith Krishna***, Vishnu Sharma, Prateek Gupta, M R Vineeth, Pawan Goyal. 2018. Building a Word Segmenter for Sanskrit Overnight. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Languages Resources Association (ELRA). Miyazaki, Japan.

6. **Amrith Krishna**, Bodhisattwa P. Majumder, Anilkumar Boga, Pawan Goyal. 2018. An 'Ekalavya' Approach to Learning Context Free Grammar Rules for Sanskrit Using Adaptor Grammar. In Proceedings of the 17th World Sanskrit Conference (WSC 2018). D K Publishers Distributors Pvt. Ltd. Vancouver, BC, Canada, 83-112.

7. **Amrith Krishna**, Pavankumar Satuluri, Pawan Goyal. 2017. A Dataset for Sanskrit Word Segmentation. In Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature Workshop, 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017). Vancouver, Canada, 105-114.

8. **Amrith Krishna**, Pavankumar Satuluri, Harshavardhan Ponnada, Muneeb Ahmed, Gulab Arora, Kaustubh Hiware, 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 Workshop, 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017). Vancouver, Canada, 66-75

9. **Amrith Krishna**, Pavankumar Satuluri , Shubham Sharma, Apurv Kumar, Pawan Goyal . 2016. Compound Type Identication in Sanskrit: What Roles do the Corpus and Grammar Play? In Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), The 26th International Conference on Computational Linguistics. Osaka, Japan, 1-10.

10. **Amrith Krishna**, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu , Yajuvendra Singh, Pawan Goyal. 2016. Word Segmentation in Sanskrit Using Path Constrained Random Walks. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan, 494-504.

*The first two authors contributed equally

ABSTRACT

This thesis focuses on designing data-driven Natural Language Processing solutions for a low-resource language like Sanskrit. Our design decisions are motivated by taking the language's typology, usage and resource availability into consideration. The culture bearing language of India has about 30 million extant manuscripts that are potent for digitisation. Languages across the world exhibit wide typological variations, while the design decisions for modelling NLP solutions often are based on a few resource rich languages. In fact, many of these assumptions need not even be valid for a language like Sanskrit.

First, we show that we can successfully combine linguistic information from Sanskrit grammar, lexical networks and distributional information into data driven models for word segmentation and tasks pertaining to word-formation. We identify tasks, such as compound type identification and identification of derivational nouns, which are often overlooked in other languages, but are essential for processing Sanskrit texts due to the typological characteristics of the language. We also build a word segmentation model where linguistic information is incorporated, both in constructing the search space as well as in feature engineering. In the second part of the thesis, we investigate the applicability of purely data-driven approaches for sequence level tasks in Sanskrit, especially under low resource settings. Seq2Seq model based solutions are proposed for these tasks, where we compensate for the lack of training data by synthetic generation of training data and augmenting training with auxiliary resources. Here, we formulate the task of converting the word order in a verse to its corresponding prose order as a linearisation task. This enables the use of prose-only data as auxiliary training data, thereby avoiding the need for parallel data in verse and prose order.

Finally, we propose a generic graph-based parsing framework using energy-based

models for multiple structured prediction tasks in Sanskrit. We experiment with the tasks of word-segmentation, morphological parsing, dependency parsing, syntactic linearisation and poetry linearisation. For a free word order language like Sanskrit, we observe that the graph based non-sequential processing of input outperforms sequential processing approaches even for low level tasks such as word segmentation and morphological parsing. We automate the learning of the feature function which, along with the search space, encodes relevant linguistic information for the tasks we consider. This effectively results in the use of as low as 10 % of the task-specific training data as compared to the data requirements of the current neural state of the art models for various tasks.

The major contributions of the thesis include solutions to a range of challenges in analysing Sanskrit. We observe that integration of linguistic knowledge, both in designing search space and feature function, substantially reduces the task-specific training data requirements, a desirable aspect for low-resource languages. We also introduce two novel tasks of verse to prose order conversion and poetry linearisation, which can aid digitisation and processing of Sanskrit texts. In all the tasks we discuss, we achieve state of the art results and in some of the tasks, ours is the only data driven solution.

Keywords: Sanskrit Computational Linguistics, Morphology, Syntax, Low-resource Language, Word Segmentation, Morphological Parsing, Dependency Parsing, Syntactic Linearisation, Poetry Linearisation, Word Formation, Compound Type Identification, Derivational Nouns, Post-OCR Text Correction. Structured Prediction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"A language is
not just words. It's a culture, a tradition,
a unification of a community, a whole
history that creates what a community
is. It's all embodied in a language."
Noam Chomsky

Sanskrit is a classical language [Cou92, Sap04] and was the prevalent medium of knowledge transfer in the demographics of the Indian subcontinent for about three millennia [KSG17]. It is estimated that there are about 30 million extant manuscripts in Sanskrit that are potent for digitisation [GHK$^+$12]. While there have been tremendous advancements in the digitisation of ancient manuscripts in Sanskrit in the last couple of decades, there exist relatively limited options for completely automated tools [HN18, AGP$^+$18, Hel09b] that can aid in scaling up such digitisation efforts in Sanskrit. In this thesis, we identify several challenges, which might be incidental in other languages but are essential for processing texts in Sanskrit. The thesis aims to design data-driven monolingual natural language processing (NLP) solutions to address these challenges. The thesis focuses specifically on tasks at lexical (or more specifically, word formation), syntactic and prosodic levels in Sanskrit.

The characteristics of Sanskrit have been greatly influenced, and often evolved, based on the cultural history of its usage that spans about three millennia. The language has

a rich history of knowledge transmission through oral tradition [Sta08]. This led to the development of a fairly advanced discipline of phonetics, shaping the characteristics of the language and its usage [GH16]. Similar to what one observes in connected speech in spoken utterances [MMG04, ST03], the word boundaries may get obscured due to euphonic assimilation of phones at the juncture of the words [Whi89]. The presence of an advanced discipline of phonetics in Sanskrit formalises this euphonic assimilation of phones as *Sandhi* (सन्धि). Given that the writings in Sanskrit follow phonemic orthography, these euphonic assimilations get reflected in writing as string transductions at the word boundaries. The phenomena of *Sandhi*, make word segmentation in Sanskrit a challenging task, possibly more difficult than in languages such as Chinese [GH16]. While morphological segmentation tasks, especially canonical segmentation tasks [KCS16], are familiar with such string transductions at the juncture of the morphemes, the transductions in our case happen at the word boundaries in a sentence. The sentence that has undergone *Sandhi*, on its analysis may lead to ambiguity as there can be multiple possible segmentation analyses for it. This makes identifying meaningful word units from the sequence a challenging task.

Further, Sanskrit is a highly productive language when it comes to word formation, especially with compounding and derivational morphology. Compounds are extensively used in literary works in Sanskrit, where corpora such as the Digital Corpus of Sanskrit [Hel16a] have a compound presence as high as 75 % in its vocabulary. Compound-formation is recursive, implying that multiple free morphemes can exist in a compound. Such multi component compounds are commonly used in Sanskrit, especially in literary works. For instance, '*pravaramukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ*' (प्रवरमुकुटमणिमरीचिमञ्जरीचयचर्चितचरणयुगलः), is a compound, originally used in *Pañcatantram*, and it consists of 9 stems, which are, *pravara* (प्रवर), *mukuṭa* (मुकुट), *maṇi* (मणि), *marīci* (मरीचि), *mañjarī* (मञ्जरी), *caya* (चय), *carcita* (चर्चित), *caraṇa* (चरण), *yugalaḥ* (युगलः).[1] While linguistically this compound can be treated as a single word, doing so by ignoring its component information will create a corpus with a vocabulary that has high type to token ratio. This is not desirable in data-driven models. Hence it is essential to analyse the compound by splitting it into component stems and improve the distributional evidence of those stems. Additionally, given the extensive use of compounds in Sanskrit

[1] The compound word translates to - "The pair of whose feet was covered with a stream of rays originating from the gems in wreaths of eminent noble kings,"

corpora, identifying the type of the compound is essential for the semantic processing of texts. More importantly, there exist compound words of several types, with many of them being non-compositional, and a compound of one type may be embedded within a compound of same or different type [Low15].

The language is morphologically rich, not just in derivational morphology, but also in inflectional morphology. The rich inflectional morphology expressed by the fusional language, with a tagset of 1,635 tags, results in a high degree of syncretisms. Further, the role of morphological case markers in deciding the syntactic role of the words in a sentence is a well-developed formalism in Sanskrit. The formalism, called *kāraka* (कारक) theory [BS90, BKS19], clearly shows the tight interactions between morphology and syntax in Sanskrit. Syntactic analysis, when modelled jointly with its preliminary upstream tasks has shown to improve its performance for several languages including Czech, Hebrew, Chinese etc. [SK13, Tsa06, HMMT12]. We observe this should apply to Sanskrit as well. As an illustrative case, consider the eight letter string 'nagarāṇi' (नगराणि). The string can either be tokenised (segmented) as 'na garāṇi' (न गराणि, no diseases) or be treated as an inflection of the stem 'nagara' (नगर, town). Assuming the latter is correct, 'nagarāṇi' is a plural form of the neuter gender lemma 'nagara', which can either be in its nominative-case, vocative-case or in its accusative-case. This is syncretism, and the syntactic analysis of the word is dependent on the case information of the word. Assuming the inflection to be in nominative-case, this makes the word eligible to be assigned as either the subject or as the object in the syntactic analysis of the sentence. This shows that the use of morphosyntactic information from the rest of the words in the sentence is desirable for resolving these ambiguities at various levels of processing texts.

Sanskrit is a relatively free word-order language [KSSS15], a characteristic typical of morphologically rich languages [SK13]. It is observed that, Sanskrit sentences, written in prose, generally tend to follow SOV typology. The linear ordering of words in such a sentence tends to abide by the principle of dependency locality[2] [Gib98], and the corresponding dependency analysis exhibits weak non-projectivity [KSSS15]. But, none of these holds true for verse constructions in Sanskrit. Since morphologically rich languages rely more on their morphological markers to carry the structural information in a sentence, poets often exploit

[2]The principle of dependency locality, states that the linear distance between two syntactically related words should be as less as possible, and preferably not be intervened with a syntactically unrelated word [GFP+19].

this notion while constructing verses. The primary objective of a poet while constructing a verse is to find a word arrangement, which results in a sequence (of syllables) that adheres to one of the prescribed metrical patterns in Sanskrit prosody. As a consequence of this, the word arrangement in a verse need not align with its verbal cognition [Bha90]. From the pedagogical perspective, a verse is often rearranged into its corresponding prose order by an expert, to facilitate better comprehension of the sentence [SKS+16]. A sentence, when written in either the prose or the original verse-order, still maintains the same dependency structure, despite the differing word orders. A possible exception to this will be the violation of weak non-projectivity in the verse order.

Figure 1.1 shows such a verse and the corresponding prose ordering, from Rāmāyaṇa (रामायण). The sentence in its original verse form neither follows SOV typology nor adheres to the principle of dependency locality, while its corresponding prose ordering follows both. In the verse order, the subject in the sentence appears after the object and the verb. As a matter of fact, the word *tapasvī* (तपस्वी) is an adjective to the word *vālmīkiḥ* (वाल्मीकिः), and hence *tapasvī* is syntactically related only to its headword *vālmīkiḥ* in the sentence. But, these words are placed far apart in their linear order in verse, thereby violating the principle of dependency locality. This shows that prosodic constraints take precedence over syntactic constraints in verse constructions. This will result in verse constructions with syntactically arbitrary word orderings. Texts from pre-classical and classical (300 CE - 800 CE) eras exhibit fairly abundant use of text contents in verse format. Due to the prevalence of both prose and poetry styles in Sanskrit texts, systems that consider the entire sentence-context are desirable, as compared to systems that consider only the linearly adjacent contexts.

Finally, from a data-driven modelling perspective, resource availability is another major challenge in processing Sanskrit. Sanskrit is a low-resource language in terms of data, tools and other resources for the processing of Sanskrit texts. Lack of task specific labelled data for a majority of the tasks is a challenge in itself. Summarily, the thesis focuses on designing monolingual NLP solutions for a low-resource, morphologically-rich, free word-order, compound-rich language by considering its typology, usage and resource availability.

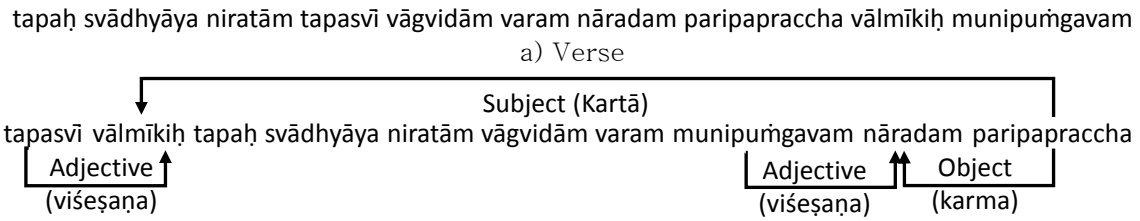


Figure 1.1 Word ordering in verse and the corresponding prose order.

1.1 Objective

As stated previously, the thesis aims to design data-driven monolingual natural language processing (NLP) solutions for a low resource language like Sanskrit. Given the history of usage of Sanskrit and the challenges in processing Sanskrit texts, we now elaborate on the objectives of this thesis.

How effectively can the linguistic knowledge and distributional information be integrated into designing data-driven solutions for Sanskrit? Sanskrit has a rich grammatical tradition, which gave rise to a generative grammar [Blo29, Kip94] like *Ashtādhyāyī* (अष्टाध्यायी, circa 500 BCE). Our objective here is to investigate how we can effectively integrate linguistic knowledge from traditional grammatical sources in Sanskrit and distributional information from data, for developing data-driven models for various tasks in Sanskrit. Generally, in Sanskrit computational linguistics, rules from *Ashtādhyāyī* were utilised under formal language frameworks for the development of rule based systems. Here, we are specifically interested in the usability of *Ashtādhyāyī* rules as discriminative features in data-driven machine learning models. We begin our investigations by considering classification models for the identification of compound type and identification of derivational nouns. *Ashtādhyāyī* is quite elaborate in specifying rules pertaining to compounding and derivational morphology, as it devotes more than one-third of the entire rule-book for these two word formation processes. The compound type identification task focuses mostly

on feature engineering, by integrating rules from *Ashtādhyāyī*, semantic relations from lexical networks and distributional information from data. We propose a graph based label propagation approach for the derivational nouns identification task. Here, in addition to designing features similar to that of compound type identification task, we rely on the string transductions involved in the derivation process of the nominals for the explicit design of the graph structure for label propagation. In the case of sentence level tasks, we formalise them into search based structured prediction tasks. Here, we obtain linguistically informed search spaces for these tasks, and also we use this information for the sampling of candidates during the inference. Additionally, for these tasks, we learn feature functions which effectively are distributional information obtained under specific linguistic constraints.

How effectively can the data requirements for purely statistical models be mitigated when designing monolingual NLP solutions for Sanskrit? Sequence to sequence models have achieved impressive performances in modelling multiple sequence level tasks in NLP, but generally under high resource settings [GHDL18, SB19]. Though incorporating linguistic information into these models is one way to mitigate the data scarcity, not all tasks are apt for this. For instance, consider the case of an OCR based solution for Sanskrit texts written in Roman scripts. Here, instead of training an OCR from scratch, we use OCRs trained for other languages written in Roman scripts. We then train a Seq2Seq Post-OCR text correction model. Here, the training data is synthetically generated by obtaining images out of digitised texts and adding distortions that resemble real world noise settings. Similarly, we consider the task of converting a verse to its corresponding prose order by framing it as a linearisation task [SRS16]. This avoids the need for parallel sequences in verse and prose order. We then augment training data by using prose only sequences in Sanskrit as training data. This model of ours reports competitive performance with a linguistically involved model. Summarily, we address practical challenges in processing Sanskrit texts by exploring the possibility of synthetic generation of training data and augmenting training with auxiliary resources for purely statistical solutions.

Can we automate the learning of linguistically motivated feature functions? For morphologically rich languages, state of the art models for morphosyntactic tasks still rely on hand-crafted features for the performance [Mor16, MSBT19]. While this is beneficial in the case of Sanskrit, it is a challenge to design features for each task. In this thesis, we

experiment with five sequence level structured prediction tasks using a common structured prediction framework we propose. Here, four of them are syntax level tasks (or preliminary tasks to syntax), and the fifth one is a prosody level task. For all the four syntax level tasks, we investigate the feasibility of learning a common feature function. Using the same framework, we automate the learning of a new feature function for the prosody level task, where the domain information requirement, i.e. prosody level information, is minimal.

1.2 Contribution

We now elaborate the main contributions from this thesis. The thesis focuses on designing data-driven Natural Language Processing solutions for a low-resource language like Sanskrit by taking the factors such as its typology, usage and resource availability into consideration. Here, we first identify the tasks, such as compound type identification, morphological derivation, etc., which are not of primary importance in other languages, but are essential for processing Sanskrit texts due to its typological characteristics. We attempt to address the low-resource nature of the language and design solutions that rely less on task-specific labelled data by incorporating rich linguistic knowledge into our models. In Sanskrit, along with syntax, prosody plays a key role in linguistics. We introduce new tasks to address these challenges and propose novel solutions to match with the requirements specific to Sanskrit for other tasks. We elaborate each of these below:

A Generic Framework for Structured Prediction We propose a graph based parsing framework using Energy Based Models [LCH$^+$06] for multiple structured prediction tasks in Sanskrit. Here, we experiment with the tasks of word segmentation, morphological parsing, dependency parsing, syntactic linearisation and poetry linearisation. The framework enables joint modelling of several related downstream tasks, thereby reducing error propagation prevalent in pipeline modelling of these tasks. The framework, in principle, is language agnostic in its design. At the same time, it enables to incorporate language specific constraints which facilitate pruning of the search space of solutions and in designing inference procedures. While many of the state of the art models for morphosyntactic tasks in morphologically rich languages rely on hand crafted features [MSBT19, MT16, SÇ15],

we automate the learning of feature function in our case. The approach essentially learns relevant horn clauses from a space of infinite possible clauses, where the only domain knowledge we require is to define the finite set of literals involved in defining the clauses.

Creating Tools, Data and Tasks for Processing Sanskrit Texts We introduce two new tasks, along with novel data-driven solutions to solve them, that deal with syntax and prosody in Sanskrit. First, we introduce the task of converting a verse to its corresponding prose order for facilitating better verbal cognition. We automate this process using a purely Seq2Seq data-driven approach and also as a linguistically involved structured prediction task using the aforementioned framework. We also introduce a prosody level task, as a linearisation task, where a given bag of words needs to be arranged into a valid verse order that adheres to prosodic constraints [MGS13]. We solve it using the aforementioned structured prediction task. We release a dataset for word segmentation in Sanskrit. Here, we effectively address the design differences between the existing resources for Sanskrit, which enabled the release of a set of 115,000 sentences for segmentation along with the search space of all the possible phonetically valid candidate splits. Our design decisions are linguistically informed and are rooted in the traditional grammatical framework. Apart from proposing new tasks, in all the tasks we address here, we either achieve state of the art results or are the first to propose a data-driven solution for the task.

Designing NLP Solutions Under Low Resource Settings Sanskrit is a low resource language both in terms of availability of task specific labelled data and tools for processing the texts. We employ numerous novel approaches by incorporating linguistic knowledge as rich pretraining, synthetic generation of training data and augmenting training with auxiliary resources. We first consider the task of digitisation of scanned documents written in Romanised Sanskrit, where there exists scarcity in terms of available tools as well as training data. Here, we make use of robust OCR solutions developed for other languages written in Roman script and then pose the problem as a post-OCR text correction task. For the task, we completely use synthetically generated training images, which replicate the settings on two different scan settings based on a real world dataset. Our Seq2Seq approach using copying mechanism outperforms existing OCR solutions for Romanised Sanskrit when tested on a real world dataset. The scarcity of enough parallel sequences for the poetry to prose conversion, motivated us to

formulate the task as that of a linearisation task. Here, we ignore the ordering of words in the verse and consider the input as a bag of words. This enables us to incorporate texts from Sanskrit Wikipedia which are written exclusively in prose order, i.e. those constructions that do not have a corresponding verse order, as an auxiliary source. While we originally had a training data close to 13,000 parallel sequences of verse and prose order, we increased this to more than 108,000 prose only training sequences, and this helped us to mitigate the data scarcity for our proposed Seq2Seq model. For the tasks in the aforementioned structured prediction framework, task-specific labelled data is particularly hard to come by. All our models use as low as 10 % of the training data as required by the current neural state of the art models for various tasks. We used around 9.26 % (10,000) and 1.5 % (8,200) of training data as against 108,000 and 0.5 million training sentences for state of the art neural models in syntactic linearisation [KSS+19] and word segmentation [HN18] for Sanskrit. The linguistically involved construction of the search spaces and the learning of effective feature functions enabled this.

1.3 Organization of the Thesis

In this section, we provide a brief description of the organization of the thesis.

- **Chapter 1** discusses the challenges in processing Sanskrit texts by elaborating the language's typological and usage related characteristics. This chapter also elaborates the objectives of this thesis. Finally, the major contributions from the thesis are also described in this chapter.
- **Chapter 2** examines the current literature in terms of advancements made in the computational processing of Sanskrit. We also review the relevant approaches used in NLP for solving similar tasks in other languages.
- **Chapter 3** essentially considers three tasks pertaining to word formation and word segmentation. Here, we focus on the merits of integrating linguistic information with distributional information in our machine learning models. We experiment with compound type identification, identification of derivational nouns in Sanskrit, and joint word segmentation and compound splitting tasks. All of these tasks required careful

linguistically motivated feature engineering. Additionally, the word segmentation problem uses a search space of linguistically relevant candidates.

- **Chapter 4** focuses on using purely statistical approaches for solving various sentence level tasks in Sanskrit. To be precise, we deal with three Seq2Seq models proposed for word segmentation, Post-OCR text correction and conversion of verse to its prose order. We first show that a standard Seq2Seq architecture with attention can be used as a scalable solution for word segmentation when trained with sufficient data (about 110,000 sequences). However, the remaining two tasks face scarcity of resources for training the Seq2Seq models. For the post OCR-text correction, we augment the standard Seq2Seq architecture with copying mechanism and then synthetically generate training data for the task. For prose to verse order conversion, we use a Seq2Seq model with a sequence level loss and two pretraining approaches.

- **Chapter 5** discusses the structured prediction framework we propose for multiple sentence level tasks in Sanskrit using Energy based models. The model is a search based structured prediction framework which expects a graph as input and the inference procedure searches for task-specific substructures. For a free word order language like Sanskrit, we observe that the non-sequential processing of input outperforms sequential processing approaches even for low level tasks such as word segmentation and morphological parsing. We also elaborate our approach for automating the learning of a linguistically motivated feature function which, along with the search space, encode relevant linguistic information for the tasks we consider.

- **Chapter 6** concludes the thesis, where we summarise the key contributions made in this thesis and outline the research directions for future work.

Chapter 2

Related Work

We present a survey of the related work that aligns with the scope of this thesis. The objective of this thesis is to develop data-driven solutions to tackle numerous NLP tasks by addressing the language-specific characteristics of Sanskrit language. Keeping this in mind, we first provide a brief description of the developments in Sanskrit Computational Linguistics over the past three decades. Next, we provide a general overview of the methodologies adopted in NLP for solving the tasks we consider in this thesis. We also discuss the relevant background and the language specific characteristics, that need to be addressed, for each of these tasks.

2.1 Sanskrit Computational Linguistics

The last three decades have shown considerable interests in computational processing of Sanskrit texts, though mostly centred around formal language theory [Kip09, GH16, Hym09] and rule based systems [Kul13, KSSS15, GHK+12]. *Aṣṭādhyāyī* (अष्टाध्यायी), the grammar treatise of Sanskrit, is admired for its brevity in the description of the grammar, completeness of its coverage of the language and the computational insights it provides [KSSJ14]. Early attempts in formalising the rules in *Aṣṭādhyāyī* were proposed by Staal [Sta67] and Cardona [Car65]. These works highlighted the context sensitiveness of several rules, especially the context sensitiveness of the metarules stated in *Aṣṭādhyāyī*. Cardona [Car09] argues that

morphology and syntax forms a continuum, and affixation procedures are well integrated with the syntactic machinery in the grammar. Hyman [Hym09] observes that the rules pertaining to *sandhi* (सन्धि), as stated in *Aṣṭādhyāyī*, can be formalised into constructs within the expressive power of regular languages. Further, Graf and Mayer [GM18] propose that the segmental process of Sanskrit *n*-retroflexion in phonology can be fit into a special class of finite state of languages, which they call as the input-output tier-based strictly local (IO-TSL). Here, the authors hypothesise that *n*-retroflexion forms the upper bound on the complexity of segmental phonology, and this implies that the entire segmental phonology processes can be captured within the complexity of IO-TSL. Moreover, Penn and Kiparsky [PK12] establishes that the underlying formalism under which *Aṣṭādhyāyī* is constructed far exceeds the power of Context-Free Languages, and it even exceeds the expressive power of multiple-component tree-adjoining languages. This further strengthens the arguments supporting the context sensitiveness of the formalism used by *Pāṇini* for *Aṣṭādhyāyī*.

Several rule based tools for morphophonemic analysis of Sanskrit texts have been developed in the past two decades [Hel09b, GH16, GHK+12]. Sanskrit Heritage Reader [GH16, Hue09] performs exhaustive enumeration of the possible segmentation and morphological analyses for Sanskrit texts. It essentially is a finite state transducer, and can be seen as an instance of Eilenberg machines, a relational computing paradigm [HR06]. Similarly, a morphological analyser [AB06, KS09] and a constraint based dependency analyser [Kul13, KPS10] was developed independently for analysing digitised texts in Sanskrit. Kumar [Kum12] proposed a compound processor, which identifies the components of a compound, constructs its constituency parse structure [KK11] and constructs the paraphrase [KSK09] for the compound from this constituency parse. These systems were also used to augment anusāraka, a language accessor and Sanskrit-Hindi translation system [BCSR95] from the Akshar Bharati group. The aforementioned systems are essentially rule based analysers, rooted in the Sanskrit grammatical tradition. While these systems exhaustively enumerate linguistically valid candidate analyses, the disambiguation of these analyses still requires human intervention. Digitisation efforts of ancient manuscripts, from the late 80s such as TITUS[1], GRETIL and Digital corpus of Sanskrit [Hel16a], resulted in a growing collection of digitised versions of ancient manuscripts, dictionaries and lexicons. With an objective to facilitate large scale analysis of such digitised texts, the aforementioned systems

[1] http://titus.uni-frankfurt.de/indexe.htm

were made interoperable and integrated into a single distributed platform [GHK+12].

2.2 Compound Type Identification

The relation between the components of a compound is often implicit [KB05], and this makes the identification of such relations challenging. The research approaches proposed for the identification of semantic relations can be broadly classified as linguistic approaches [Fin80, Dow77, Lev89] or as computational approaches [MBT+, BS98, RH01]. These approaches, especially the computational approaches, vary widely on the relation types defined, features used and the methods applied [KB13, FOV18]. The taxonomy of relations typically varies from a coarse grained set of 5 relation types to a more fine grained set of 43 relation types [TH10]. Similarly, in Sanskrit, depending on the granularity one wishes to make, there can be up to 30 different relation types. But these relations are essentially sub-types of a four-class coarse-grained classification of semantic types [Low15, Gil09], which we follow in our compound type identifier [KSS+16b].[2]

Computational approaches for semantic analysis of compounds have been proposed for several languages including English, German, Italian, Afrikaans and Dutch [VDVZVH14]. Lexical databases like WordNet [KB05] and Wikipedia [SP06], often augmented with surface level features [TH10], were extensively used to infer semantic relations between the components in a compound. Similarly, methods relying on distributional approaches [SC13] or combining both lexical and distributional information [NSSSS06] were also studied. Further research explored the role of verb-semantics [KB06] and the influence of word sense of the components involved in compound formation [KB13]. Weller et al. [WCM+14] proposed a method to identify compositional compounds from the non-compositional ones for machine translation tasks. Séaghdha [Séa09] defines kernel functions defined on the WordNet lexical hierarchy for identifying the relational similarity between the components. Recent approaches focused on utilising pretrained word embeddings [DH15] and dependency path based embeddings [SW18] as features for compound type identification. The aforementioned approaches employed different learning methods ranging from nearest neighbour classifiers [KB05], maximum entropy models [TH10], SVMs [Séa09] and

[2]This model is elaborated in Section 3.2

neural network based classifiers [DH15, SW18] including multi task and transfer learning settings [FOV18]. In Sanskrit, Satuluri [Pav16] proposed a compound generator, which doubles as a tool for pedagogy as it emulates the entire derivation process of deriving a compound from its components as per *Aṣṭādhyāyī*. Kulkarni and Kumar [KK13] proposed a compound type identification approach, which investigates the efficacy of rules from *Aṣṭādhyāyī* for the task. The rule based system is often constrained due to its coverage issues when applied on large compound collections. We use insights from this system for feature engineering in our proposed classifier (Section 3.2).

2.3 Identification of Derivational Nouns

NLP solutions for identification of morphologically related words, primarily focus on capturing the similarity of the orthographic patterns between these words [GJ04, SB08, DE11, Wic04]. Previously, a number of works have attempted to integrate semantic information for the analysis of morphologically related words, where the semantic information is applied over a filtered set of candidates obtained using orthographic similarity [SJ00, YW00]. Baroni et al. [BMT02a] obtain two independent filtered lists of candidate word pairs, one based on orthographic similarity using minimum edit distance and the other based on semantic similarity using mutual information. These lists were then combined using a deterministic weighting function [NBJ15]. Similar to Schone and Jurafsky [SJ00], where latent semantic analysis was used to incorporate semantic feature vectors, newer unsupervised approaches incorporated word embeddings [SO15, NBJ15]. Narasimhan et al. [NBJ15] addressed the recursiveness involved in the affixation of word forms and proposed an approach to learn morphological chains, which identify the base forms by performing morphological segmentation. While all these approaches used semantic vectors as features, Lazaridou et al. [LMZB13] used the CDSM [ML10] framework to find the compositional representations of the derived word forms, where they assume that the word embeddings for the morphemes involved are provided. Bhatia et al. [BGE16] proposed to learn embeddings for a word-form from its morphemes, where the morphological segmentation of a word is performed using Morfessor. This approach, similar to Soricut and Och [SO15], does not make a distinction between inflectional and derivational morphology. Cotterrel and Schütze [CS17], proposed

a joint model for canonical segmentation of derived words, along with learning semantic vectors for the morphemes involved. Methodology wise, our work for identification of derivational nouns[3] [KSP+17] stands closest to the work of Faruqui et al. [FMS16], originally proposed for morpho-syntactic lexicon induction using a semi supervised graph based label propagation approach. We use Modified adsorption [TC09], a label propagation approach that enables the explicit design of the graph structure for our task. Unlike Faruqui et al. [FMS16], where their approach focuses on inflectional morphology, our approach is designed specifically for derivational morphology. Previously, Krishna and Goyal [KSSJ14] proposed a tool for the generation of derivational nouns in Sanskrit, where the system provides not only the derived form but also all the intermediate string transductions and the trace of the rules from *Aṣṭādhyāyī* involved in the derivation process of the particular word. The system design is inspired by the arrangement of the grammar rules in *Aṣṭādhyāyī* [Bha89, Deo07].

2.4 Word Segmentation and Morphological Parsing

Word segmentation and morphological parsing are low-level, yet non-trivial, tasks for multiple languages and are extensively researched in NLP. Traditionally, solutions for these tasks were proposed using (Probabilistic/Weighted) Finite state transducers [KK81, SSGC96, Moh97, BK03] and using unsupervised pattern discovery approaches [Gol01, AAAK04, JG09]. Broadly these approaches could be categorised as lexicon driven [Hue03, CL92], purely statistical [Eis02, SSGC94] or both [SSGC96]. Sequence labelling approaches such as HMMs [HTOT00] and CRFs [SST05, Xue03] were later proposed for these tasks. Further, lattice parsing based approaches, where the input search space was represented as word-level lattices, were incorporated into CRFs [SST05] for joint modelling of these tasks [KYM04]. Neural sequence labelling approaches currently achieve state of the art performance in word segmentation, especially for Chinese, Korean, Japanese etc. [WVM14, SHN18]. Similarly, higher-order CRFs [MSS13] and neural morphological taggers [MGN18, TS18] are widely used for morphological parsing. Recently, neural lattice based approaches were proposed for word segmentation [BN18, YZL19]. Interestingly, lattice based structures and lattice-parsing techniques remain hugely popular for morphological parsing, especially for morphologically

[3]This method is elaborated in Section 3.3

rich languages [SÇ15, Mor16, MSBT19].

For word segmentation in Sanskrit, Natarajan and Charniak [NC11] proposed a Bayesian modelling approach, based on the model from Goldwater et al. [GGJ06], where the authors tailored their model to handle transductions due to *Sandhi*. Similarly, Mittal [Mit10] uses maximum aposteriori estimates to score the candidate segments. Subsequently, Krishna et al. [KSS+16a] outperformed both these models with a percentage gain of 28.82 % as compared to the aforementioned two models.[4] The problem was posed as an iterative query expansion problem and used the search space from Sanskrit Heritage Reader [GH16]. Hellwig [Hel15b] was the first to pose the problem as a neural sequence labelling task. Similarly, Hellwig [Hel16c] proposed a sequence level model for neural morphological tagging in Sanskrit, but with a limited tagset of 86 tags where the tense and the mode information of the verbs were ignored. Both Reddy et al. [RKS+18][5] and Aralikatte et al. [AGP+18] proposed Seq2Seq models for the task. Aralikatte et al. [AGP+18] considered the word segmentation as a multitask problem, using a common encoder with two decoders, where one decoder predicts the split location and the other is used to generate the characters in the split word. Hellwig and Nehrdich [HN18] proposed a considerably improved sequence tagger for the task, where the architecture incorporated recurrent and convolutional neural units, in addition to the use of shortcut connections. Krishna et al. [KSB+18] proposed a joint model for word segmentation and morphological parsing in Sanskrit using Energy Based Models [LCH+06].[6] The model formalised the problem set up in Krishna et al. [KSS+16a] as a structured prediction task, where both models used Sanskrit Heritage Reader for creating the search space.

In this thesis, we address word-segmentation in Sanskrit both as a open-vocabulary problem as well as a lexicon driven closed-vocabulary based problem. The linguistically motivated approaches of ours[7] use a lexicon driven shallow parser, the Sanskrit Heritage Reader [Hue15, GH16] for generating the search space of possible split candidates. On the other hand, we propose a Seq2Seq model[5] that treats the problem as an open vocabulary problem. The model uses subword based tokensiation mechanisms, such as Sentencepiece [SN12] and BPE [SHB16], to learn a new vocabulary for the task. We describe these systems

[4] This model is elaborated in Section 3.4

[5] This model is elaborated in Section 4.1

[6] This model is elaborated in Chapter 5

[7] These models are discussed in Section 3.4 and Chapter 5.

briefly here. We also provide a short description of the dataset we use for these systems.

Sanskrit Heritage Reader The Sanskrit Heritage Reader is a lexicon driven system. It uses finite state methods in the form of a lexical juncture system to obtain the possible segments for a given sentence. We follow the definitions as used in Goyal and Huet [GH16] and it is recommended to refer the work for a detailed review of the system. A *lexical juncture system* on a finite alphabet Σ is composed of a finite set of words $L \subseteq \Sigma^*$ and a finite set R of rewrite rules of the form $u|v \rightarrow f/x\_\_$ [KK94], with $x, v, f \in \Sigma^*$ and $u \in \Sigma^+$. In this formalization, Σ is the set of phonemes, R is the set of Sandhi rules, and L is the vocabulary as a set of lexical items. We define $z \in L$ as a 3-tuple (l, m, w), where l denotes the lemma of the word, m denotes the morphological class of the word, w denotes the inflected word form generated from l and m. Given a sentence s, its segmentation solution/sandhi analysis, S_i, can be seen as a sequence $\langle z_1, \sigma_1, k_1\rangle; ...\langle z_p, \sigma_p, k_p\rangle$. Here, $\langle z_j, \sigma_j, k_j\rangle$ is a segment with $z_j \in L$, $k_j \in \mathbb{N}$ denotes the position at which the word w_j begins in the sentence s and $\sigma_j = [x_j]u_j|v_j \rightarrow f_j \in R$ for $(1 \leq j \leq p)$, $v_p = \epsilon$ and $v_j = \epsilon$ for $j < p$ only if $\sigma_j = o$, subject to the matching conditions: $z_j = v_{j-1}y_jx_ju_j$ for some $y_j \in \Sigma^*$ for all $(1 \leq j \leq p)$, where by convention $v_0 = \epsilon$. Finally $s = s_1...s_p$ with $s_j = y_jx_jf_j$ for $(1 \leq j \leq p)$, ϵ denotes the empty word.

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

A segment $\langle z_j, \sigma_j, k_j\rangle \in D(\mathcal{S})$, iff $\langle z_j, \sigma_j, k_j\rangle$ exists in at least one S_i. Two segments $\langle z_i, \sigma_i, k_i\rangle$ and $\langle z_j, \sigma_j, k_j\rangle$ are said to be 'conflicting' if $k_i \leq k_j < k_i + |w_i|-1$ or $k_j \leq k_i < k_j + |w_j|-1$. No two conflicting segments exist in a valid segmentation solution.

Subword Tokenisation Systems In neural Seq2Seq approaches, such as Neural Machine Translation, the open vocabulary issue is often dealt with by breaking up the rare words into subword units [Kud18]. These approaches essentially learn a new vocabulary for a given corpus by identifying frequent character patterns (subwords) from the corpus. These regularities are captured using greedy approaches and are used to create a vocabulary of the desired size. The subword tokenisation approaches can be fine-tuned to enhance the size and the type-to-token ratio of the entries in the vocabulary and also to avoid Out of Vocabulary words in corpora. Another advantage with these approaches is that theses can effectively

balance the vocabulary size and the step size [Kud18]. Once the vocabulary is learnt, it can be used to re-encode any sequence in the language, such that the sequence contains tokens from the newly learnt vocabulary. However, a downside for these approaches is that the same sequence may be split into different subword-sequences, even with the same vocabulary. This introduces non-determinism in the behaviour of these systems [Kud18]. In this thesis, we use two different subword tokenisation systems, namely, BPE [SHB16] and Sentencepiece [SN12]. While learning the vocabulary, both the systems initialise their vocabulary with the set of characters in the language's alphabet and then split each sequence in the training data into a sequence of entries in the current vocabulary. Then these entries are iteratively merged to form subword units. In the case of BPE, the frequency of these entries is used to determine the subword patterns to merge. However, in sentencepiece, entries that can increase the vocabulary likelihood on the training data the most, are chosen. Further, while BPE learns patterns that strictly confine within the boundaries of a word, the sentencepiece model does not restrict itself to word boundaries. It may learn patterns across word boundaries as well.

Digital Corpus of Sanskrit We use the Digital Corpus of Sanskrit (DCS) [Hel16a], a morphologically tagged corpus of Sanskrit, for all our word segmentation related experiments. The corpus mostly consists of texts from Epic and scientific domains such as medicine, astronomy, grammar etc. [HN18]. DCS contains digitised works from periods that span over 3000 years and contain constructions written in prose or poetry. This makes the corpus a collection of manuscripts which are stylistically, topically and chronologically diverse. It contains more than 66,000 unique lemmas and 3,200,000 tokens. Identifying sentence boundaries in Sanskrit constructions is a challenge of its own [Hel16b]. DCS currently has split the corpus into more than 561,596 text lines, all of which need not be following explicit sentence boundaries [HN18, KSB$^+$18]. The analysis of a text line essentially consists of the word, the stem of the word and its morphological analysis.

2.5 Linearisation

Word linearisation has been used in various Natural Language Generation tasks for the past 3 decades. Mel'čuk [Mel88] identifies linearisation as one of the three steps in a syntactic

framework for language generation. The three steps are syntactic generation, syntactic linearisation and morphological realisation. Linearisation approaches take a bag of words as input, with optional constraints, and the task is to obtain a valid ordering of the words to form a sentence [LZ15]. Linearisation tasks are generally solved by incorporating syntactic information [HWGL09], semantic information [PZS17] or by using language models [HST+17]. Approaches incorporating syntactic information can further be categorised into full-tree linearisation [ZC15,HWGL09] or partial-tree linearisation [Zha13] depending on the amount of syntactic information used. Liu at al. [LZCQ15] proposed a syntactic linearisation approach, which augmented a transition based system for dependency parsing [ZN11] so as to perform word ordering in addition to the parsing. Similarly, Horvat and Byrne [HB14] proposed a language model based syntactic linearisation approach, where a graph is constructed with the words in the input bag of words forming the vertices in the graph. The edge weights represent the language model probability between the words. The problem is then formulated as solving the (Asymmetric) Travelling Salesman Problem [ABCC06]. In language model based linearisation approaches purely distributional information is used for the task. Recently, the language model based linearisation approaches have shown to outperform syntax based linearisation approaches in English [SRS16, WR16]. While Schmaltz et al. [SRS16] proposed a neural language model with a beam decoder, Wiseman and Rush [WR16] proposed a Seq2Seq model with a margin level loss that penalises the function when the gold sequence falls off the beam during training. In Sanskrit, similar to Wiseman and Rush [WR16], Krishna et al. [KSS+19] proposed a Seq2Seq model for syntactic linearisation in Sanskrit.[8] The sequence level model, along with two of its accompanying pretraining steps, has outperformed other linearisation models, originally proposed for English.

In the case of poetry linearisation, a task which we introduce in this thesis, a bag of words has to be arranged into a verse ordering. Here, the ordering so obtained has to adhere to one of the prescribed metre patterns as per Sanskrit Prosody. The identification of metre in a verse in Sanskrit and its syllable level alignment is a deterministic process, primarily due to the existence of an advanced discipline of Prosody right from the classical era (circa 300 CE) [SASG15]. This need not be true for metre identification tasks in general [MS18]. Melnad et al. [MGS13] proposed a metre identification tool for Sanskrit verses. Recently, Rajagopalan [Raj] extended the framework and released it as an online tool. Nevertheless, arranging a bag

[8] This model is elaborated in Section 4.3

of words into a valid metre pattern, similar to syntactic linearisation, is a computationally challenging task due to the combinatorial explosion of the search space. The search space essentially consists of the possible permutations of the words in the sentence. We propose a solution for the problem using the structured prediction framework we elaborate in Chapter 5.

2.6 Dependency Parsing and Joint Modelling of Morphosyntactic Tasks

Transition based approaches [KMN09] and graph based approaches [MPRH05] are primarily two approaches adopted for data-driven dependency parsing. Easy-First parsing approach [GE10] can be seen as a third alternative where the easy decisions related to identifying dependencies are made first and the partial structures so obtained are incorporated for deciding on the harder decisions [Mor16]. This approach does not follow a sequential processing of the input. Similar to graph based dependency parsing approaches, this enables consideration of dependencies between far placed words [Mor16], but at a time complexity of $O(nlogn)$, a notable improvement in running time as compared to graph based approaches. Graph based parsers are globally optimised parsers, which parameterise over the substructures of a dependency tree. McDonald et al. [MPRH05] proposed an arc-factored approach for non-protective parsing. It is shown that incorporating higher order features will result in NP hard complexity for non-projective dependency parsing using graph based parsing approaches [MS07].

Majority of the methods proposed for joint morphosyntactic parsing also fall into either transition based or graph based approaches. Tsarfaty [Tsa06] observed that jointly solving downstream tasks such as morphological parsing and dependency parsing is beneficial as compared to performing these in a pipeline. Such approaches have proven to be effective for a host of morphologically rich languages such as Hebrew, Turkish, Czech, Finnish, Arabic, etc. [CS07, GT08, BNB+13]. Hatori et al. [HMMT12] formulated the problem of joint word segmentation, POS tagging and dependency parsing in Chinese using a transition based framework. Seeker and Çetinoğlu [SÇ15] performed the joint morphological segmentation and analysis along with dependency parsing for Hebrew and Turkish. The approach con-

structs a sentence level graph with word level morphological lattices and performs a dual decomposition wherein the predicted dependency tree and the morphological paths need to be arrived at an agreement for the final solution. Recently, More et al. [MSBT19] proposed a transition based framework [ZC11], which encompasses a joint transition system, training objective and inference for the morphological processing and syntactic parsing tasks. The system outperforms their previous standalone transition based system for morphological analysis [MT16], emphasising on their benefits of joint morphosyntactic parsing.

For dependency parsing, Kiperwasser and Goldberg [KG16] proposed to replace the traditional feature engineering approach with vector representations of tokens obtained from BiLSTM based neural architectures. Several neural parsing models have extended this idea, including Kuncoro et al. [KBK$^+$17]; Dozat and Manning [DM17] and they report competitive results for languages such as English. Nevertheless, these models have shown to be of limited effectiveness as compared to feature engineered models for Morphologically rich languages [MSBT19]. In our framework, we do not use any hand-crafted features. Instead, we automate the learning of the feature function and formulate it as learning of horn clauses [LC10, GM15]. The task of obtaining the features for the edge vector is similar to obtaining a distributional composition between two words [WWRK16]. Our work stands close to the attempts such as algebraic formulation for feature extraction by Srikumar [Sri17] or the Monte Carlo Tree Search based feature selection approach by Gaudel and Sebag [GS10]. In Krishna et al. [KSB$^+$18], we used the Path ranking algorithm [LC10], a random walk based approach for learning horn clause across a heterogeneous information network (HIN). The type of horn clauses mentioned in PRA is widely known as metapaths [Sun10] in HINs [SLZ$^+$17]. Traditionally metapaths, like feature engineering, were manually constructed. But, recent approaches such as PRA and FSPG [MCM$^+$15] automate the generation of metapaths. In the structured prediction framework, we elaborate in Chapter 5, we use the FSPG approach for automatic learning of the feature function.

Chapter 3

Incorporating Linguistic Knowledge in Data-Driven Models for Sanskrit

3.1 Introduction

Explicit modelling and encoding of linguistic knowledge into data-driven approaches in NLP have shown to be of great value, especially in terms of performance and robustness of these systems [Tsv16]. In fact, morpho-syntactic parsers for morphologically rich languages such as Hebrew, Turkish etc. still rely on hand-crafted features for achieving their state of the art performances [MSBT19, MT16, SÇ15]. Similarly, encoding linguistic knowledge has shown to compensate for the resource-constraints often faced in low-resource language settings [ZLP+17, ZNR19]. Sanskrit is both a morphologically-rich as well as a low-resource language. Under this pretext, we investigate the interplay of linguistic knowledge and distributional information for three different tasks in Sanskrit. The three tasks we investigate in this chapter are semantic type identification of compounds, analysis of derived word-forms from nominal derivational morphology and word segmentation.

3.1.1 Morphological Derivation and Compounding

Word formation is a highly productive process in Sanskrit. While Pāṇini's grammar recognises a set of non-derivable base verbal roots and nominal stems, it clearly prescribes rules to accommodate for the productivity of the words in the language [Kip09]. To illustrate this, let us consider how Pāṇini (पाणिनि) describes a nominal in his grammar rule-book *Aṣṭādhyāyī* (अष्टाध्यायी). Here, a nominal is defined in two rules.

- *1-2-45 arthavat adhātuḥ apratyayaḥ prātipadikam* (अर्थवदधातुरप्रत्ययः प्रातिपदिकम्) : a word which is meaningful, but which is neither a verbal root nor an affix is called a nominal (prātipadikam).[1]
- *1-2-46 kṛttaddhitasamāsāh ca* (कृत्तद्धितसमासाश्च): A word which is generated by compounding of one or more nominals (samāsa) or derived by affixation of derivational affixes grouped under the labels of *kṛt* (कृत्) and *taddhita* (तद्धित) is also called as *prātipadikam* (प्रातिपदिकम्). The *kṛt* suffixes are applied to verbal roots to derive a nominal, while *tadddhita* affixes are applied to nominal stems to derive a nominal.

With this definition, Pāṇini's *Aṣṭādhyāyī* has left much to the judgement of the speaker, on what constitutes a nominal. However, to assist the speaker in the interpretation, *Aṣṭādhyāyī* well elaborates rules and conditions which mandate, and in some cases restrict, these word-formation operations. In fact, *Aṣṭādhyāyī* devotes about 1115 of the 4000 rules for the generation of derivational nouns using taddhita affixes. This makes it one of the most extensive thematic sections in the *Aṣṭādhyāyī* [Sha87]. Similarly, about 400 rules in *Aṣṭādhyāyī* deal with compounding and more importantly, Sanskrit corpora like DCS [Hel16a] contain about 75 % of its vocabulary as compounds. Given the extensive use of compounds and derivational nouns in Sanskrit, their analysis can significantly benefit further downstream tasks in a data-driven NLP pipeline [LMZB13, WCM+14]. Among the different categories of word-formation, we are primarily interested in the processes of compounding and morphological derivation using *taddhita* affixes. Both of these are tightly linked to lexical semantics [Lig96, Lie09]. Here, we integrate knowledge from rules in *Aṣṭādhyāyī*, external lexicons and distributional information as features for the tasks related

[1] The number '1-2-45' indicates the index of the rule as per *Aṣṭādhyāyī*. The rule is the 45th rule which appears in the first module, second quarter of *Aṣṭādhyāyī*

to these word-formation processes. The statistical models we introduce here are currently the state of the art models for their respective tasks.

Semantic Type Identification of Compounds: Compounding is a word-formation process that combines two or more free morphemes to form a new word. Pāṇini recognises four main classes of compounds, namely, adverbial, copulative, determinative and exocentric [Muñ13]. Our task is to classify a given compound into one of the four classes, and we formulate this as a multi-class classification problem. While we use three standard ensemble-based classifiers for the task, the novelty of the work is in its feature engineering. We construct an elaborate feature space for our classifiers by combining rules from the *Aṣṭādhyāyī*, semantic relations between the compound components from a lexical database Amarakośa (अमरकोश) [NK10] and statistically discriminative subword units from the compound components obtained using Adaptor Grammar [JGG06]. We then analyse the impact of each type of features in the performance of the classifiers. We find that the use of rules from *Aṣṭādhyāyī* along with other linguistically motivated features, such as morphological patterns and lexical lists result in an overall accuracy of 59.34 % for our best performing Random forest based classifier. The performance further improves when the feature space was augmented with lexical relations, between the compound components, from Amarakośa (63.81 %) and the subword units from Adaptor grammar (75. 08 %).

Analysis of Derivational Nouns from Nominal Derivational Morphology: Derivational affixes are non-meaning preserving affixes, through which a new word is formed from another. The word so formed undergoes a semantic shift from its source word based on the semantic sense the affix carries [Deo07]. Consider the word 'Indianness', derived from 'Indian', where the affix '-ness' denotes 'the state of being'. In fact, Indian is derived from India, where the affix '-ian' denotes provenance [Deo07]. Here, the words Indian and Indianness were always potent to be in existence, right from the instant when the word India came into existence. Hence, derivational morphology should be understood as a theory of possible words [Hal73], which can capture this form of productivity [O'D15]. We propose a semi-supervised transductive learning approach for the identification of derivational nouns in Sanskrit. We not only identify the derivational nouns but also link them to their corresponding source words. We specifically use, modified adsorption [TC09],

which is a graph based label propagation approach, for the task. The novelty of our work is primarily in its design of the network structure for the task. The edges in the network are formed based on the similarity of patterns arising out of string transductions between source stem and their derived words. The edge weights are then featurised based on the phonetic, morphological, syntactic and the semantic similarity shared between the words to be identified. For obtaining these similarities, we rely on rules from *Aṣṭādhyāyī*, subword units learned using Adaptor grammar and word2vec embeddings [MSC+13]. We find that our model is effective for the task, even in the cases where we employ a labelled dataset which amounts to only 5 % to that of the entire dataset.

3.1.2 Word Segmentation

Identifying word boundaries in a sentence in Sanskrit is a challenging task. The writings in Sanskrit follow a phonemic orthography, where the words in the sentence often undergo phonetic transformations. This euphonic assimilation of phones is formalised as *Sandhi* in Sanskrit Phonetics [GH16]. *Sandhi* occurs at the juncture of two independent words as well as between the stems of a compound-word [HN18]. In the case of compounds, performing the Sandhi operation, whenever applicable, is mandatory. However, Sandhi between independent words is optional and left to the discretion of the speaker of the sentence. This results in sentences where the word boundaries are not only obscured but often the phones at these boundaries get modified as well. Analysis of such a fused sentence may lead to ambiguity, as there can be multiple possible word segmentation analyses for it. Our task here is to predict the semantically most valid word segmentation for a given fused sentence.

We introduce a new segmentation model for Sanskrit, which similar to Hellwig [Hel15b], jointly performs the tasks of compound splitting and word segmentation. In addition to obtaining the word boundaries, we also identify the individual nominal stems in a compound word. Here, we integrate linguistic information into the model, in terms of constructing the search space as well as in feature engineering. We first use a lexicon driven shallow parser [GH16] to exhaustively enumerate all the phonetically valid candidate segmentations, i.e. all the candidate segmentations adhering to *Sandhi*. This forms the search space, and we treat the problem as an iterative query expansion problem. Here at every iteration a candi-

date is greedily chosen, where the chosen candidate has the highest aggregated association score with the candidates that already are part of the partial solution. The association score between the candidate words is captured using linguistically motivated hand-crafted feature templates. These templates make use of morphological and syntactic properties based on the language's grammar and act as constraints for improved candidate selection. Path Constrained Random Walks (PCRW) [LC10] framework is employed for feature generation, where the generated features fit into one of the templates defined for the task.

3.2 Compound Type Identification in Sanskrit

Compounding is a productive process of word-formation in languages where two or more nominals are used together to generate a new nominal. Compound-analysis is computationally challenging, primarily due to three factors: i). Compounds are highly productive in nature, ii). The relation between the components is implicit and iii). The correct interpretation of a compound is often dependent on contextual or pragmatic features [KB05]. For example, 'houseboat' and 'boathouse' are compounds formed from the same pair of nominals, 'house' and 'boat', but do not mean the same.[2] Similarly, the relation between 'olive' and 'oil' in 'olive oil' does not hold between 'baby' and 'oil' in 'baby oil'. Identifying the head of a compound can lead to significant improvements in semantic analysis tasks like Machine Translation, Question Answering, etc. [WCM+14, Tie05]. The head of a compound, in general, is indicative of the referent(s) of the compound. It also determines the syntactic properties of the compound. For example, in 'paleface' paraphrased as 'a person who has a pale face', the head of the compound is an external entity. Here a word to word translation of the components would yield undesirable results. In 'bittersweet', both the stems 'bitter' and 'sweet' are the heads of the compound. In both 'houseboat' and 'boathouse', the final component forms the head.

On our empirical investigation of the Digital Corpus of Sanskrit (DCS)[3], we find plenteous use of compounds in the sentence constructions. While DCS contains about 2.5 million tokens, about 373,000 of them are compounds. This is almost double in com-

[2] http://wikidiff.com/houseboat/boathouse

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

parison to languages like German, which report 5.5-7% of corpus presence of the compounds [Sch05, BMT02b]. In DCS, 75% of the vocabulary consists of compounds, as against 47% vocabulary share [BMT02b] of compounds in German. We also find that 43% of the 66,000 lemmas in the corpus vocabulary were part of the compound formation as compared to 3-4% in English [SC13]. Given the extensive use of compounds in Sanskrit, We propose a multiclass classifier to identify the semantic type of Sanskrit compounds. *Aṣṭādhyāyī*, an ancient grammar treatise on Sanskrit, discusses the generation of four broad classes of compounds, namely, *Avyayībhāva* (अव्ययीभाव), *Tatpuruṣa* (तत्पुरुष), *Bahuvrīhi* (बहुव्रीहि) and *Dvandva* (द्वन्द्व). Our classifier is trained to classify a given compound into one of these four classes. We find that the aforementioned notion of 'head' in compounds is discriminative as per this categorisation. For our classification task, we successfully combine features extracted from rules in *Aṣṭādhyāyī*, taxonomy information and semantic relations inferred from *Amarakośa* (अमरकोश) ontology [NK10], and class-specific sub-word units learned using Adaptor grammar [JGG06]. We perform an in-depth analysis of the performance of the system and highlight where the existing rules of *Aṣṭādhyāyī*, a generative grammar, are inept in classifying the data samples. Further, we show how the additional features we incorporate, help us improve the performance of the classifier.

3.2.1 Compounds in Sanskrit

Compounds in Sanskrit are concatenative in nature, with a strict preference for the ordering of the components [Gil91]. A generated compound is treated as a fully qualified word (*pada*), such that the compound is subject to all the inflectional and derivational modifications applicable to nominals. Affixation occurs at the end of the compound, similar to compounds in languages like Greek, and not within the components [ZvdP16, Gil91]. Any compound can be analysed by decomposing it into two immediate component nouns. Linguists in Sanskrit have deeply analysed the aforementioned regularities and proposed different categorisations and further sub-categorisations of the compound types [KK13, Gil09]. Here, we consider the four broad categorisations proposed for compounds, namely, *Avyayībhāva*, *Tatpuruṣa*, *Bahuvrīhi* and *Dvandva*. We now explain these four classes of compounds and discuss various discriminative aspects about them that we can capture from the generated word-forms and use in our system.

1. *Avyayībhāva* (अव्ययीभाव) Compounds - In *Avyayībhāva* compound, the first component is an indeclinable (*avyaya*), which generally forms the head of the compound. The compound so generated will also become an indeclinable. For instance, in *'upakṛṣṇam'* (उपकृष्णं, near to *Kṛṣṇa*), the word '*upa*' (उप, near) is an indeclinable and the second component '*kṛṣṇa*' bears an inflectional affix. However the compound becomes an indeclinable.
2. *Tatpuruṣa* (तत्पुरुष) Compounds or Determinative Compounds - They are endocentric compounds in which the second component is generally the head of the entire compound. For example, the phrase '*rājñaḥ puruṣaḥ*' (राज्ञः पुरुषः, King's man) yields *rājapuruṣaḥ* (राजपुरुषः). The second component, '*puruṣaḥ*' forms the head in the canonical paraphrase and hence the head of the compound [Gil91]. The relation between the components is marked by the genitive case inflection of the first component *rājñaḥ*. *Tatpuruṣa* compounds constitute a distinctive sub-categorisation, namely, Descriptive compounds or *Karmadhāraya* (कर्मधारय). In *karmadhāraya* compounds, the first component generally acts as an adjective to the final component. For example, the first component in '*nīlameghaḥ*' (नीलमेघः, blue cloud), '*nīla*' (blue), is qualifying the head word '*megha*' (cloud).
3. *Bahuvrīhi* (बहुव्रीहि) Compounds or Exocentric Compounds - When the components in the compound refer to some external entity, say a person or an object, we call it a *Bahuvrīhi* compound. Here, the referent of the compound becomes the head of the compound. For example, '*pītāmbaraḥ*' (पीताम्बरः) is paraphrased as '*pītam ambaraṃ yasya saḥ* (पीतम् अम्बरम् यस्य सः)'. Here the words *pītam* (yellow) and *ambaram* (cloth) together form the compound referring to the Lord Vishnu. In absence of the paraphrase, the referent often needs to be inferred from the context in which the compound is used. However, the gender differences between the final component and that of the compound is a convenient heuristic that can be used to identify the compound type in some of the cases [GH13].
4. Dvandva (द्वन्द्व) or Copulative Compounds - They are conjunctive compounds where the components are compounded to show the collectiveness of the individuals. The components involved may be nouns or adjectives. Since the components share a conjunctive relation, often multiple components are compounded together in a single process. In *Dvandva* compounds, the compound generally assumes the gender of the final component. However, during copulative compound formation, deciding the final component can be tricky as all the words involved are equivalent with respect to the compounding process. For example, a *Dvandva* compound formed from the paraphrases '*pitā ca mātā*

ca (पिता च माता च)' and '*mātā ca pitā ca*' (mother and father) will always be of the form '*mātāpitarau*' (मातापितरौ), which is in masculine due to the masculine noun *pitā* (father), but will never be '*pitāmātarau*' (पितामातरौ), which should be in feminine gender. The formation of the latter is prohibited in the grammar, thereby eliminating the possibility of a conflict.

Compound type identification is a challenging task, even for humans. Resolving these challenges might require features ranging from morphological characteristics to pragmatics. It is especially tricky when the same pair of components can form compounds belonging to different semantic types. For example, *pītāmbaraḥ* (Lord Vishnu) and *pītāmbaram* (yellow cloth) are *Bahuvrīhi* and *Tatpuruṣa* compounds respectively, formed from the same components, *pīta* and *ambaram*. Here the gender of the compounds becomes a discriminative feature. Here, the stem '*ambara*' is in neuter and hence in *Tatpuruṣa* compounds, the compound also maintains the neuter gender. But, for *Bahuvrīhi* compounds, the gender is based on the referent, which in this case is masculine. Now, if we consider the case of compounds where the first component denotes a negation marker, there can be ambiguities between *Tatpuruṣa* and *Bahuvrīhi* classes. The specific sub-categories are called as *Nañ-Tatpuruṣa* and *Nañ-Bahuvrīhi* compounds, respectively. For instance, the compound '*aputraḥ*' (अपुत्रः) is paraphrased as '*na putraḥ*'(न पुत्रः, not a son) in the case of *Tatpuruṣa* and '*avidyamānaḥ putraḥ yasya saḥ*'(अविद्यमानः पुत्रः यस्य सः, having no son) in the case of *Bahuvrīhi*. Similarly, consider the case of ambiguity between *Tatpuruṣa* compounds and *Avyayībhāva* compounds as well. While a compound with its first component as an indeclinable generally belongs to the *Avyayībhāva* class, there exists a sub-categorisation of *Tatpuruṣa*, where the first component is an indeclinable. For example in '*ativanam*' (अतिवनम्), the compound consists of two components viz '*ati*' and '*vanam*'. Here the first component '*ati*' is an indeclinable, a strong characteristic of *Avyayībhāva* compounds. The paraphrase of '*ativanam*' in the case of *Avyayībhāva* is '*vanasya atyayaḥ*' (past the forest) and '*vanam atikrāntaḥ*' (वनम् अतिक्रान्तः, having passed the forest) in the case of *Tatpuruṣa*. The intention behind the usage, i.e. the implicit relation between the components, needs to be inferred to resolve such issues. But this is a challenge in itself, in fact a harder one, and hence we do not address it in this work.

3.2.2 Method

In our current work, we treat the problem as follows. When given a compound word decomposed into two immediate components of the compound, we identify the semantic type of the given compound and classify it into one of the four classes as discussed in Section 3.2.1. We build a feature-rich multi-class classifier and analyse the effectiveness of the features for the classification task. In *Aṣṭādhyāyī*, the generation of a compound is assumed to start with the canonical paraphrase of the compound. The noun declensions, modifiers and relation words in the paraphrase are then elided to form the compound. In our current settings, we only consider the compound and its individual split components. In this section, we describe the various features used for this classification task.

Rules from *Aṣṭādhyāyī* Kulkarni and Kumar [KK13] provides a categorisation of the rules in *Aṣṭādhyāyī* which are useful for compound analysis. Table 3.1 provides a summary of the type of rules that we employ in our system. The type 1 rules are lexical lists which contain lists of nouns and indeclinables that appear as a component in the compound. Type 2 considers the morphological properties of the components. Due to syncretism prevalent in the inflectional morphology of Sanskrit and non-availability of analysers for derivational morphology in Sanskrit, we settle for utilising the string patterns present in the component words as features for the type 2 rules. The last two rule types are semantic in nature. Rule type 3, i.e., rules that check for the semantic property of the component, is captured using manually curated lists of lexicons such as list of rivers, parts of day and night, etc. It essentially contains word lists stated outside of *Aṣṭādhyāyī*. The last type of rule looks into the possible relations between the components, where we utilise the lexical database *Amarakośa*.

Relations from Lexicons Lexical databases with annotated semantic networks are beneficial in identifying the semantic compatibility between individual nouns and hence can be used in compound analysis [KB05, SC13]. We utilise '*Amarakośa*', an ancient dictionary which covers about 11580 words (9031 unique lemmas) in altogether 4035 *synsets*. With efforts from Nair and Kulkarni [NK10], *Amarakośa* is digitised, forming a semantic network labelled with semantic relations between the words. The lexicon primarily consists of six relations, of which three of the relations, namely, 'part-whole','is a kind of' and 'master-

Rule Type	Rule	Example
Type 1: Lexical lists	*Aṣṭādhyāyī* rules like A.2.1.40 enlist specific lists of stems to be used as a component in compound formation	**akṣaśauṇḍaḥ** (अक्ष-शौण्डः, skilled in dice) - *śauṇḍa* (skilled) is listed in the rule A.2.1.40
Type 2: Morphological Properties	Rules like A.2.1.25 use inflectional suffix, derivational suffix etc. of the components in paraphrase as conditions for compounding	*kṛta* (कृत, Done) in the compound **svayamkṛta** (स्वयम्कृत, done by one's self) bears a derivational suffix *ta*.
Type 3: Semantic property of the component	Rules like A.2.1.45 state specific properties of objects as conditions for compounding, e.g., part of day.	Stem *pūrvāhṇa* (forenoon) in **pūrvāhṇakṛta** (पूर्वाह्ण-कृत, done in the morning) denotes a part of day.
Type 4: Semantic relations between the components	Rules like A.2.1.57 check for specific relations between the components, e.g., Modifier - Modified relation	**nīlotpalam** (नीलोत्पलम्) - *nīla* (blue) describes the second component utpalam (lotus).

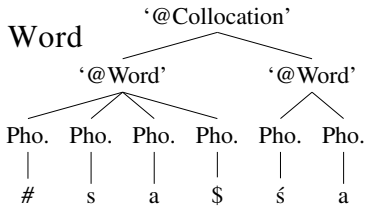
Table 3.1 Various rule types in *Aṣṭādhyāyī* for compound analysis [KK13]. A.2.1.40 etc. indicate the rule numbers in the book.

possession', are useful in identifying *Tatpuruṣa* compounds. Two of the three remaining relations, namely, 'child-parent' and 'husband-wife', are helpful in identifying *Dvandva* compounds. An additional advantage with *Amarakośa* is that we get gender information about the individual nouns from the e-lexicon, which is a discriminative factor in identifying *Bahuvrīhi* compounds as mentioned in Section 3.2.1. For each component, the gender, headword and the corresponding word with which the component bears the relation, are used as features. We consider all the six relations in *Amarakośa* between the compound components.

Variable Length Character N-grams from Adaptor Grammar We capture discriminative string patterns present in our dataset as variable length character n-grams, obtained using Adaptor grammar [JGG06]. It is a non-parametric Bayesian approach for learning probabilistic context-free grammar (PCFG) from data. When provided with data and a skeletal grammar, it learns the productions and the probabilities associated with these productions to form PCFG. A skeletal grammar, as shown in Listing 3.1a, typically provides the structure of the grammar to be learned. It involves the set of nonterminals and the production rules which capture the interactions between these nonterminals. However, for a subset of these nonterminals, the productions are learned from the data. We call such nonterminals as adapted nonterminals; these are marked with an '@' symbol in Listing 3.1a. The grammar learns a distribution over trees rooted at each of the adapted nonterminal [ZBGC14]. The productions so learnt in our case are variable-length character n-grams from the compounds data.

```
'@Collocation' > Word Collocation | Word
'@Word'  > Phoneme Word | Phoneme
Phoneme  > {Sanskrit Alphabet,
                        $, #}
```

(a)



(b)

Listing 3.1 a) Skeletal grammar for the adaptor grammar [JGG06]. b) Derivation tree for an instance of a production '#sa$ śa' for the nonterminal '@Collocation'

Listing 3.1b shows the derivation tree for one such string '#sa$ śa'. The input to the grammar is basically a sequence of two components with a '$' marker as the delimiter between them. Also, the input sequence begins and ends with a '#' marker indicating the beginning of the first component and end of the second component, respectively. Now, the string '#sa$ śa' is an outcome of two '@Word' productions, which is a recursive '@Collocation' production. Here since the first '@Word' production ends with a '$' symbol, it is clear that the string in the first production should be part of the first component and the string in the second '@Word' production should be part of the second component in the compound. Here, the first component should exactly be 'sa' and nothing else. For the particular pattern, we had 31 compounds in our training data, of which 22 of them belonged to *Bahuvrīhi*. 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.
We learn three separate grammars, G1, G2 and G3 using the same skeletal structure in Listing 3.1a, but with different data samples belonging to *Tatpuruṣa*, *Bahuvrīhi* and *Dvandva*, respectively. Each grammar is instrumental in identifying discriminative string patterns for each of these classes. Further we learn a fourth grammar G4, which contains samples from all the compound classes as well as random word pairs, where none of the words appeared as a compound component in a large Sanskrit Corpus.[4] Every production in the learned grammars has a probability to be invoked, where the likelihood of all the productions of a nonterminal sums to one. To obtain discriminative productions from G1, G2 and G3, we find conditional entropy of these productions matching 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 it, both of them were kept at the value of three. Though we obtain discriminative string patterns using adaptor grammar, the model basically outputs PCFGs for each of these classes. As a means of intrinsic evaluation of these grammars, we compare the likelihood of predicting unseen compound words that belong to the class, with that of the words not belonging to the class. For *Avyayībhāva*, though tried learning a grammar, it assigned a similar likelihood to the words in the class as well as others. Here, we hypothesise that our training data for *Avyayībhāva* may not be sufficient for effective generalisation of the grammar. Hence, we do not learn a grammar for *Avyayībhāva*.

Other Features We look for specific patterns that check for the lexical similarity between components. For example, consider the compound *bhāvābhāvau* (भावाभावौ) where the final component *a-bhāva* (अभाव) is the negation for the first component *bhāva* (भाव). The prepositions '*a*' and '*an*' represent negation of entities. We identify those compounds, where the first and second components differ only by *a* or *an*. This heuristic has its limitations, as these markers do not mark all the cases of negations. We also use Jaro-Winkler distance, an edit distance variant, between both the components as an additional feature to capture the lexical similarity between the components. We find that the mean Jaro-Winkler distance between components of *Dvandva* compounds (0.48) is higher than that of other compounds (0.31 - 0.38). We also consider the last three characters of the second component, where the second compo-

[4]We use the Digitial Corpus of Sanskrit as the corpus here.

Word	Position	Compound Class
iti	First	*Bahuvrīhi*
sva	First	*Tatpuruṣa*
manāḥ	Final	*Bahuvrīhi*
mātā	First	*Dvandva*
dharmā	Final	*Bahuvrīhi*

Table 3.2 Sample of filtered words and their position in the compound.

Classifiers	P	R	F	A
Random Forests	0.76	0.75	0.74	0.74
Extreme Random Forests (ERF)	0.76	0.75	0.74	0.75
Gradient Boosting Methods (GBM)	0.62	0.54	0.53	0.54
Adaboost Classifier	0.71	0.69	0.69	0.69

Table 3.3 Precision (P), Recall (R), F-Score (F) & Accuracy (A) for the competing systems on held-out dataset.

nent bears the nominal inflections of the compound word. We also used a handful of specific suffix patterns filtered based on the entropy score of the patterns in discriminating the classes. The patterns are indicative of the affix information. We finally filtered 34 words and patterns by manual inspection, that originally had lower entropy scores than our thresholds, but have a linguistic motivation for their inclusion. Table 3.2 shows a sample of such filtered words.

3.2.3 Experiments

Dataset We obtained a labelled dataset of compounds and the decomposed pairs of components from the Sanskrit studies department, UoHyd.[5] The dataset contains more than 32,000 unique compounds. The compounds were obtained from ancient digitised texts including *Śrīmad Bhagavad Gītā* (श्रीमद् भगवद् गीता) and *Caraka Saṃhitā* (चरक संहिता) 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 4,000, to match with the second highest class, *Bahuvrīhi*. We find that the *Avyayībhāva* compounds are severely under-represented in the dataset, with about 5 % of the *Bahuvrīhi* class. From the dataset, we filtered 9,952 different data-points split into 7,957 data points for training and the remaining as held-out dataset. For all the features mentioned in Section 3.2.2, we have considered data points which are in the training set and we have not considered data from the held-out in calculating any of the features, including Adaptor grammar.

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

Class (C)	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

(a)

C	P	R	F
A	0.85	0.48	0.61
B	0.84	0.76	0.80
D	0.94	0.25	0.39
T	0.75	0.85	0.80

(b)

C	P	R	F
A	0.84	0.67	0.74
B	0.88	0.73	0.79
D	0.69	0.61	0.65
T	0.72	0.87	0.79

(c)

Table 3.4 Classwise Precision (P), Recall (R) and F-Score (F) results for three different setups. a) On held-out data (Accuracy - 0.75). b) With 10-fold cross validation over training data (Accuracy - 0.79). c) Easy ensemble on held-out data (Accuracy - 0.77). A, B, D and T represent the classes *Avyayībhāva*, *Bahuvrīhi*, *Dvandva* and *Tatpuruṣa* respectively.

Results Probably due to a large feature space of 2737 features we employ, and an imbalanced dataset, the performance of the classifier models like SVM and decision tree were near to chance with SVM making no predictions to the *Avyayībhāva* class. We use ensemble-based approaches for our system, and the results are presented in Table 3.3. The results presented in the table are predictions over held-out data, where the classifier was trained with the entire training data. We find that the Extreme Random Forests (ERF) [GEW06,PVG+11] gives the best performance amongst all the compared systems in Table 3.3. The performance of the Random Forests and the ERF were almost similar with reported performance measures varying only from the third decimal point. Table 3.4b shows the result for the ERF classifier over training data when trained with 10 fold cross validation. The class-wise precision and recall for the model over held out dataset are presented in Table 3.4a. We find that the classifier fares poorly for *Avyayībhāva* and *Dvandva*, primarily due to sparsity in the data as they both amount to about 5% and 33% of the other two classes, respectively. To measure the impact of different types of features we have incorporated, we train the classifier incrementally with different feature types as reported in Section 3.2.2. We report the results over the held-out data. At first, we train the system with only *Aṣṭādhyāyī* rules and features discussed in the 'other features' section in 3.2.2. We find that the overall accuracy of the system is about 59.34%. We do not report the accuracy of the system when we use *Aṣṭādhyāyī* rules alone as it was not sufficient to cover all the samples. Then we augmented the classifier by adding features from *Amarakośa*. We find that the overall accuracy of the system has increased to 63.81%. Notably, the precision for *Dvandva* and *Bahuvrīhi* increased by absolute values 0.15 and 0.06, respectively. We then add the

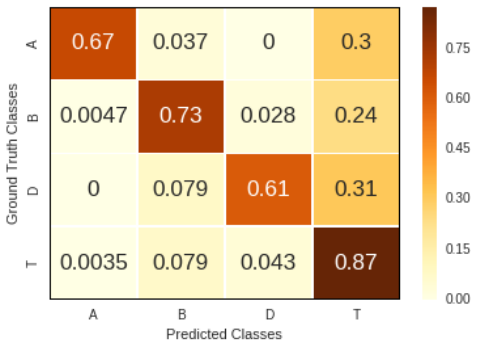


Figure 3.1 Confusion-matrix heat map for the easy-ensemble classifier

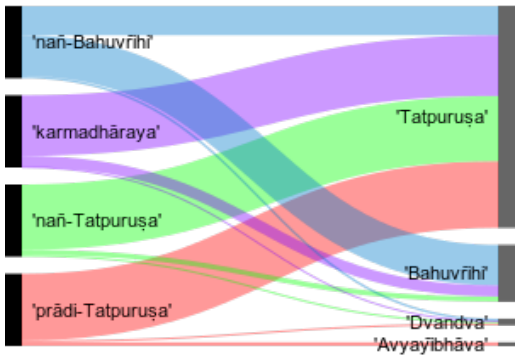


Figure 3.2 Alluvial graph showing the classification outcome for specific sub-classes

Adaptor grammar features to the feature set and Table 3.4a presents the result of the system with the entire feature set. We perform feature ranking based on entropy measure. We try to iteratively drop the least ranked features in steps of 50, till 1,700 of 2,737 features are dropped. We find that the accuracy does not change much, but mostly drops from the reported accuracy of 0.75 by a maximum of 0.38% (0.747).

To handle the imbalanced data set, we employed the easy-ensemble [LWZ09] approach. In the easy-ensemble approach, we form multiple training sets, each of which is a subset of the original training set such that the data samples in the minority classes remain intact whereas the majority classes are under-sampled to a defined proportion. In effect, we have multiple training sets where the data samples in the majority class are distributed across the subsets. Now with each of the subset, we run ERF classifier and average out the results. As can be seen from Table 3.4c, this approach gives consistent results across the four classes, with significant improvements in F-Score for *Dvandva* and *Avyayībhāva* classes.

We further look into specific cases of compound classes which get misclassified. Figure 3.1 shows the confusion matrix heat-map for our best performing system, the easy ensemble classifier. From the heat-map we can observe that most of the misclassifications go to *Tatpuruṣa*, resulting in a lower precision of 0.72 for *Tatpuruṣa*. It can also be noted that there are no misclassifications between compounds of the classes *Dvandva* and *Avyayībhāva* . Figure 3.2 represents the classification of the specific cases of sub-types as discussed in Section 3.2.1. *Avyayībhāva* and *Tatpuruṣa* can potentially be conflicting as there exist specific types of *Tatpuruṣa* where the first component can be an *avyaya*. Of all the *Tatpuruṣa* compounds that got mispredicted as *Avyayībhāva*, though just 6 in number, all of them had an *avyaya* as their first

component. Similarly, out of 70 misclassifications of *Tatpuruṣa* compounds to *Bahuvrīhi*, 38 belong to *karmadhāraya* class. *Nañ-Bahuvrīhi* and *Nañ-Tatpuruṣa* are also potentially conflicting cases, and we find that in *Nañ-Tatpuruṣa*, 8 of the 11 misclassifications happen to *Bahuvrīhi* class and in *Nañ-Bahuvrīhi* 13 of the 14 misclassifications happen to *Tatpuruṣa* class. But in all the aforementioned cases, the majority of the data samples got correctly classified. Summarily, we built an automated classifier for identifying the semantic type of a compound in Sanskrit. With an ensemble based classifier approach, we tackle the challenge of an imbalanced dataset, and our system effectively classifies data into appropriate semantic classes. We successfully incorporate rules from the ancient grammar treatise *Aṣṭādhyāyī*, lexical relations from *Amarakośa* and we also learn linguistic structures from the data using adaptor grammars. In our work, we show the improvement in performance after incorporating each of the aforementioned feature types. We also analyse and discuss the specific cases of conflicts between these semantic types that lead to erroneous predictions by the system.

3.3 A Graph Based Semi-Supervised Approach for Analysis of Derivational Nouns

Derivational affixes are a prevalent means of vocabulary expansion used in natural languages. They, unlike inflectional affixes, are non-meaning preserving affixes which induce a new word when applied to an existing word. The affixes carry one or more semantic senses that is passed onto the new derived word [Mar69]. Whenever a new word comes into existence in a language, all of its derived forms are potent to be part of the language's vocabulary as well. For example, if a new country is formed with a name *Nauratia*, an English speaker can infer the meaning for the word *nauratian* as "a person residing in Nauratia", in spite of never having heard the derived word previously. A human need not rely explicitly on the corpus usage of the derived word to infer its meaning, but it suffices to know only about the source word and the affix used. Similarly, it is desirable to identify a derived word and link it to its corresponding source word automatically. Further, this information can benefit in obtaining compositional representations for the derived-form from its source word, as it is often the case that the corpus evidence for all the possible derived words might be hard to come by [CS17, LMZB13].

Identifying derived words from a corpus is challenging. Usage of pattern matching

approaches in strings alone is often inept as they result in a large number of false positives. For example, while the word 'postal' is generated from 'post', the word 'canal' is not generated from 'can'. Also, these string matching approaches might skip identification of many valid source and derived word pairs, and result in a low recall. This is often due to the high variations in string transductions that occur during the derivation process, even for the same affix. Both 'postal' and 'minimal' are derived using the affix 'al', but the source word for postal is 'post', while the source word for minimal is 'minimum'. Soricut and Och [SO15], recently proposed an approach for analysis and induction of morphology in words using word embeddings. But, the authors find that their approach, though effective for inflectional affixes, has limitations with derivational affixes.

In this work, we propose a transductive learning approach for analysis of derivational nouns in Sanskrit. Currently, there exists no analyser for Sanskrit that deals with the derived words. This leads to issues with large scale processing of texts in Sanskrit, especially given its prevalence in the Sanskrit grammar tradition. We use the Modified Adsorption algorithm [TC09], a variant of the label propagation algorithm, for the task. Using a high-recall but low-precision pattern matching approach, we first obtain a set of potential derived and source pairs. These word pairs form the nodes in the graph. Using the transductions involved between these word pairs, we find other word pairs with such similar transductions between them to form the edges. We then effectively use a diverse set of features including rules from *Aṣṭādhyāyī*, subword units learnt from the data using Adaptor grammar [JGG06] and word embeddings for the candidate words using word2vec [MSC+13].

The novel contributions of our task are:

1. We propose a semi-supervised framework using Modified adsorption for identification of derived words and their corresponding source words for Sanskrit.

2. We are able to scale our approach onto unlabelled data by using a small set of labelled data. We find that our model is effective even under experimental settings where we use only 5 % of the entire dataset as the labelled data. In other words, we can label up to 20 times more data than the labelled data we have, and we perform a human evaluation to validate our claim on the unlabelled datasets.

3. By leveraging on the rules from *Aṣṭādhyāyī*, we not only find different pattern differences between the source and derived word pairs, but we also group patterns that are likely

to emerge from the same affixes. Currently, given a pattern, we can narrow down the possible affixes for a pair to a maximum of 4 candidates from a set of 137 possible affixes.

3.3.1 Challenges in Sanskrit Derivational Nouns

The class of affixes used for deriving a nominal from other nominals is known as *taddhita* affixes in Sanskrit grammar. *Taddhita* covers non-category changing derivations and can be recursive as well [Bha89]. The extent of string transductions involved in the derivation process is dependent on the affix used and can be quite varying. The transductions need not occur only at the word boundary but also occur at internal portions of a word. Table 3.5 illustrates some cases which are discussed here. In the case of '*upagu*', the derived word gets an internal change and forms '*aupagava*'. But in the case of '*daṇḍa*', '*daṇḍin*' is derived, where no internal modifications occur.

In Sanskrit, there are 137 affixes used in *taddhita*. The edit distance between the source and derived words due to the transductions during the derivation tends to vary from 1 to 6. For example, consider the word '*rāvaṇi*' derived from '*rāvaṇa*' (Table 3.5), where the edit distance between the words is just 1. But, '*Āśvalāyana*' derived from '*aśvala*' has an edit distance of 6. Since the possible variations that can be expected are quite high, this might lead to a large candidate space when we use the pattern based matching approaches for identifying potential pairs of source and derived words. Additionally, a number of affixes used in *taddhita* are used for other purposes as well. For example, *kṛdanta*, i.e. nouns derived from verbs, share some of these affixes with *taddhita*. In Table 3.5, '*stutya*', a *kṛt*, derived from '*stu*' follows similar pattern as with the derivation of '*dākṣiṇātya*', a *taddhita* word, derived from '*dakṣiṇā*'. Now, '*kālaśa*' is derived from '*kalaśa*', where only an internal change is visible. But the similar pattern between '*karaṇa*' (Instrument) and '*kāraṇa*' (Reason) is a mere coincidence and hence will be a false positive as a potential derived-source pair. For the set of 137 affixes, we identify about 400 unique patterns based on the string transductions observed.[6]

With our knowledge from *Aṣṭādhyāyī*, we can abstract out some of the regularities in these 400 patterns we observe, especially for those transductions happening at the internal portions of a word. We rely on two string transduction operations, explicitly mentioned in the Sanskrit grammar *Aṣṭādhyāyī* and of relevance to derivation.

[6]These patterns were obtained using manual inspection of derived-source word pairs from different sources including lexicons, dictionaries and *Aṣṭādhyāyī*.

- **Vṛddhi (वृद्धि)** - The sounds '*ā*', '*ai*' and '*au*' are designated as *vṛddhi*. During derivation, *vṛddhi* is often used as an operational rule. If the rule is applicable in a derivation, then the first occurring vowel in the source word stem gets replaced with one of the *vṛddhi* vowels. For instance, in the case of the derived word *aupagava*, '*u*' is the first vowel in its source word *upagu* and it got replaced with '*au*' during derivation. In Table 3.5, *upagu*, *pramukha*, *aśvala* and *kalaśa* are some *taddhita* words that show *vṛddhi* of its words.[7]
- **Guṇa (गुण)** - The sounds '*a*','*e*' and '*o*' are called as *guṇa*. Whenever the *guṇa* operation is invoked in *Aṣṭādhyāyī*, the mentioned vowels will be replaced in place of other vowels. While deriving '*aupagava*' from '*upagu*', the string takes the form '*aupagu a*' at some point. Here, the '*u*' gets converted to '*o*' by virtue of *guṇa*, thereby resulting in an intermediate form of '*aupago a*'. This step is crucial in the derivation process of the final form, *aupagava*.[8]

We define the character sequence which gets modified or eliminated from the source word during the derivation as 'source pattern' (*'sp'*), and the character pattern that appears in the derived word is termed as 'end-pattern' (*'ep'*). But here, we do not consider the transductions that occur by virtue of *Vṛddhi* and *Guṇa* into these patterns. For instance, consider the words '*tilya*' and '*prāmukhya*' derived from the source words '*tila*' and '*pramukha*', respectively, in Table 3.5. Here, we consider that both these cases have undergone the same transduction during derivation, i.e., the 'sp' that gets removed is '*a*' and the 'ep' that gets added is '*ya*'. The change of '*a*' to '*ā*' is abstracted under *Vṛddhi*. With this abstraction, which in principle is inspired by the traditional grammatical practices in Sanskrit, we narrow down the pattern variations to about 70 end-patterns (ep).

3.3.2 Method

We define our task over a dataset of a finite set of vocabulary $\mathcal{C}$. We enumerate all the possible 70 end-patterns as mentioned in Section 3.3.1, that can be applied on a source word. With the extracted patterns, we identify word pairs $wp_i = (w_j, w_k) \in \mathcal{C}^2$ and represent each such pair as a tuple $t_{wp_i} = \langle w_j, w_k, sp, ep, vṛddhi = T/F, guṇa = T/F, a_{wp_{i1}}, a_{wp_{i2}}, a_{wp_{i3}} \rangle$, where sp, ep are the source pattern in w_j and the end-pattern added to the derived word

[7]The operation is not exclusive to *taddhita* and occurs in other instances as well. *sṛ*, *kṛ* in Table 3.5 are some examples.

[8]For the complete derivation procedure of the derivational noun '*aupagava*' from *upagu* as prescribed in *Aṣṭādhyāyī*, please refer to *Table 1* in Krishna and Goyal [KG15].

Word	Derived Word	Type	ep
upagu (उपगु, Name of a person)	aupagava (औपगव, Male offspring of Upagu)	Taddhita	a
śiva (शिव, Name of a Hindu god)	śaiva (शैव, Male offspring of śiva)	Taddhita	a
rāvaṇa (रावण, A mythological character)	rāvaṇi (रावणि, Male offspring of Rāvaṇa)	Taddhita	i
tila (तिल, Sesame)	tilya (तिल्य, Which is beneficial to sesame)	Taddhita	ya
pramukha (प्रमुख, (Prominent)	prāmukhya (प्रामुख्य, Prominence)	Taddhita	ya
daṇḍa (दण्ड, Stick)	daṇḍin (दण्डिन्, One who carries stick)	Taddhita	in
sṛ (सृ, To go)	sārin (सारिन्, One who moves)	Kṛt	–
kṛ (कृ, To do)	kāraka (कारक, One who does)	Kṛt	–
aśva (अश्व , Horse)	aśvaka (अश्वक , bad horse)	Taddhita	ka
aśvala (अश्वल, Holy priest of King Janaka)	āśvalāyana (आश्वलायन, Male offspring of aśvala)	Taddhita	āyana
stu (स्तु, To praise)	stutya (स्तुत्य, Worthy of praise)	Kṛt	–
dakṣiṇā (दक्षिणा, South direction)	dākṣiṇātya (दाक्षिणात्य, Southern)	Taddhita	–
kalaśa (कलश, Pitcher)	kālaśa (कालश, Related to pitcher)	Taddhita	a
karaṇa (करण, Instrument)	kāraṇa (कारण, Reason)	Random	–

Table 3.5 Derivational nouns and their corresponding source words in Sanskrit. Additionally, possible cases of false positives that follow similar patterns to derivational nouns are provided as well.

w_k, respectively. Both $vṛddhi$ and $guṇa$ are binary variables and are set to T, if the corresponding operations are applicable and F otherwise. For each word-pair wp_i, we

obtain three vectors $\vec{a}_{wp_{i1}}, \vec{a}_{wp_{i2}}, \vec{a}_{wp_{i3}}$, which encode properties of the word pair based on the rules in *Aṣṭādhyāyī*, subword units identified by adaptor grammar [JGG06] and word embeddings from word2vec [MSC+13], respectively. Each of these are essentially different properties of the input word-pair, i.e. different views of the input, and are denoted by $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ respectively. For example, the word '*daṇḍin*', derived from '*daṇḍa*' can be represented as a tuple $\langle daṇḍa, daṇḍin, a, in, vṛddhi = F, guṇa = F, a_{wp_{i1}}, a_{wp_{i2}}, a_{wp_{i3}} \rangle$. With the extracted word-pairs, $W_{candidates} \subseteq C^2$, we propose a binary relevance model that trains a separate classifier for every unique end-pattern.

Modified Adsorption We use Modified Adsorption (MAD) algorithm, a graph based semi-supervised transductive learning approach, for our task [TC09]. Unlike other semi-supervised algorithms [ZG02, ZBL+], MAD allows us to design the network structure explicitly as required. Further, the objective function penalises the results when similar nodes are assigned with different labels. This setting helps us to use a minimal set of labelled nodes as seed nodes for supervision. At the same time, MAD enables to incorporate unlabelled nodes into the system and captures the structural and feature-wise regularities present in them. In MAD, every node has a label associated with it and is seen as a distribution of the labels rather than a binary assignment. The unlabelled nodes initially have no label assignments, but once the algorithm is executed, every node is updated with a distribution of the labels in the label space. In $MAD(G, V_{Seed})$, the algorithm inputs a graph structure $G(V, E, W)$ and additionally a set of nodes denoted by V_{seed} along with their label distribution as the seed nodes. Here, the set V_{seed} is a subset of the vertex set of the graph G, i.e. $V_{seed} \subseteq V$. The algorithm, after its run, outputs a label distribution, for every $v \in V$. For our setting, we find that $W_{candidates} = \mathcal{U} \cup \mathcal{S} \cup \mathcal{G}$, where $\mathcal{U}$ is the set of unlabelled nodes, $\mathcal{S}$ is the set of seed nodes used as labelled nodes for training and $\mathcal{G}$ is the set of gold nodes which is used as the test data for evaluation of the model.[9] For the system a node obtained from $\mathcal{U}$ and $\mathcal{G}$ are indistinguishable. Also, all three sets are mutually disjoint. For every end-pattern, ep_i, we construct a separate classifier, and we denote this as MAD_i. Let the set of all the nodes with the end pattern ep_i be denoted as V_i, where $V_i \subseteq W_{candidates}$. Each MAD_i instance is a sequential pipeline of 3 runs of the MAD algorithm, each of which we denote as $MAD_{i1}, MAD_{i2}, MAD_{i3}$. In each run, the vertex set V_i remains the same, but the edge set and the edge weights vary. In our approach, the network structure is influenced

[9] We follow the same naming conventions as Faruqui et al. [FMS16] wherever possible

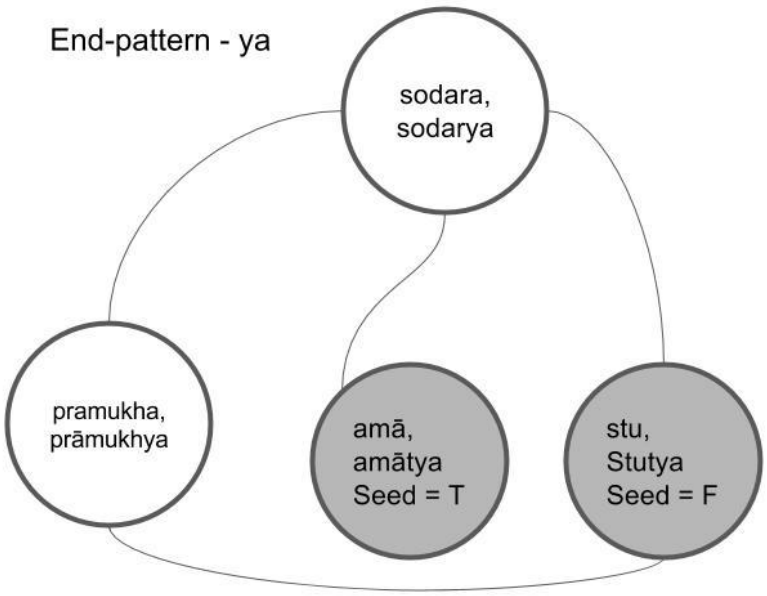


Figure 3.3 Graph structure for the end-pattern 'ya'. The nodes are possible candidate pairs in $W_{candidates}$. Nodes in grey denote seed nodes, where they are marked with their class label. The nodes in white are unlabelled nodes.

by the edge sets $\{E_{i1}, E_{i2}, E_{i3}\}$ and the corresponding weight sets $\{W_{i1}, W_{i2}, W_{i3}\}$, and both are decided by 3 different set of features $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. Further for an end pattern ep_i, we initialise its first MAD configuration in the pipeline as $MAD_{i1}(G_{i1}, \mathcal{S}_i)$. Here $\mathcal{S}_i$ is the set of seed nodes, along with their label distribution. In the next run, i.e. for MAD_{i2} we initialise it with the output label distribution from MAD_{i1}, and likewise MAD_{i3} is initialised with the output label distribution from MAD_{i2}. Figure 3.3 shows the graph structure G_{ik}, for the $ep_i = ya$. We provide our manually labelled seed set only for MAD_{i1}, i.e. at the first run. In MAD_i, the vertex set V_i remains same in all the runs of the pipeline and is essentially a set of all word pairs that follow a certain end-pattern ep_i. We essentially use rules from *Aṣṭādhyāyī*, subword units from Adaptor Grammar and the word vectors from word2vec as the features, and now we explain our feature engineering approach.

Phase 1: *Aṣṭādhyāyī* Rules

About 1115 rules of the 4000 in *Aṣṭādhyāyī*, i.e., more than 25 % of the rules, are devoted to affixation of derivational nouns. The rules related to *taddhita* either are string rewriting

Rule No	Rule	Semantic Relation	Source Word	Derived Word
4.1.95	ata iñ (अत इञ्)	Patronym	daśaratha	dāśarathi
4.1.112	śivādibhyo'ṇ (शिवादिभ्यः अण्)	Patronym	śiva	śaiva
4.1.128	catakāya airak (चटकाया ऐरक्)	Patronym	catakā	cātakaira
4.2.16	saṃskṛtaṃ bhakṣāḥ (संस्कृतं भक्षाः)	Processed	kalaśa	kālaśāḥ

Table 3.6 Conditional rules related to selection of suitable affix for derivational nouns from *Aṣṭādhyāyī*.

rules, conditional rules, or attribute assignment rules [KG15]. Table 3.6 illustrates some of the rules related to *taddhita*, the sense they carry and effect on source word due to the affixation. We consider only the conditional rules used by Pāṇini for the task, which can further be sub-categorised as given below.

1. Phonological and phonemic - Pāṇini uses the presence of certain phonological or phonemic entity in the source word as a condition for affixation. For example, the rule '*A.4.1.95 - ata iñ*', states that a lemma ending in '*a*' will be given the affix '*iñ*' when the affix is used to denote the sense of patronymy.
2. Morphological and lexical properties - Pāṇini incorporates a predefined set of lexical lists like gaṇapāṭha where words that are suitable for similar affixal treatment are grouped together. For example, the rule 'A 4.1.112' in Table 3.6, states to apply the affix '*aṇ*' to all the words in the lexical list headed by '*Śiva*'.
3. Semantic and pragmatic - *Aṣṭādhyāyī*, which was intended for human usage relies on semantic and pragmatic conditions as well. We use additional lexical lists instead of the semantic and pragmatic aspects for the purpose. For example, the rule 'A.4.2.16' applies to those words that signify 'food that is processed or prepared'. Here Pāṇini does not enumerate the list of such foods, but simply mentions the quality.

In Phase 1 we consider all the rules that deal with any of the phonological, phonemic, morphological and some of the semantic properties. Each rule is considered a separate feature at Phase 1 and the collection is represented as $\mathcal{A}_1$. From these rules in $\mathcal{A}_1$, we first identify the number of rules that are applicable for a given node $v_k \in V_i$ with the tuple t_k. This is encoded as a binary vector $\vec{a}_{k1}$, where a 1 indicates that the rule is applicable for

the node and 0 otherwise. Now as per Equation 3.1, the vertex score $\wp_k$ is essentially the number of rules in $\mathcal{A}_1$ that are applicable for the node v_k. Please note that $a_{k_1,l}$ denotes the l^{th} component of the vector $\vec{a}_{k_1}$. For a given node pair $v_k, v_j \in V_i$, we form an edge between them, if they have at least one rule in common. The edge weight is decided based on the number of common rules applicable to both of them and is normalised as per Equation 3.2. It needs to be understood that, here we consider the rules from *Aṣṭādhyāyī* merely as a means to measure the similarity between different pair of nodes. But a rule, which is applicable for a certain node, does not automatically guarantee that it will get applied in the derivation process of the derived-source word pair in the node. It is possible that a rule might get blocked by a stronger rule [PS92, Deo07].

$$\wp_k = \sum_{l=1}^{|\mathcal{A}_1|} a_{k_1,l} \qquad (3.1) \qquad W_{i1}^{v_k,v_j} = \frac{\sum_{l=1}^{|\mathcal{A}_1|} a_{k_1,l} \cdot a_{j_1,l}}{max(\wp_k, \wp_j)} \qquad (3.2)$$

$$E_{i1}^{v_k,v_j} = \begin{cases} 1 & W_{i1}^{v_k,v_j} > 0 \\ 0 & W_{i1}^{v_k,v_j} = 0 \end{cases} \qquad (3.3)$$

Phase 2: Character N-grams Similarity by Adaptor grammar

Pāṇini prized brevity in forming the rules for *Aṣṭādhyāyī*, as his grammar treatise was supposed to be memorised and recited orally by humans [Kip94]. 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 we could not encode in our phase 1 features, such as rules involving pragmatics in *Aṣṭādhyāyī*. In order to identify the regularities in the patterns in words, we follow the Adaptor grammar framework [JGG06], which we discussed in Section 3.2.2. Adaptor grammar is a non-parametric Bayesian approach for learning productions for a Probabilistic Context-Free Grammar (PCFG) from data. In Listing 3.2, 'Word' and 'Stem' are nonterminals, which are adapted. The nonterminal 'Suffix' consists of the set of various end-patterns. In this formalism, the grammar can only capture sequential aspects in the words. Hence the string modifications due to attributes like *vṛddhi*, need not be captured by the system as these modifications necessarily do not occur at the same locality as the rest of the string modifications.

$@Word \rightarrow Stem\ Suffix$
$@Word \rightarrow Stem$
$@Stem \rightarrow Chars$
$Suffix \rightarrow a|ya|.....|Ayana$
Listing 3.2: 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 nonzero, we find the sigmoid of the value so obtained and use it as the edge weight. Equation 3.4 uses the Iverson bracket [Knu92] to show the conditional sum operation. The equation essentially makes sure that the probabilities associated with only those character n-grams get 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.

$$\tau_{j,k} = \sum_{l=1}^{|\mathcal{A}_2|} a_{k_2,l}[a_{k_2,l} = a_{j_2,l}] \tag{3.4}$$

$$E_{i2}^{v_k,v_j} = \begin{cases} 1 & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases} \tag{3.5}$$

$$W_{i2}^{v_k,v_j} = \begin{cases} \sigma(\tau_{j,k}) & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases} \tag{3.6}$$

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

Phase 3: Semantic Word Vectors

In phase 3, we try to leverage the similarity between word embeddings [MSC+13] to propagate the labels. Due to limited resources at our disposal, we find it challenging to train word embeddings for Sanskrit. We resort to finding synonyms of words using the digitised version of Monier-Williams Sanskrit-English dictionary and then use the corresponding pre-trained English word vectors for the task. We find the word vectors only for the source words as the dictionary entries for derived words are even scarcer to obtain. We identify those nodes in the graph, for which there exists a word vector. We then compute the pairwise

cosine similarity between all such nodes in the graph, which form edge weights in this phase. We find that our graph structure G_{i3} for many end-patterns results in multiple disconnected components, as not all words in $W_{candidates}$ have an entry in the dictionary. We assign teleportation probability [BNG18] to every node in the graph in order to handle this issue.

3.3.3 Experiments

We explain the experimental settings and evaluation parameters for our model in this section.

Dataset

We use multiple lexicons and corpora to obtain our vocabulary $\mathcal{C}$. We use IndoWordNet [KDK$^+$10], the Digital Corpus of Sanskrit[10], a digitised version of the Monier Williams[11] Sanskrit-English dictionary, a digitised version of the Apte Sanskrit-Sanskrit Dictionary [GHK$^+$12] and we also utilise the lexicon employed in the Sanskrit Heritage Engine [GH16]. We obtained close to 170,000 unique word lemmas from the combined resources.

End-pattern	$W_{candidates}$	Seed $\mathcal{S}$	Gold Labels $\mathcal{G}$	Recall	Precision	Accuracy
a	2500	350	88	0.77	0.72	73.86
aka	1200	120	30	0.67	0.77	73.33
in	1656	270	68	0.82	0.74	76.47
ya	1566	258	64	0.72	0.7	70.31
i	1455	166	42	0.52	0.55	54.76
ika	803	122	30	0.6	0.69	66.67
tā	644	34	12	0.5	0.6	58.33
la	360	48	12	0.67	0.8	75
tva	303	22	12	0.67	0.8	75
īya	244	40	12	0.67	0.67	66.67
eya	181	34	12	0.83	0.71	75

Table 3.7 Recall (R), Precision (P) and Accuracy (A) for the candidate nodes evaluated on the gold labels.

Obtaining Ground Truth Data - For our classifier MAD, we obtain the seed labels $\mathcal{S}$ and the gold labels $\mathcal{G}$ from a digitised version of Apte Sanskrit-Sanskrit dictionary. The

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

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

dictionary has preserved the etymological information of the entries in the dictionary. For each end-pattern, we filtered out the pair of words which are related by *taddhita* affixes. Seed nodes for the negative class were obtained using candidate pairs which were either marked as *kṛdanta* words in the Apte Dictionary or were found in the dictionary, but are not related to each other. Additionally, we tagged several word pairs manually so as to obtain a balanced set of labels. We narrowed to 11 separate end-patterns for which we have at least 100 candidate pairs and have at least 5 % of word pairs as seed nodes in comparison to the size of the candidate pairs for the end-pattern. Table 3.7 shows the statistics related to each of the 11 end-patterns on which we have performed our experiments.

Baselines

We propose the following systems as the competing systems. We use label propagation [ZG02] as a strong baseline, and we also compare the output at each of the phases as separate baseline systems. Altogether we compare four systems as follows:

1. Label Propagation (LP_i) - We propose a label propagation based semi supervised classifier [PVG$^+$11] for each of the end-pattern. For each node, we find the top K similar nodes and assign edges to only those nodes, where K is a hyperparameter. The similarity is obtained from a feature vector that defines a node, with features from the first 2 phases incorporated into a single feature vector. We do not use the word embeddings from Phase 3 directly, but find the cosine similarity between the embeddings of the words and perform a weighted sum with the similarity obtained from the combined feature vector.
2. $MADB1_i$ - Here, we perform only one run of MAD, i.e. MAD_{i1}, and use features only from phase 1 as defined in Section 3.3.2.
3. $MADB2_i$ - Here, we perform only 2 sequential runs of MAD, i.e. MAD_{i1} and MAD_{i2}, and use the features from phase 1 in MAD_{i1} and features from phase 2 in MAD_{i2} as defined in Section 3.3.2.

Results

Table 3.7 shows the final results of our proposed system MAD_i, for each of the 11 end patterns. We report the Precision, Recall and Accuracy for each of the classifier w.r.t the true class. Our results are calculated based on the predictions over the test data in $\mathcal{G}$. Seven of eleven patterns have accuracy above 70 %. End-pattern '*i*' is reported to perform the worst

among the 11 patterns provided. We find that the average degree for G_{i1} for the pattern '*i*' is about 77.62, much higher than the macro average degree for G_{i1} for all the patterns, which is 43.86. This is the effect of some of the rules in *Aṣṭādhyāyī* with high coverage, applicable for the pattern 'i'. Here, we randomly down-sampled the number of neighbours to 44 to match with the macro average, and the accuracy increased to 61.9 %. Table 3.8 shows the results for the competing systems. We compare the performance of five end-patterns, selected based on the vertex set size V_{i1}. Our proposed system, MAD_i performs the best for all the five patterns. Interestingly, $MADB2_i$ is the second best-performing system in all the cases beating LP_i. For the pattern 'aka', the share of word vectors available was $< 10\%$ overall. So, in effect, only one of the false-positive nodes got the true negative label, after the third step is performed. Thus the recall remains the same after both the steps.

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. We find that for those 3 patterns ('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. We report the best result for each of the systems in Table 3.8.

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
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
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

Pattern	**System**	**P**	**R**	**A**
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
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

Table 3.8 Comparative performance of the four competing models.

Evaluation for Unlabelled Nodes

In order to evaluate the effectiveness of our system, we pick nodes from unlabelled set $\mathcal{U}$ and evaluate the word-pairs based on human evaluation. We take top 5 unlabeled nodes predicted as *taddhita* and top 3 unlabelled nodes predicted as not *taddhita* from each of the 11 end-patterns. We collate the predictions and divide them into 3 mutually exclusive sets of 22 entries each, as the remaining 22 of the original 88 were filtered out. Each set is evaluated by 3 experts and in case of a conflict, we go with the majority votes for each of the set. Seven experts, with a background in Sanskrit linguistics, labelled the dataset.[12] Since the entries are selected from the top-scoring nodes, we expected the results to be better than the macro-average performance of the system. We find that the evaluation of our system provides a precision of 0.84, recall of 0.91 and an accuracy 81.82, micro averaged over the 66 predictions.

3.3.4 Discussion

In Sanskrit, multiple affixes may give rise to similar patterns. An affix in Sanskrit contains two parts, where one part pertains to the pattern to be induced, and other is a marker which gets elided before the affixation. The presence of the marker, termed as '*it*' marker, also plays a role in determining the type of rules that get triggered during the derivation. For example, consider the word '*prāmukhya*' derived from '*pramukha*' and the word '*sodarya*' from '*sodara*'. Both the words have the same end-pattern '*ya*'. However, only in the case of the former, *vṛddhi* operation takes place but not in the latter. Now, affixes that carry the same pattern might differ by the '*it*' markers. Encoding every candidate word pairs with the suitability of rules of *Aṣṭādhyāyī* in $\mathcal{A}_1$, we can narrow down the possible candidates for the affix to at most four candidates of the 137 possible affixes. In order to disambiguate further, we require semantic and pragmatic level information, which is currently unavailable. In this work, we only consider the derivations in *taddhita*, as we find that jointly modelling a system for both *kṛdanta* and *taddhita* is challenging. The rule arrangement for *kṛdanta* is different from that of *taddhita* in *Aṣṭādhyāyī*. Here we might require a different model design for organising the rules in $\mathcal{A}_1$, i.e., phase 1 in Section 3.3.2. Hence in this work, we restrict ourselves to resolving *taddhita* nouns, which is the larger section in *Aṣṭādhyāyī* among the two.

In this work, we developed a graph-based semi-supervised approach for analysis of

[12] Here, 5 of the experts evaluated one set each and 2 of the experts evaluated 2 sets each. One of the expert evaluators was a co-author in the published proceedings of this work

derivative nouns in Sanskrit. We successfully integrate the rules from *Aṣṭādhyāyī*, variable length character n-grams learnt from Adaptor grammar and word embeddings to build a 3 step sequential pipeline for the task. We find that our work outperforms label propagation, which primarily shows the effect of the explicit design of network structure. We find that using the label distribution outputs at each phase, for the input at the successive phases, improves the results of the model.

3.4 Word Segmentation

Word segmentation in Sanskrit is challenging, primarily due to *sandhi* as it obscures the word boundaries in a sentence. The analysis of such a sentence results in ambiguity as it can be split into multiple possible segmentations. Let us consider such an analysis which shows all the phonetically valid word splits for a verse in Sanskrit, as shown in Figure 3.4a.[13] Here by virtue of *sandhi*, the words *brūyāt* (ब्रूयात्) and *brūyam* (ब्रूयम्), prefixed with numbers 14 and 15 respectively appear as two valid candidates in spite of the phonetic differences they possess from that of the original sentence. The word splits in Figure 3.4 are based on the analysis by a lexicon-driven analyser, Sanskrit Heritage Reader (SHR)[14]. The segmentation analysis of the aforementioned verse results in 1,056 different possible candidate solutions, such that each one of those solutions covers the entire input. We call such a solution as an '*exhaustive segmentation*'. Our task is to find an '*exhaustive segmentation*', which is also semantically valid. Figure 3.4b shows the semantically valid solution for given the sentence. The major contributions of this work are :

1. We propose a word segmentation approach which incorporates linguistic information in terms of a search space as well as in feature engineering.
2. We create a feature-set consisting of a total of 43 linguistically motivated features which makes use of various morphological and syntactic properties based on the traditional grammatical framework in Sanskrit [Bha90].
3. We use the framework of Path Constrained Random Walks [LC10], to formulate the linguistic features as horn clauses. This enables to calculate the relative strength of

[13] A saying from *subhāṣitam* text: 'One should tell the truth, one should say kind words; one should neither tell harsh truths, nor flattering lies; this is a rule for all times.

[14] Available at http://sanskrit.inria.fr/, SHR is a lexicon-driven segmenter which produces all the valid word splits. An interface is provided for manual selection of a solution [GH16].

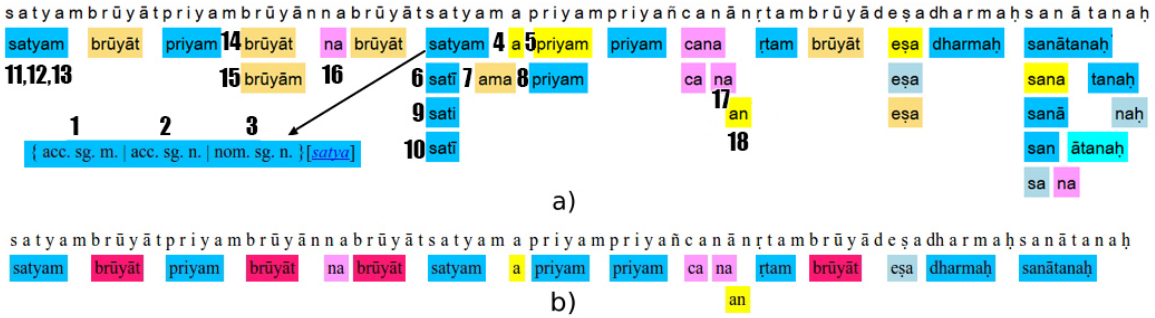


Figure 3.4 a) All the phonetically valid segmentations (link) for '*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanānṛtambrūyādeṣadharmaḥsanātanaḥ*' from (*subhāṣitam*) as output by Sanskrit Heritage Reader (SHR) and b) correct segmentation selected from the candidate space.

association between various candidate words in input based on distributional information from a morphologically tagged corpus. Leveraging the morphological constraints is particularly important as the word ordering in sentences in Sanskrit, especially for verse constructions, are of not much use.

3.4.1 Method

We treat the word segmentation problem as an iterative query expansion problem. Figure 3.5 presents an overview of the workflow of our approach. An input sentence in fused form is passed through the Sanskrit Heritage Reader. It provides all the possible segmentations for the input sentence, as shown in Figure 3.4. It also provides the morphological information for each of the segments, shown as entries marked 1, 2 and 3 in Figure 3.4. We now conceptualise a graph with these candidate segments as its nodes. Every pair of nodes that can coexist in an exhaustive solution forms an edge. We use about 43 linguistically motivated features, and a weighted aggregate of these features forms the edge weights. The features are formulated as horn clauses using Path constrained Random Walks (PCRW) [LC10]. We start the query expansion with the most promising candidate(s) as our initial query node. Then, at every iteration, we greedily choose a candidate, that has the highest aggregated edge score with the candidates that already are part of the partial solution. We now elaborate on each of these steps in detail.

Graph Formation Given an input string, we obtain our search space of possible word splits using SHR, as shown in Figure 3.4. The search space represents all the possible exhaustive segmentations with possible gaps and overlaps between the word splits in each of the exhaustive segmentation [Kud06, ON93, WW77].[15] We treat the search space as a graph, or to be precise, a multi digraph. Here, we choose to add edges between every pair of word splits that can coexist as words in at least one exhaustive segmentation. It needs to be noted that nodes 14 and 15 are suggested as alternatives to one another. Hence they cannot coexist in a single exhaustive segmentation. Such nodes are defined as *conflicting* node pairs, and they will not form edges between them. Henceforth, the graph so formed will be referred as the sentence graph $G(V, E, W)$. Here the edge density of the graph so formed will be quite high, and such a design decision is made due to the free word order nature of the language, especially for verses. Two words, irrespective of how far they are in their linear ordering, can still be closely associated with each other as elaborated in Chapter 1.
In Figure 3.4a, identical surface-forms with the same root are grouped together and displayed as a single entity. But we consider, every unique combination of the root, morphological class and the word position in SHR as a separate node in $G(V, E, W)$. Hence the surface-from *satyam* (सत्यम्), appears as six separate nodes numbered from 1-3 and 11-13 in Figure 3.4a. Here the nodes 1-3 differ from each other in terms of their morphological classes. The nodes 1 and 11 differ only in terms of their position owing to its repeated occurrence in the input. The position information is opaque to our proposed system and is used only in the formation of the nodes for the sentence graph. The edges in our model should capture the likeliness of two nodes to co-occur in the final solution. Hence, every pair of nodes that can co-occur in an 'exhaustive segmentation' forms two directed edges, one each at either of the directions.

Language Model The strength of association between two nodes in the sentence graph G, i.e. how likely are two nodes to co-occur in an exhaustive solution, is calculated based on the distributional information from a morphologically tagged corpus. We represent our corpus, from which we obtain the distributional information for our task, as a Graph $G_2(V_2, E_2, W_2)$. We will henceforth refer to this graph as 'corpus graph'. We add every unique lemma[16], morphological tag and the inflected surface-form that occur in the corpus as a vertex in the graph G_2. This makes the corpus graph a typed graph, which has nodes of three types.

[15] The word splits in an exhaustive segmentation often overlap and sometimes leave gaps by virtue of Sandhi.
[16] Here we use the term 'lemma' loosely to indicate a nominal stem or a verbal root.

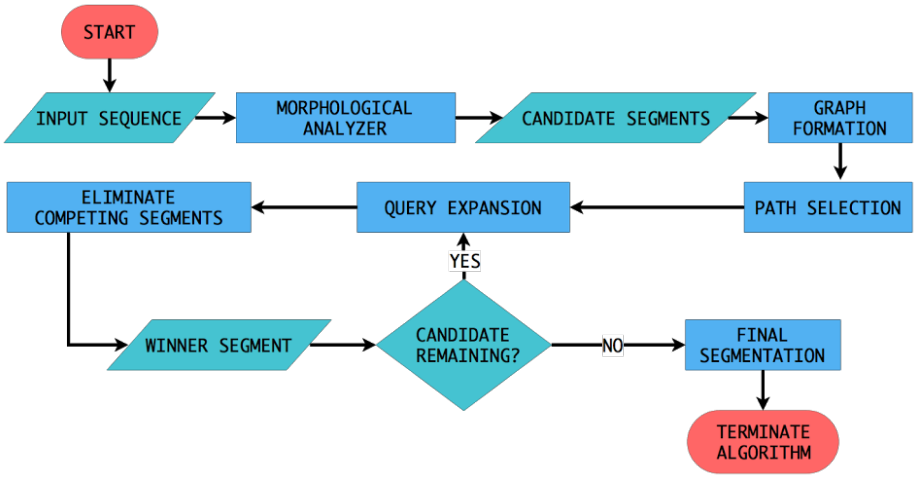


Figure 3.5 Flowchart for the proposed approach.

So, we define V_2, the vertex set as the union of the vocabulary of each of the three attributes. Now we form directed edges (in E_2) between every pair of nodes that co-occur in a sentence in the corpus. We calculate the edge weights (in W_2) using the expression 3.7.

$$P_{co}(j|i) = \frac{count(j,i)}{count(i)} \tag{3.7}$$

Here P_{co} essentially calculates the likelihood that the node j will occur in a sentence given that node i already occurs in that sentence. In expression 3.7, $count(i,j)$ denotes the count of sentences in the corpus G_2 in which the nodes i and j co-occur. $count(i)$ is the count of sentences in which node i occurs in the corpus. For example, in order to compute the edge-weight for a directed edge in G_2 starting from the morphological tag node 'genitive(gen), masculine(m), singular(sg)' to a lemma node '*hari*' (हरि), we calculate $P(hari|gen.m.sg.)$, i.e, conditional probability of the lemma '*hari*' to occur in a sentence given that a word with the morphological tag 'gen.m.sg.' already occurs in it. Similarly, we find edge weights from every possible pair of nodes that co-occur in the given corpus.

Identifying Features as Path Selection Using PCRW A node in the sentence graph G encodes three different linguistic attributes, namely the lemma (L), morphological tag (M) and the surface form of the word (W). Henceforth we will denote an attribute as $\mathcal{A}$ where it can either be an L, M or W. In the corpus graph G_2, each of these attributes is a type and the corresponding attribute value is a node. Now, for two nodes a, b in the sentence graph, we can obtain nine different 'P_{CO}' ($\mathcal{A}_a \rightarrow \mathcal{A}_b$) scores based on distributional information

from the corpus graph. Given that sentence graph G is a multi digraph, we can form 9 different edges between the nodes a and b. Each edge indicates the edge weight of the attribute pair as obtained from G_2. From a graph traversal perspective, this can be seen as nine different ways by which one can reach b directly from the node a. But, we extend this notion of reachability. We will construct an edge in G between a and b if there exist various directed paths between the corresponding attributes of these nodes in G_2. Given the possibility of exponential blow-up in the number of possible paths between a given pair of words, we restrict this by filtering only those paths which satisfy certain linguistic constraints. Each such path will be considered a feature. We use the Path Constrained Random walks (PCRW) [LC10] to formalise this path generation and filtering approach.
PCRW can be seen as a procedure for calculating feature values from a typed graph, such as knowledge bases. Here each feature is formulated as a horn clause or equivalently as a sequence of typed nodes, i.e. a path where each node belongs to a specific type [LC10, GM15]. Though the PCRW approach facilitates to learn horn clauses, we instead define linguistically motivated features manually and formulate them as horn clauses. This enables us to calculate the score of the features from the typed graph. To formalise the feature definition in terms of PCRW, we first define feature templates, henceforth to be referred to as path types. Now, let us illustrate the notion of path-types. We have already seen that there can be nine different weighted edges between the node pairs a and b. The feature that calculates the strength of association between the lemma attribute of the node a and the surface form attribute of the node b can be denoted with the expression 3.9. Similarly, all the nine single edge features can be represented using a single feature template, i.e. a path type, as shown in expression 3.8.

$$\mathcal{A}^{*}_{node\ id} \xrightarrow{P_{co}} \mathcal{A}^{*}_{node\ id} \qquad (3.8) \qquad\qquad L^{*}_{a} \xrightarrow{P_{CO}} M^{*}_{b} \qquad (3.9)$$

By adding constraints to the path types, we add more expressivity to this formalism. The general form for any single edge path type is represented using the expression 3.10

$$\mathcal{A}^{constraint}_{node\ id} \xrightarrow{P_{co}} \mathcal{A}^{constraint}_{node\ id} \qquad (3.10) \qquad\qquad L^{Nominal}_{a} \xrightarrow{P_{CO}} M^{Verb}_{b} \qquad (3.11)$$

The superscript is a constraint, and only those nodes in G that satisfy the constraint can be used to form the feature from the template. The feature in expression 3.11 is such an instance where the node a must be a nominal, and the node b must be a verb. For nodes of other types, this feature is not valid. Please note that the constraint 'Nominal' is defined

as the set of all the possible morphological tags that a nominal word-form can assume. Each morphological tag in the Nominal set conveys the case, gender and plurality of a word. We similarly define disjoint sets of morphological tags for verbs, compounds and indeclinables. The set '$Constraint$' is defined in the expression 3.12, where $\mathcal{P}$ denotes a power set. By defining '$Constraint$' as a power set, the formulation enables to group different morphological tags as a set and use it as a constraint.

$$Constraint = \{\mathcal{P}(Nominal) \cup \mathcal{P}(Verb) \cup \mathcal{P}(Compound) \cup \mathcal{P}(Indeclinable) \cup \{*\}\} \tag{3.12}$$

Here, we use $*$ as a wild card to indicate that there are no constraints applied on the path type. Expression 3.8 uses this wild card, implying the features from this template do not have additional constraints for the nodes to satisfy.

Now we can extend the edge path-types to arbitrary length path types as follows. Path-types starting from node a to node b can be defined recursively as follows:

$$PathType : \mathcal{A}_a^{Nominal} \xrightarrow{P_{CO}} \mathcal{A}_{j_1}^{Verb} \xrightarrow{P_{CO}} \mathcal{A}_{j_2}^{Verb} .... \xrightarrow{P_{CO}} \mathcal{A}_b^{Verb} \tag{3.13}$$

$$P(PathType) = P(j_1|a).P(b|j_k) \prod_{l=2...k} P(j_l|j_{l-1}) \tag{3.14}$$

Expression 3.14 calculates the association score between the nodes a and b based on the feature. It is nothing but the product of all P_{CO} scores present in the directed path originating from the node a and terminating at node b.[17] In the case of paths from path-types of length one, the path-value is same as the P_{CO} value obtained from G_2 for the attribute pair. The path-values so obtained can be seen as a random walk score, where the random walk traversal is over the graph G_2 starting at node $\mathcal{A}_a$ and terminating at $\mathcal{A}_b$ with all other nodes in the path as intermediate nodes.

Table 3.9 defines various path types that we use in our system. Here, we also mention the linguistic property against each path type, that motivated us to formulate them. Though we do not enumerate all the path types shown, it is implied that path types of other attributes with a similar structure are also formed. The number of path types for each of the linguistic property is mentioned inside the parenthesis. For instance, in case of general relations, there

[17] Here, when we say node a and node b, we imply an attribute of a and b, i.e. $\mathcal{A}_a$ and $\mathcal{A}_b$.

Id	Linguistic Property	Path types
1	General Relations (3)	$L_i^* \longrightarrow L_j^*$
2		$M_i^* \longrightarrow M_j^*$
3	Compounds (1)	$p(L_i^{compound} \vert L_j^{compound})$
4	Expectancy (27)	$M_i^{nominal} \longrightarrow M_j^{verb} \longrightarrow L_k^{nominal}$
5		$L_i^{nominal} \longrightarrow L_j^{verb} \longrightarrow L_k^{nominal}$
6	Proximity Nominals (4)	$W_i^{nominal} \longrightarrow L_j^{nominal}$, where,$M_i = M_j$
7		$L_i^{nominal} \longrightarrow L_j^{nominal}$, where,$M_i \neq M_j$
8	Proximity Verbs (4 $\times$ 2)	$L_i^{verb-\{absolutive\}} \longrightarrow L_j^{absolutive}$
9		$L_i^{verb-\{gerund\}} \longrightarrow L_j^{gerund}$

Table 3.9 Path-types for path. The number of path types per linguistic property is shown in parenthesis in the 'Linguistic Property' column. For proximity verbs, we exclude the participles, gerund and absolutive, as marked in their constraints. In proximity nominals and verbs, we exclude the M attribute from the path-types and hence a total of 12 path-types.

exists an additional path $W_i^* \rightarrow W_j^*$, which uses the word surface-form attribute between every pair of nodes in the search space graph G. Now, we elaborate on the linguistic properties based on which we designed our feature template, i.e. the path-types.

- **General Relations:** The general relations use the '$*$' constraint implying that a node of any type can be part of this typed path. Here we restrict ourselves to the path types which have the same attribute types at both the ends of the path (edge). This leads to a total three path types.
- **Compounds:** The components in a compound strictly follow a sequential order in their formation. Hence, we use the bigram counts between the components instead of the P_{CO} score.
- **Expectancy of a Verb:** These are path types of length 3, where the middle node should always be a verb, and the other two nodes should be a nominal. We consider each of the 27 possible attribute combinations as a separate feature. Every word, especially the verb, expects different syntactic roles in its sentence usages. The morphological markers of an inflected word are indicative of its syntactic characteristics. The path types 4 and 5 try to capture this notion of expectancy between the verbs. These paths do not use any explicit syntactic information due to unavailability of labelled data, and instead, use the morphological markers as a noisy alternative to capture this notion. We call this notion as expectancy

(ākāṇkṣā) based on the traditional syntactic analysis framework in Sanskrit [KSSS15]. This concept stands close to the concept of valency of verbs, but with an important distinction that expectancy, unlike valency, is mutual between the words involved.[18]

- **Proximity** - The concept of proximity in Sanskrit stands close to the notion of 'Dependency locality' [KSSS15, Gib98]. But this may not be strictly followed in Sanskrit poetry. In the case of verses, the poet often needs to adhere to the metrical constraints. we focus specifically on the violation of proximity in the case of modifier-modified relation between two nominals [KSSS15]. In such situations, morphological markers are one of the ways to identify related words. Paths 6 - 9 attempt to capture this notion. In path 6, we find the co-occurrence probabilities of two nodes when both appear in the same Morph, which is indicative of two nodes being potentially a pair of modifier-modified words. Path 7 exactly finds the inverse where we find the co-occurrence probabilities of two nodes when they are not in the same Morph. Paths 8 and 9 try to capture the proximity between the verbal inflections with that of the absolutives and the gerunds.

Iterative Query Expansion with PCRW For the iterative query expansion, we first form induced subgraphs of G, where all the edges formed in G due to a particular feature form a separate induced-subgraph. We formulate our task as that of iterative query expansion, performed independently over each of the induced sub-graphs. Inspired by the maximum matching approach, originally proposed for Chinese word segmentation [CL92], we choose the initial query node by identifying the longest candidate from the set of candidates. The initial query is then expanded by adding nodes iteratively to it. For the query expansion, we first consider the feature scores from features where the source node will be a query node and the target node is not already a part of the query, i.e. the partial solution. The target node with the highest score, aggregated across all the induced sub-graphs, is then chosen as the winner node. Further, for the general relations and the path type 6 in Table 3.9, we perform random walks with restarts over their corresponding induced subgraphs to capture the structural properties of the graph. The edge weights to initialise the weights, in this case, are obtained again from P_{CO} likelihood estimated over G_2. With other features, we do not perform random walks with restarts over their induced subgraphs, but instead, the scores obtained from G_2 are directly used. This can be seen as random walk traversal performed over the corpus graph G_2.

[18] It needs to be noted that two nominals also can express expectancy between them (such as modifier-modified relations) and is not exclusive to verbs. But here we consider only expectancy related to verbs.

This effectively helps us to leverage the structural properties of the content graph structure G and the corpus graph G_2, which ideally represents the distributional properties in the language (corpus). The final score of a target node is the weighted sum of scores from each of the feature, where the weights are learned using a supervised parameter estimation approach.

Supervised Parameter Estimation for Path-Types Every path type can be considered as a feature, and the random walk score at each instance for a pair of (query, source) node is a feature-value. We need to estimate the relative importance of each path type. Our input is a pair of words, the source word and target words. We generate all possible positive instances and down-sample the negative training instances from a training set of 5000 sentences. Each such instance is labelled as a +1 or 0, based on whether or not they co-occur in a sentence in the corpus. We use logistic regression with L-BFGS [AG07] optimisation procedure to handle over-fitting. We follow the same optimisation procedure as followed in Lao et al. [LC10]. The authors argue that this approach is superior to the alternative one-weight-per-edge-label approach.

3.4.2 Experiments

Comparison with the Existing Approaches We compare our system's performance with the existing approaches for word segmentation. Here we compare results from 'Sanskrit Sandhi Splitter (S3)' [NC11] and two configurations from Mittal [Mit10]. Table 3.10 reports the results of our system with the other three aforementioned systems on a test dataset of 2148 strings, used in Natarajan and Charniak [NC11]. Our approach provide overall improvement of 28.81 percentage points in F-score from the previously best performing system (NC4). A total of 1784 of 2148 (83.05%) strings were segmented with an F-score of one.

Dataset We use the Digital Corpus of Sanskrit (DCS), to build the language models and train our model. We used a version of the corpus with 430,000 sentences, which was the available version at the time. We randomly select 100,000 sentences of varying length from the DCS corpus, from which 90,000 of the sentences, henceforth to be referred to as "DCS 90K", were used for calculating the co-occurrence values P_{CO}. We keep the remaining 10,000 sentences as held-out data which was not used at any point in our framework. None of these sentences was used for the supervised parameter estimation component of our

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

Table 3.10 Performance evaluation of our proposed system, S with the state of the art, NC4 [NC11], OT2 and OT3 [Mit10].

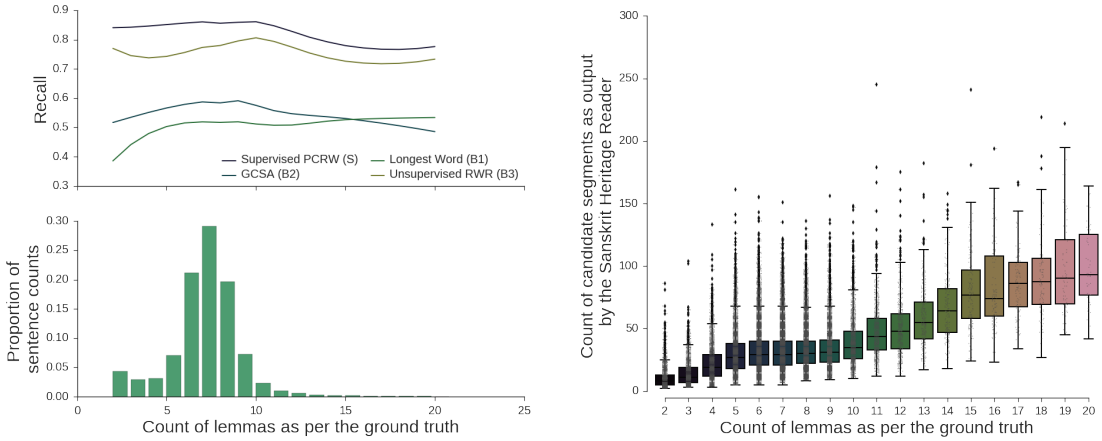
	DCS 90K			Held-out Dataset		
System	P	R	F	P	R	F
B1	37.02	50.81	42.83	36.89	50.74	42.72
B2	42.25	58.62	49.115	42.04	58.45	48.91
B3	65.13	76.78	70.48	65.07	76.70	70.41
S	74.11	84.92	79.15	73.28	82.73	77.72

Table 3.11 Performance evaluation of our proposed system, S with the competing baselines B1, B2 and B3 over 100,000 sentences from DCS.

model. The sentences in DCS90K form the data from which corpus graph G_2 is constructed for the language model.

Baselines Systems previously proposed for word segmentation in Sanskrit are not designed to process sentence level inputs. Additionally, the systems do not consider the morphological information in their training phases, and hence it will be unfair to use those systems for further analysis. We propose the following methods as our baselines. In all the systems, we use the morphological analyser output as the input to predict the correct segmentation.

- **Longest Word Selection (B1)** - Inspired by the maximum matching approach from Chen and Liu [CL92], we iteratively select the longest candidate available from the given set of possible segmentations. At each step of the iteration, we eliminate the candidates conflicting with the current winner. This method does not use any morphological information as such.
- **Greedy Candidate Selection Approach (B2)** - Here we consider the morphological analyser output to be like a tree, and we perform a greedy selection that maximises the overall likelihood of the selection. In this approach, we combine the co-occurrence probabilities mentioned in paths 1, 2 and 6 in Table 3.9.
- **Unsupervised RWR - General Relations (B3)** - In this approach, we form the graph G for the sentence, similar to as mentioned in Section 3.4.1. However, we only use path-types from the general relations and path type 6 of Table 3.9. We perform Random Walks with Restarts (RWR) over the graph G and then average the scores from different random walk runs to obtain a winner node.



(a) Recall Vs. GT frequency distribution **(b)** Candidate Vs. GT frequency distribution

Figure 3.6 Analysis of systems based on frequency distribution of ground truth (GT) lemmas.

- **Supervised PCRW (S)** - This is our proposed system, as defined in Section 3.4.1. Here supervision is performed only for parameter estimation used in the weighted sum of the path scores, based on which the winner node is greedily selected.

Results We report the results for the joint task of word segmentation and compound splitting for all the aforementioned systems. We use the macro-averaged Precision (P), Recall (R), F1-score (F) to report the performance on our experiments. Table 3.11 presents the results for all the competing models. Our final system S performs the best with a percentage increase of 13.79 % and 10.60 % in precision and recall, respectively, from the next best system B3 on the DCS 90K dataset. For the held-out dataset, the results remain more or less consistent with those obtained over DCS 90K for all the systems. Our system could predict all the correct segmentations for as many as 38.64 % of the sentences against the 19.57 % fully correct predictions of B3. The line graphs in Figure 3.6a illustrate the recall of each system over the frequency distribution on the 'DCS 90K' based on the number of lemmas per sentence in the ground truth. It can be observed that our system works better for the sentences of all the lengths. The average length of sentences in our dataset is 6.87, and the distribution is presented in the bar chart. We also find that higher the number of words in a sentence, the count of candidate segmentations increases manifold. This can be observed from the boxplot, as shown in Figure 3.6b plotted for the dataset of 100,000 sentences from DCS.

Summarily, we presented an approach for word segmentation in Sanskrit using supervised PCRW. We treated the problem as a query expansion problem. For this, we used a heuristic search over an exhaustive search space as produced by the Sanskrit Heritage Reader. We find that the inclusion of morphological information by means of linguistically motivated ILP path-types greatly increases the performance of the system by as much as 12.30 percentage points in F-Score. Also, our system outperformed the previously best performing system by 28.81 percentage points in F-Score.

3.5 Summary

Pāṇini's grammar, *Aṣṭādhyāyī* , is admired universally across the various school of thoughts in linguistics for its coverage of the Sanskrit language [Kip09, Hym09]. In computational linguistics, rules from *Aṣṭādhyāyī* have been primarily used for formal language based analyses and development of rule-based systems for processing of Sanskrit texts [GHK$^+$12, Kul13, GHK$^+$12]. But here, we show that the rules from *Aṣṭādhyāyī* can be used effectively as discriminative features for data-driven approaches as well. Similar in spirit to the efforts by Hellwig [Hel09b], we propose data-driven models for processing Sanskrit texts. Our primary focus has been to integrate linguistic knowledge explicitly into the data processing pipelines for various NLP tasks in Sanskrit. In this chapter, we looked into three different tasks of compound type identification, analysis of derivational nouns and word segmentation. The work on compound type identification can be seen as an extension to the line of works suggested in Kulkarni and Kumar [KK13], and ours is the first such system to incorporate semantic relations in taxonomy as well as class specific subword unit structures for the task. Results from our system demonstrate the effectiveness of using information from lexical databases for the task and it is a promising direction to be explored. We can extend the current system by incorporating other lexicons such as Indo-wordnet [SRB06] along with *Amarakośa*. Our work on derivational noun analysis will be beneficial to the Sanskrit Computational Linguistic community for parsing of digitised ancient manuscripts. Currently, no other analyser in Sanskrit handles derivational nouns. Our work doubles as a tool for pedagogy, as we are able to abstract out regularities between the patterns and narrow down the possible affix candidates to at most four possibilities, for a word pair. For both of these tasks related to word formation, we find that the use of linguistic knowledge such as rules from *Aṣṭādhyāyī* , subword patterns learned using Adaptor Grammar etc. has shown to be ef-

fective for the aforementioned tasks. The auxiliary information from statistically significant discriminative sub-word units and word vectors, complement the rule based features and improve the overall performance of such models. Our models still provide the state of the art performances for their respective tasks. The word formation in compounding and derivation via Taddhita affixes are often recursive [Bha89, Low15]. In our current models, we only considered cases where there are exactly two morphemes involved. It would be desirable to build end-to-end systems that can handle words generated using an arbitrary number of morphemes, stems in compounds and affixes in derivation. We leave this for the future.

In the word-segmentation task we discussed here, our focus was primarily on designing a hand-crafted feature set and obtaining a valid input representation, i.e. designing the search space. Since there is no guarantee of proximity to be maintained between words in a sentence, we presume that treating the input as a sequence might be a serious limitation for the task. Instead, we design our search space as a graph, which is agnostic to the poetry or prose word-order followed in the construction. Through our experiments, we showed the effectiveness of our system on a dataset that contains both prose and poetry constructions. We also showed that our approach is scalable and can be applied to larger datasets and longer constructions, as compared to previous systems. As an additional outcome of our efforts, we also release a dataset for word segmentation in Sanskrit.[19] It consists of 115,000 sentences obtained from the Digital Corpus of Sanskrit [Hel16a], along with the search space obtained from the Sanskrit Heritage Reader. Here we bridge the gap in alignments between the candidate segmentations obtained from SHR and the gold-standard segmentation solution from DCS, which is a linguistically involved process. We hope the release of this dataset would catalyse the research efforts in this direction. In the subsequent chapters, we first show how structured prediction tasks such as word-segmentation and other downstream tasks perform when we use purely statistical approaches. Further, as a natural extension to our current approach for segmentation, we propose a search-based structured prediction framework which uses the search space obtained from SHR as input and also automates the learning of feature function instead of the hand-crafted features we use here.

[19]The dataset can be accessed at https://zenodo.org/record/803508#

Chapter 4

Sequence to Sequence Models for Low Resource Language Processing

The last decade has seen tremendous advancements in the digitisation of ancient manuscripts in Sanskrit. However, there exist relatively limited options for completely automated tools [HN18, AGP+18, Hel09b, Hel16c] that can aid such digitisation efforts in Sanskrit to scale up. Purely data-driven approaches, such as deep learning, have shown to be effective and scalable for various NLP tasks across different languages [Man15]. In this chapter, we consider three tasks, namely, word segmentation, post-OCR text correction and poetry to prose conversion. Here, all the three tasks are formulated as sequence to sequence (Seq2Seq) models [SVL14, Yu18]. In these models, we use little to no linguistic information and rely entirely on the supervised Seq2Seq models to capture the structural regularities present in the sequences. We specifically investigate the feasibility of such approaches for a low-resource language like Sanskrit.

The models learnt here for all the three aforementioned tasks are essentially conditional language models. Seq2Seq models consist of an encoder to encode an input sequence into a context vector and a decoder that predicts a sequence of entries from its vocabulary. Here, the probability of each prediction is dependent on its previously predicted entries and the context vector obtained from the encoder. The generation of an entry is constrained by its presence in the fixed-set vocabulary of the decoder. This implies that the effectiveness of such a model is dependent on the size of the vocabulary and a training corpus with a high type-to-token ratio. Corpora in morphologically rich languages like Sanskrit express a low type-to-token ratio. Hence the use of a token level vocabulary can adversely affect the

performance of the data-driven approaches. Instead, we choose to learn a new vocabulary for the encoder-decoder models using greedy tokenisation approaches like BPE [SHB16] or sentencepiece [SN12], which find recurring and statistically significant string patterns in a corpus and use them as entries in a new vocabulary. Such approaches can be fine-tuned to enhance the size and the type-to-token ratio of the entries in the vocabulary and also to avoid Out of Vocabulary words in corpora [Kud18].

We first consider the task of word segmentation in Sanskrit. Unlike word segmentation tasks in languages like Chinese, Korean, etc., the word segmentation in Sanskrit involves string transductions due to *Sandhi* (सन्धि).These transductions happen at the juncture of words of a sentence, which makes sentence-wide contextual information desirable for the task. We consider a standard Seq2Seq configuration with attention for the word segmentation task. Here, other than the parallel joint and segmented sequences as training data, we do not incorporate any task-specific linguistic information to the model. We find that, with sufficient data, the Seq2Seq model can still outperform a linguistically involved word segmentation model, as discussed in Section 3.4. Similarly, we propose Seq2Seq solutions for post-OCR text correction and poetry to prose word order conversion tasks, but with augmentations to the standard architecture. For the Post-OCR text correction task, we incorporate the copying mechanism [GLLL16] that estimates the probability of an entry in the input to be copied to the decoder side. This is in addition to estimating the probability of predicting an entry from the decoder vocabulary in a standard Seq2Seq architecture. In the case of poetry to prose ordering task, we employ a Seq2Seq architecture with gated CNNs and a sequence-level loss for achieving state of the art performance for the task. We briefly explain each of these tasks and the settings now.

Word Segmentation: We propose a knowledge-lean data-driven approach for the word segmentation. We introduce a scalable Seq2Seq model [SVL14] that only requires the sequence as the input during runtime. The training involves the use of the fused and segmented sequences as the input and the ground truth respectively for the task. We use a standard encoder-decoder architecture with stacked LSTMs, by following the set up of Wu et al. [WSC+16], originally proposed for Neural machine translation. In spite of being a generic Seq2Seq model with no task specific modifications, the model obtained an improvement of 16.79 % in F-Score as compared to the then state of the art linguistically involved model [KSS+16a], which we discussed in Section 3.4. We find that the Seq2Seq

configuration with attention significantly outperforms the variant without attention with a percentage improvement of 23.49 %.

Post-OCR Text Correction: OCR based solutions for digitising texts written in Romanised Sanskrit are scarce. Instead of developing a full OCR solution from scratch, we make use of robust OCR solutions developed for other languages written in Roman script and then pose the problem as a post OCR-text correction task. For the task, we synthetically generate training images, which replicate the settings on two different scan settings based on a real world dataset of 430 images. The use of copying mechanism [GLLL16] yields a percentage increase of 7.69 % in Character Recognition Rate (CRR) than the current state of the art model in solving monotone sequence-to-sequence tasks [SEDDG16]. We also find that our approach outperforms current OCR based solutions for Sanskrit. Here, the system essentially learns two probability distributions, one for copying the input unit in the sequence and the other for predicting a unit from the decoder vocabulary. The sum of the probabilities for each of the unit is then used for making the prediction. A human judgement survey, performed on the models, shows that our proposed model results in predictions which are faster to comprehend and faster to improve for a human than the other systems.

Poetry to Prose conversion: The word ordering in a Sanskrit verse is often not aligned with its corresponding prose order. Conversion of the verse to its corresponding prose helps in better comprehension of the construction. Owing to the resource constraints, we formulate this task as a word ordering (linearisation) task. In doing so, we completely ignore the word arrangement at the verse side. *kāvyaguru* (काव्यगुरु), the approach we propose, essentially consists of a pipeline of two pretraining steps followed by a Seq2Seq model. The first pretraining step learns task specific token embeddings from pretrained embeddings [KWC18]. In the next step, we generate multiple hypotheses for possible word arrangements of the input [WCM18]. We then use them as inputs to a neural Seq2Seq model for the final prediction. Both the pretraining steps use self-attention mechanism to obtain the task-specific embeddings and the multiple hypotheses for the Seq2Seq model. The Seq2Seq model we employ uses a convolutional Seq2Seq architecture [GAG$^+$17], to make use of a sequence level loss to update the parameters of the model. We empirically show that the hypotheses generated by our pretraining step result in predictions that consistently outperform predictions based on the original order in the verse. Overall, *kāvyaguru* outperforms current

state of the art models in linearisation for the poetry to prose conversion task in Sanskrit.

4.1 Word Segmentation as a Seq2Seq Model

We propose a purely data-driven word segmentation (and compound splitting) approach for Sanskrit sentences.[1] We follow the standard sequence to sequence (Seq2Seq) learning architecture [SVL14] by Wu et al. [WSC+16] originally proposed for Neural Machine Translation. Our model consists of an encoder and a decoder network, both of which are realised as stacked LSTMs of 3 layers each. We also experiment with a variant where an additional attention network [BCB15] is incorporated into the encoder-decoder framework. The input sequence is converted into a sequence of vectors, which are then fed to the LSTM cells sequentially. The hidden state of the LSTM gets updated at each time step, and the sequence is represented as the sequence of each of these hidden state vectors. The encoder finally outputs a single fixed-size vector as the representation for the input sequence. The decoder then predicts one symbol after the other from its vocabulary. Here the probability of predicting a symbol from its vocabulary is conditioned on the sequence representation obtained from the encoder side and the predictions made till the previous time step at the decoder. We use softmax activation function at the decoder and perform greedy decoding to obtain the final prediction. We update the parameters using Adam optimiser [KB15] for our model. While the standard vanilla Seq2Seq uses the same fixed length vector from the encoder as the context vector throughout the decoding, the attention network enables to dynamically obtain a context vector at each decoding step by using the vectors for each input entry at the encoder [BCB15]. In our case, the unsegmented ('*sandhied*') sentences become the input and the corresponding segmented (*unsandhied*) sentences become the target sequences to be predicted at the decoder. Following the insights from Sutskever et, al. [SVL14], we reverse the sequence order at the input. Our training objective is to maximise the log probability of a segmented sequence T for a given unsegmented sequence S [SVL14].

$$\frac{1}{|\mathcal{S}|} \sum_{(T,S)\in\mathcal{S}} log\ p(T|S)$$

where the $\mathcal{S}$ is the training set. During inference, we need to predict the sequence T', which is the most likely sequence as per the trained model for the given input [SVL14].

[1] The code for the model can be found at https://github.com/cvikasreddy/skt

$$T' = \underset{T}{\operatorname{argmax}}\ p(T|S)$$

Vocabulary Enhancement for the Model: - Sanskrit shows high productivity in terms of compounding and is a morphologically rich language. Constraining the model to predict only from a fixed token vocabulary in a corpus may not be effective for such languages. Instead, we adopt a purely statistical approach for our model. Rather than relying on the real word boundaries, we use the '*sentencepiece*' model, an unsupervised text tokeniser [SN12] to obtain a new vocabulary for the corpus. The new vocabulary so learnt consists of string units that need not be semantically valid but are recurring patterns in the corpus. This is a greedy approach used to identify new text units from a corpus that maximises the likelihood of the language model so learnt [SN12]. The method was originally proposed for the segmentation problem in Japanese and Korean speech recognition systems. Use of such an approach enables to limit the number of entries in a vocabulary, thereby increasing the corpus evidence of each of the entry.

Sandhied Sentence (Original)
putraṁ vaṁśakaraṁ rāma nṛpasaṁnidhau

Sandhied Sentence (GibberishVocab)
_putraṁ _vaṁś akar aṁ _rāma nṛpa saṁnidh au

Unsandhied Sentence (GibberishVocab)
putraṁ _vaṁśa _kara m_rāma_nṛ pa_saṁ nidhau

Unsandhied Sentence (Original)
putraṁ vaṁśa karam rāma nṛpa saṁnidhau

Figure 4.1 Sandhied and unsandhied sentence expressed in original writing and with the new learnt vocabulary 'GibberishVocab'.

Figure 4.1 shows the instance of text-units learnt from the *sentencepiece* model corresponding to the original input from the corpus. In the *sentencepiece* model, the 'space' in the original input is also treated as a character and is replaced with the special symbol '_'. So 'aṃ_rāma', which originally is part of two words, is an entry in our encoder vocabulary. Hence, the sandhied sentence, when retokenised with the sentnecepiece vocabulary has the string 'aṃ_rāma'. Our model uses only the sequences tokenised as per the new vocabulary, henceforth to be referred to as '*GibberishVocab*'. Note that the decoder also outputs entries in *GibberishVocab*. The output from the decoder is then converted to the original

vocabulary for evaluating the outputs. Given the explicit marker added by sentnecepiece, it is a deterministic process to retrieve the original sequence from it.

4.1.1 Experiments

Dataset: We use a dataset of 107,000 sentences, obtained from the Sanskrit Word Segmentation Dataset [KSG17]. This is a subset of the Digital Corpus of Sanskrit [Hel16a]. From the dataset, we use only the input sentences and the ground truth inflected word-forms. We ignore all the other morphological and lemma information available in the dataset. We used 214,000 strings both from input and output strings in the training data to obtain the *GibberishVocab* using *sentencepiece*. We find that the default vocabulary size of 8,000 for the *GibberishVocab* works best. Of the 8,000 vocabulary entries, the encoder vocabulary size is 7,944, and the decoder vocabulary size is 7,464. This shows a high overlap in the vocabulary in *GibberishVocab* at both input and output sides, in spite of the difference in phonetic transformations due to sandhi. Originally the training data contained 60,308 segmented words at the output side. By reducing the vocabulary size at the decoder side to 7,464, we make the probability distribution (softmax) at the decoder layer denser. A linguistically motivated morphological analysis still would have resulted in 16,473 unique lemmas in the training dataset, much more than double the vocabulary learnt. From the dataset, we use a test data of 4,200 sentences which were not used in any part of the training.

Evaluation: We use macro averaged precision, recall and F-Score to evaluate our model as is the standard with evaluating word segmentation models.

Baselines

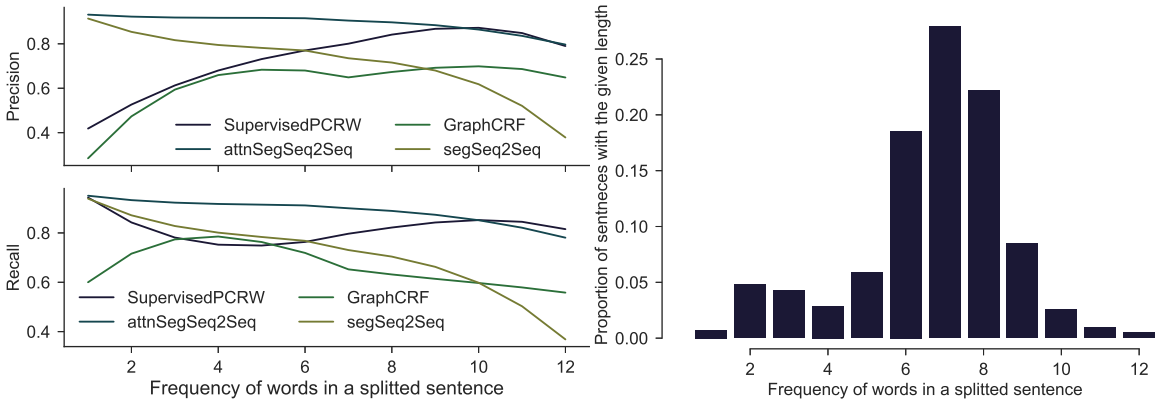
We compare the performance of our system with two other baseline systems.

- **supervisedPCRW [KSS$^+$16a]:** This is the linguistically involved iterative query expansion approach we discussed in 3.4.
- **GraphCRF:** We use a structured prediction approach using a second order Conditional Random Fields model [MB14]. We first obtain the possible candidate segments using a shallow parser, Sanskrit Heritage Reader [GH16] and then convert the segments into a graph. For every node segment, we learn a word vector using fastText [BGJM17].
- **SegSeq2Seq:** This is the proposed encoder-decoder model but without attention.

- **AttnSegSeq2Seq:** This is the proposed encoder-decoder model with attention.

Training Procedure and Hyperparameters

Our models have three layers each at the encoder and decoder. These models contain an embedding layer which is a trainable matrix with individual word vector having a size of 128. Our LSTM layers consist of 128 cells at both the encoder and decoder layers. We train the sentences in a batch size of 128 and keep the sequence length of each sequence to 35. The initial learning rate was set at 0.001, and we trained our system for 80 epochs after which the network parameters converged. We used Adam optimiser with parameter values β_1, β_2 as 0.9 and 0.999, respectively. The dropout in the hidden layers was set to 0.2, after experimenting with different settings from 0.1 to 0.4 in step sizes of 0.1. Dropout helps to avoid over-fitting of data [SHK+14]. Both the 'SegSeq2Seq' and 'AttnSegSeq2Seq' models follow the same architecture and have the same hyperparameter settings, and they vary only on the attention mechanism.



(a) Precision and Recall for the competing systems grouped based on the count of words in each of the test sentence.

(b) Distribution of strings in test dataset grouped based on the count of words in each sentence.

Figure 4.2 Results on the test dataset. The sentences are grouped based on the count of words in the segmented sentences.

Results

Table 4.1 shows the performance of the competing systems. We observe that the system 'AttnSegSeq2Seq' outperforms the then state of the art SupervisedPCRW with a percentage increase of 16.29 % in F-Score. The model 'SegSeq2Seq' falls short of supervisedPCRW

Model	Precision	Recall	F-Score
GraphCRF	65.20	66.50	65.84
SupervisedPCRW	76.30	79.47	77.85
SegSeq2Seq	73.44	73.04	73.24
AttnSegSeq2Seq	90.77	90.3	90.53

Table 4.1 Macro-averaged Precision, Recall and F-Score for the competing systems on the test dataset of 4200 strings.

with a percentage decrease of 6.29 % in F-Score. It needs to be noted that the systems 'AttnSegSeq2Seq' and 'SegSeq2Seq' are exactly the same architectures other than the addition of attention in the former. However, 'AttnSegSeq2Seq' reports a percentage increase of 23.61 % as compared to the performance of the 'SegSeq2Seq' system. One probable reason for this could be the relatively free word order nature of the language. Here, attention enables to dynamically capture the salient features of the entire input, rather than compressing the entire input sequence into a single static representation as done in the case of a standard Seq2Seq model without attention.

Figure 4.2 shows the results of the competing systems on strings of different lengths in terms of words in the sentence. This should not be confused with the sequence length. Here, we mean the 'word' as per the original vocabulary and is common for all the competing systems. For all the strings with up to ten words, our system 'AttnSegSeq2Seq' consistently outperforms all the systems in terms of both precision and recall. SupervisedPCRW performs slightly better than our system, for sentences with more than ten words. It needs to be noted that the average length of a string in the Digital Corpus of Sanskrit is 6.7 [KSS$^+$16a]. The proportion of sentences with more than ten words in our dataset is less than 1 %. The test dataset has slightly more than 4 % sentences with ten or more words. The 'SegSeq2Seq' model performs better than the state of the art for both Precision and Recall for strings with less than or equal to 6 words. Figure 4.2a shows the proportion of sentences in the test data based on the frequency of words in it. Figure 4.2b shows the proportion of strings in the test dataset based on the number of words in the strings. Our system AttnSegSeq2Seq takes overall 11 hours 40 minutes for 80 epochs in a 'Titan X' 12GB GPU memory, 3584 GPU Cores, 62GB RAM and Intel Xeon CPU E5-2620 2.40GHz system. For SegSeq2Seq, it takes 7 hours under the same system configurations.

4.1.2 Discussion

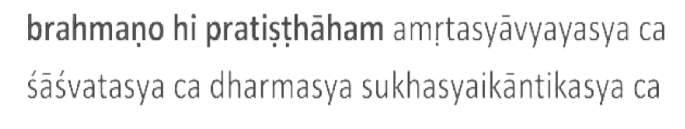
The Seq2Seq model of ours, with GibberishVocab as its vocabulary, outperforms the previous models by a huge margin. But at the same time, it restricts us to model the segmentation task with further downstream tasks jointly. Joint modelling of multiple related tasks [Tsa06], or more recently multitask learning approaches [SG16], has shown to benefit the performance of downstream tasks as compared to pipeline models. In our case, since we learn a new vocabulary altogether, the real word boundaries are opaque to the system. The decoder predicts from its vocabulary. However, morphological parsing, the next immediate downstream task in a typical NLP pipeline, requires the knowledge of exact word boundaries. Models that use linguistic resources are at an advantage here. Here, we overlook this aspect of segmentation as a preliminary task and instead see it as an end in itself. So we achieve scalability at the cost of missing out on providing valuable linguistic information. Nevertheless, the use of new vocabulary can mitigate the effect of OOV words, and in effect, this makes the Seq2Seq model capable of open vocabulary string transductions [SHB16].
In this work, we presented a model for word segmentation in Sanskrit using a purely statistical approach. Our model with attention outperforms SupervisedPCRW [KSS+16a], the then state of the art model. Our experiments, in line with the measures reported in Krishna et al. [KSS+16a], show that our system performs robustly across strings of varying word sizes.

4.2 Reusing OCRs for Post-OCR Text Correction in Romanised Sanskrit

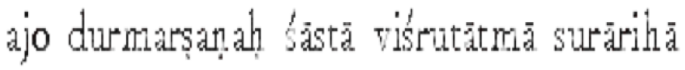
Sanskrit is primarily a spoken language with a rich oral tradition. This implies that the preferences for a writing system, more specifically the script, varied across the regions as well as over time. With the advent of the printing press, Devanāgari (देवनागरि) emerged as the prominent script for representing Sanskrit. With the standardisation of Romanisation using IAST in 1894 [MW99], printing in Sanskrit was extended to roman scripts as well. There has been a surge in digitising printed Sanskrit manuscripts written in the Roman srcipt, such as the ones currently digitised by the 'Krishna Path' project[2].
In this work, we propose a model for post-OCR text correction for Sanskrit texts written

[2] http://www.krishnapath.org/library/

in Roman script.[3] Post-OCR text correction, which can be seen as a special case of spelling correction [SEDDG16], is the task of correcting errors that tend to appear in the output of the OCR in the process of converting an image to text. The errors incurred from OCR can be quite high due to numerous factors, including typefaces, paper quality, scan quality, etc. The text can often be eroded, can contain noises, and the paper can be bleached or tainted as well [SEDDG16]. Figure 4.3 shows the sample images we have collected for the task. OCR systems are usually trained to predict orthographically and visually similar characters. However, these systems have limited capacity for capturing distributional information. Such systems might lead to incorrect word boundaries or incorrect graphemes due to this, as shown in Figure 4.4 for such cases. Here, incorporating a post OCR correction model with language specific distributional information can alleviate such errors.



(a) Bhagavad Gītā



(b) Sahaśranāma

Figure 4.3 Sample images from our test set with different stylistic parameters.

Ground Truth: kāmahā kāmakṛt kāntaḥ kāmaḥ kāmapradaḥ prabhuḥ
OCR Output: kāmahi kāmakpthāntal kimah kamapeadah prablhuh

Ground Truth: tāsāṁ brahma mahad yonir ahaṁ bīja-pradaḥ piṭā
OCR Output: tasambrahma mahad yoniraham bija-pradah pita

Figure 4.4 OCR outputs where word boundaries are detected improperly.

In the case of Indic OCRs, there have been considerable efforts in collection and annotation of data pertaining to Indic Scripts [KJ07, BMK+04, GS09, KSSJ14]. Earlier attempts on Indian scripts were primarily based on handcrafted templates [GS90, CP97] or features [ABN+10,PWK09] which extensively used the script and language-specific information [KSSJ14]. Sequence labelling approaches were later proposed, that take the word level

[3]The data and the codes for our system are available here - https://github.com/majumderb/sanskrit-ocr

inputs and make character level predictions [SPS08, Hel15a]. The word based sequence labelling approaches were further extended to use neural architectures, especially using RNNs and its variants such as LSTMs and GRUs [SJ12a, KSSJ14, SAC+17, ASA+18, MSJ16]. But, OCR is putative in exhibiting few long-range dependencies [SEDDG16]. Singh and Jawahar [SJ15] find that extending the neural models to process the text at the sentence level leads to improvement in the performance of the OCR systems. This was further corroborated by Saluja et al. [SAC+17] where the authors found that using words within a context window of 5 for a given input word worked particularly well for the Post-OCR text correction in Sanskrit. In the case of providing a text line as input, we are essentially providing more context about the input in comparison to the word level models and the RNN (or LSTM) cells are powerful enough to capture the long-term dependencies. Use of a sequence level approach rather than a word-level approach is more of a necessity for languages like Sanskrit, as the word boundaries in the sentences might be obscured due to *Sandhi*. Hence, in our case, we assume an unsegmented sequence as our input, and then we perform our Post-OCR text correction on the text. We hypothesise that this will improve the segmentation process and other downstream tasks for Sanskrit in a typical NLP pipeline. Our major contributions are:

1. Contrary to what is observed in Schnober et al. [SEDDG16], an encoder-decoder model, when equipped with copying mechanism [GLLL16], can outperform a traditional sequence labelling model in a monotone sequence labelling task. Our model outperforms Schnober et al. [SEDDG16] in the Post-OCR text correction for Romanised Sanskrit task by 7.69 % in terms of CRR.

2. By making use of digitised Sanskrit texts, we generate images as synthetic training data for our models. We systematically incorporate various distortions to those images so as to emulate the settings of the original images.

3. Through a human judgement experiment, we asked the participants to correct the mistakes from the predicted output from the competing systems. We find that the participants were able to correct predictions from our system more frequently, and the corrections were done much faster than the CRF model by Schnober et al. [SEDDG16].

4.2.1 Model Architecture

Availability of OCR Solutions for Romanised Sanskrit: Currently, there exist limited options for recognising Romanised Sanskrit texts from scanned documents. Possibly, the commercial OCR offering by Google as part of their proprietary cloud vision API and the SanskritOCR[4] might be the only two viable options. SanskritOCR essentially is an online interface to the Tesseract OCR, an open source multilingual OCR [Smi07, SAL09, Smi87], trained specifically for recognising Romanised Sanskrit. Additionally, we trained an offline version of Tesseract to recognise the graphemes in the Romanised Sanskrit alphabet. In both the Tesseract based models, we find that many scanned images, especially similar to the one shown in Figure 4.3b, were not recognised by the system. We hypothesise this to be due to lack of enough font styles available in our collection, in spite of using a site with the richest collection of Sanskrit fonts[5]. Instead, we chose to use Tesseract[6], with models trained for other languages written in Roman script. All the Latin or Roman scripts in the pre-trained models of Tesseract are trained on 400,000 text-lines spanning about 4,500 fonts[7].

Language	**Bhagavad Gītā**				**Sahasranāma**				**Combined**
	CRR	Ins	Del	Sub	CRR	Ins	Del	Sub	CRR
English	84.92	23	63	1868	64.06	73	696	1596	80.08
French	84.90	21	102	1710	63.91	91	702	1670	80.04
Finnish	82.61	15	141	1902	61.31	80	730	1821	78.81
Italian	83.45	20	73	1821	62.19	84	690	1673	79.03
Irish	84.52	12	78	1810	63.81	72	709	1841	79.93
German	84.40	33	72	1821	63.79	87	723	1874	79.12

Table 4.2 OCR performances for different languages with overall CRR, total Insertion, Deletion and Substitution errors.

Use of OCR with Pre-trained Models for Other Languages: French alphabet has the highest grapheme overlap with that of the Sanskrit alphabet (37 of 50), while all other languages have one less grapheme common with Sanskrit. Hence, we arbitrarily take 5 of

[4] https://sri.auroville.org/projects/sanskrit-ocr/. It provides interface to tesseract and Google OCR as well.

[5] http://pratyatosa.com/?P=41

[6] Our motivation to choose Tesseract over Google OCR, is due to the consideration of the fact that working with a commercial offering from Google OCR may not be an affordable option for various digitisation projects.

[7] https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract-4.00

the languages in addition to French and perform our analysis. Table 4.2 shows the character recognition rate (CRR) for OCR using alphabets of different languages when performed on a dataset of 430 scanned images (Section 4.2.2). The table also shows the count of error types made by the OCR after alignment [JKS07, DGG16]. All the languages have a near similar CRR with English and French leading the list. Based on our observations on the OCR performance, we select English for our further experiments.

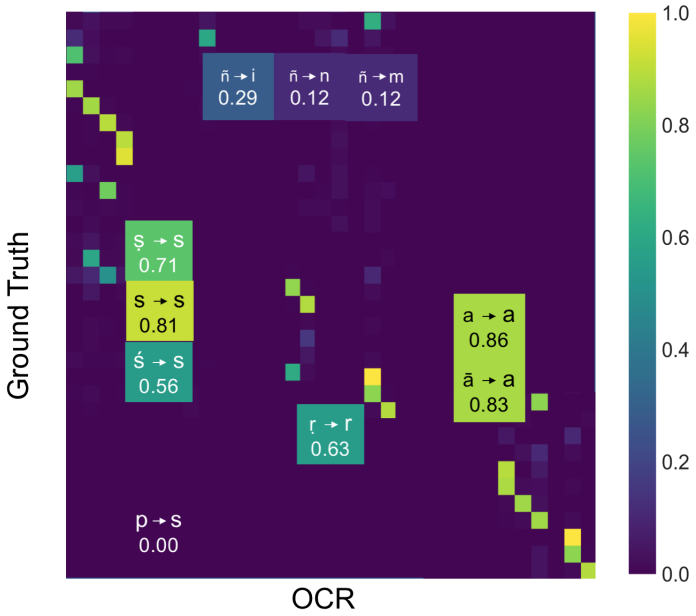


Figure 4.5 Heatmap of occurrences of majorly confusing character pairs between Ground Truth and OCR.

Upcycling such a pre-trained model brings its challenges. For instance, the missing 14 Sanskrit graphemes[8] in English are naturally mispredicted to other graphemes. This leads to ambiguity as the correct, and the mispredicted characters now share the same target. Figure 4.5 shows the heat-map for such mis-predictions when we used the OCR on the set of 430 scanned images. Here, we zoom the relevant cases and show the row-normalised proportion of predictions.[9]

[8]The missing graphemes are 'ā, ī, ū, ṛ, ḷ, ḹ, ṃ, ḥ, ṭ, ś, ṣ, ḍ, ṇ and ñ'.

[9]A more detailed figure with all the cases are available at https://github.com/majumderb/sanskrit-ocr/tree/master/heatmaps

System Descriptions

We formalise the task as a monotone Seq2Seq model. We use an encoder-decoder framework that takes in a character sequence as input, and the model finds embeddings at a sub-word level both at the encoder and decoder side. Here the OCR output forms input to the model. Keeping the task in mind, we make two design decisions for the model. One is the use of copying mechanism [GLLL16], and other is the use of Byte Pair Encoding (BPE) [SHB16] to learn a new vocabulary for the model.

CopyNet [GLLL16]: Since there may be reasonable overlap between the input and output strings, we use the copying mechanism as mentioned in CopyNet [GLLL16]. The model essentially learns two probability distributions, one for generating an entry at the decoder and the other for copying the entry from the encoder. We henceforth refer to these probability distributions as 'copy' probability and 'generate' probability respectively. The final prediction is based on the sum of both the copy and generate probabilities for the class. Given an input sequence $\boldsymbol{X} = (\boldsymbol{x_1}, ..., \boldsymbol{x_N})$ we define $\mathcal{X}$, for all the *unique* entries in the input sequence. We also define the vocabulary $\mathcal{V} = \{v_1, ..., v_N\}$. Let the out-of-vocabulary (OOV) words be represented with UNK. The probability of the generate mode g and copy mode c are given by

$$p(\boldsymbol{y_t}, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(\boldsymbol{y_t})}, & \boldsymbol{y_t} \in \mathcal{V} \\ 0, & \boldsymbol{y_t} \in \mathcal{X} - \mathcal{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & \boldsymbol{y_t} \notin \mathcal{V} \cup \mathcal{X} \end{cases}$$

$$p(\boldsymbol{y_t}, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:\boldsymbol{x_j} = \boldsymbol{y_t}} e^{\psi_c(\boldsymbol{x_j})}, & \boldsymbol{y_t} \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases}$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode (g) and copy-mode (c), respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{\boldsymbol{x} \in \boldsymbol{X}} e^{\psi_c(\boldsymbol{x})}$. The scoring function for both the modes, respectively, are

$$\psi_g(\boldsymbol{y_t} = v_i) = \boldsymbol{v}_i^\top \boldsymbol{W_o} \boldsymbol{s_t}, \quad v_i \in \mathcal{V} \cup \text{UNK}$$

$$\psi_c(\boldsymbol{y_t} = \boldsymbol{x_j}) = \sigma\left(\mathbf{h}_j^\top \mathbf{W}_c\right) \mathbf{s}_t, \quad \boldsymbol{x_j} \in \mathcal{X}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$, and σ is a non-linear activation function [GLLL16].

BPE [SHB16]: Sanskrit is a morphologically rich language. A noun in Sanskrit can have 72 different inflections, and a verb may have more than 90 inflections. Additionally, Sanskrit corpora generally express a compound rich vocabulary [KSS+16b]. Hence, in a typical Sanskrit corpus, the majority of the tokens appear less than five times (Section 4.2.2). These are generally considered to be rare words in a corpus [SHB16]. However, corpora dominated by rare words are challenging to handle for a statistical model like ours. To combat the sparsity of the data, we convert the tokens into sub-word n-grams using Byte Pair Encoding (BPE) [SHB16]. Methods such as wordpiece [SN12] as well as BPE [SHB16] are means of obtaining a new vocabulary for a given corpus. Every sequence in the corpus is then re-written as a sequence of tokens in terms of the sub-word units which forms the type in the new vocabulary so obtained. These methods essentially use a data-driven approach to maximise the language-model likelihood of the training data, given an evolving word definition [WSC+16].
We explicitly set the minimum count for a token in the new vocabulary to appear in the corpora as 30. We learn a new vocabulary of size 82 with 22 of them having a length one and the rest with a length 2. The IAST standardisation of the Romanised Sanskrit contains 50 graphemes in Sanskrit alphabet. About 12 of the graphemes are represented using a combination of 2 roman characters. Now, in the vocabulary learnt using BPE, 7 of the graphemes were not present. Hence, we add them in addition to the 82 entries learnt as vocabulary. This makes the total vocabulary to be 89. By using the new vocabulary, it is guaranteed that there will be no Out Of Vocabulary (OOV) words in our model.
We use three stacked layers of LSTM at the encoder and the decoder with the same settings as in Bahdanau et al. [BCB15]. To enable copying, we share the embeddings of the source and the target vocabulary. By eliminating OOV, we make sure that copying always happens by virtue of the evidence from the training data and not by the presence of an OOV word.

4.2.2 Experiments

Dataset

Sanskrit is a low-resource language. It is extremely scarce to obtain datasets with scanned images and the corresponding aligned texts for Romanised Sanskrit. We obtain 430 scanned images, as shown in Figure 4.3 and manually annotate the corresponding text. We use this as our test dataset, henceforth to be referred to as *OCRTest*. For training, we synthetically generate images from digitised Sanskrit texts and use them as our training set

Process	Parameters	Range		Step size
Gamma Correction (GM)	gamma (γ)	4	64	4
Salt & Pepper Noise (SPN) (with 50% salt and 50% pepper)	percentage of pixels corrupted	0.1%	1%	0.1
Gaussian Noise (GN) (mean =0)	standard deviation	2.5	3.5	0.25
Erosion (E) (one iteration)	kernel size (m×m)	2	5	1
Horizontal perspective distortion (HPD)	image width by image height	0.3	1	0.05

Table 4.3 Image pre-processing steps and parameters.

and development set. The images for training, *OCRTrain*, were generated by synthetically adding distortions to those images to match the settings of the real scanned documents.

OCRTest: It contains 430 images from 1) scanned copy of Vishnu Sahasranāma[10] (सहस्रनाम) and 2) scanned copy of Bhagavad Gītā (भगवद् गीता), a sample of each is shown in Figure 4.3a and 4.3b. 140 out of these 430 are from Sahasranāma, and the remaining are from Bhagavad Gītā.

OCRTrain: Similar to Ul-Hasan and Breuel [UHB13], we synthetically generate the images, which are then fed to the OCR, to obtain our training data. We use the digitised text from Śrīmad Bhāgavatam[11] (श्रीमद् भागवतम्) for generating the synthetic images. The text contains about 14,094 verses in total, divided into 50,971 text-lines. The dataset is divided into 80-20 split as the training set and the development set, respectively. The corpus contains a vocabulary of 52,882 word types. 48,249 of the word types in the vocabulary appear less than or equal to 5 times, of which 32,411 appear exactly once. This is primarily due to the inflectional nature of Sanskrit. We find similar trends in the vocabulary of Rāmāyaṇa[12] and Digital Corpus of Sanskrit [Hel16a] as well.

Synthetic Generation of Training-set

Using the text-lines from Bhāgavatam, we generate synthetic images using ImageMagick[13]. The images were generated with a quality of 60 Dots Per Inch (DPI). The number of pixels

[10] http://kirtimukha.com/
[11] https://www.vedabase.com/en/sb
[12] https://sanskritdocuments.org/sites/valmikiramayan/
[13] https://www.imagemagick.org/script/index.php

along the height for each textline was kept constant at 65 pixels. We add several distortions to the synthetically generated images to visually match the same settings as that of *OCRTest*. Table 4.3 shows the different parameters, namely, gamma correction, noise addition, use of structural kernel for erosion and perspective distortion, that we apply sequentially on the images so as to distort and degrade the images [CYJ+14]. We use grid search for the parameter estimation for these processes, where those parameters and the range of values experimented with are provided in Table 4.3. Finally, we filter 7 (out of 38,400 combinations) different configurations based on the distribution of Character Recognition Rate (CRR) across the images compared with that of the *OCRTest* using KL-divergence. Among these seven configurations, four are closer to the settings for Bhagavad Gītā and the remaining three for Sahasranāma. Figure 4.6 shows the two different settings (closer to each of the source textbook) for the string *"ajo durmarṣaṇaḥ śāstā viśrutātmā surārihā"* (अजो दुर्मर्षणः शास्ता विश्रुतात्मा सुरारिहा), along with their corresponding parameter settings and KL-Divergence. Our training set contains images from all the seven settings for each of the textline in OCRTrain[14].

Figure 4.6 Samples of synthetically generated images. The parameter settings for the distortions are mentioned below the corresponding image.

Evaluation Metrics: We use three different metrics for evaluating all our models. We use Character Recognition Rate (CRR), and Word Recognition Rate (WRR) averaged over each of the sentences in the 430 lines in the test dataset [SJ12a]. CRR is the fraction of characters recognised correctly against the total number of characters in a line, whereas WRR is the fraction of words correctly recognised against the total number of words in

[14]Samples of all the seven configurations are shown in https://github.com/majumderb/sanskrit-ocr/blob/master/supplementary_CoNLL2018.pdf

a line. Additionally, we use a sentence level metric, called the acceptability score. The measure indicates the extent to which a sentence is permissible or acceptable to the speakers of the language [LCL15]. From Lau et al. [LCL15], we use the *NormLP* formulation for the task, as it is found to have a high correlation with the human judgements in evaluating acceptability. NormLP is calculated by obtaining the likelihood of a predicted sentence as per the model, and then normalising it by the likelihood of the string as per a unigram language model trained on a corpus with gold standard sentences. A negative sign is then given to the score. The higher the score, the more acceptable the sentence is.

Baselines

Character Tagger - Sequence Labelling Using BiLSTMs: This is a sequence labelling model which uses BiLSTM cells and input is a character sequence [SAC$^+$17]. We use categorical cross-entropy as the loss function and softmax as the activation function. For dropout, we employ spatial dropout in our architecture. The model consists of 3 layers, with each layer having 128 cells. Embeddings of size 100 are randomly initialised, and the learnt representations are stored in a character look-up table similar to Lample et al. [LBS$^+$16]. In addition to every phoneme in Sanskrit as a class, we add an additional class 'no change' which signifies that the character remains as is. We also experimented with a variant where the final layer is a CRF layer [LMP01]. We henceforth refer to both the systems as *BiLSTM* and *BiLSTM-CRF*, respectively.

Pruned CRFs [SEDDG16]: They are higher order CRF models [Ish11] that approximate the CRF objective function using coarse-to-fine decoding. Schnober et al. [SEDDG16] adapt the sequence labelling model as a Seq2Seq model that can handle variable length input-output pairs. Schnober et al. [SEDDG16] show that none of the neural Seq2Seq models considered in their work were able to outperform the Pruned CRF with order-5. The features to the model are consecutive characters within a window of size w in either of the directions of the current position at which a prediction is made. The model is designed to handle 1-to-zero and 1-to-many matches, facilitated by the use of alignment prior to training. We consider all the three settings reported in Schnober et al. [SEDDG16] and report the results for the best setting. The order-5 model, which uses 6-grams within a window of 6, performs the best. Henceforth, this model is referred to as *PCRF-Seq2Seq* (also referred to as PCRF interchangeably).

Encoder-Decoder Models: For the Seq2Seq model [SVL14], we use three stacked layers

of LSTM each at the encoder and the decoder. Each layer is of 128 dimensions, and weighted cross-entropy is used as the loss. We also add residual connections among the layers in a stack [WSC+16]. To further capture the entire input context for making each prediction at the output, we make use of attention [BCB15], specifically Luong's attention mechanism [LPM15]. We experiment with two variants where *EncDec+Char* uses character level embeddings, and *EncDec+BPE* uses embeddings with BPE.

CopyNet+BPE: The model discussed in Section 4.2.1. We use CopyNet+BPE and CopyNet interchangeably throughout the Section 4.2.

Results

Model	**Bhagavad Gītā**		**Sahasranāma**		**Combined**	
	CRR	WRR	CRR	WRR	CRR	WRR
OCR	84.81%	64.40%	35.76%	0.65%	77.88%	23.84%
BiLSTM	93.79%	68.60%	61.31%	7.28%	85.23%	45.60%
BiLSTM-CRF	94.68%	68.60%	65.31%	7.28%	85.82%	45.60%
PCRF-Seq2Seq	**96.87%**	70.56%	81.77%	9.34%	87.94%	57.17%
EncDec+Char	91.48%	68.00%	63.63%	15.74%	82.51%	47.37%
EncDec+BPE	90.92%	68.00%	61.53%	15.74%	83.14%	45.98%
CopyNet+BPE	**97.01%**	**75.21%**	**87.01%**	**33.47%**	**89.65%**	**68.71%**

Table 4.4 Performance in terms of CRR and WRR for all the competing models.

Table 4.4 shows the results for all the competing systems based on the predictions from *OCRTest*. CopyNet performs the best among the competing systems on both the source texts and across both the OCR related evaluation metrics, CRR and WRR. The acceptability scores, as shown in Table 4.5, show similar trends in the system performances. For the relatively cleaner Gītā dataset, the models CopyNet and PCRF-Seq2Seq report similar performances. However, Sahasranāma is a noisier dataset, and we find that CopyNet outperforms all other models by a huge margin. The WRR for the system is double that of the next best system (EncDec) on this dataset.

System Performances for Various Input Lengths: From Figure 4.7a, it can be observed that the performance in terms of CRR for CopyNet and PCRF is robust across all the lengths

Model	**Bhagavad Gītā**	**Sahasra-nāma**	**Combined**
	Norm LP		
OCR	–	–	–
BiLSTM	-0.553	-1.292	-0.852
BiLSTM-CRF	-0.548	-1.281	-0.847
PCRF-Seq2Seq	-0.227	-1.216	-0.803
EncDec+Char	-0.542	-1.321	-0.865
EncDec+BPE	-0.496	-1.384	-0.842
CopyNet+BPE	**-0.165**	**-0.856**	**-0.551**

Table 4.5 Performance of the systems in terms of Norm LP (acceptability) scores.

	CRR	WRR
Bhagavad Gītā	96.80%	71.23%
Sahasra-nāma	82.81%	26.01%
Combined	87.88%	60.91%

Table 4.6 Performance in terms of CRR, WRR for Google OCR.

on strings from Gītā and never goes below 90%. For Sahasranāma, as shown in Figure 4.7b, CopyNet outperforms PCRF across inputs of all the lengths except for one setting. But, in the case of WRR, CopyNet is the best performing model across all the lengths, as shown in Figure 4.7d.

Performance Comparison to Google OCR: Google OCR is probably the only available OCR that can handle Romanised Sanskrit. We could not find technical details about the architecture of the OCR used, and whether the service employs a post-OCR text correction module. We empirically compare the performance of Google OCR on *OCRTest* with our model. Table 4.6 shows the results for Google OCR. Overall we find that CopyNet outperforms Google OCR across all the metrics. We find that Google OCR reports a similar CRR for Gītā with that of ours, but still reports a lower WRR than ours. The system performs better than PCRF in all the metrics other than CRR for Gītā.

Error Type Analysis: In Table 4.7, we analyse the reduction in specific error types for PCRF and CopyNet after the alignment of the predicted string with that of the ground truth in terms of insertion, deletion and substitution. We also report the system induced errors, where a correct component at the input (OCR output) is mispredicted to a wrong output by the model. CopyNet outperforms PCRF in correcting the errors, and it also introduces a lesser number of errors of its own. Both CopyNet and PCRF [SEDDG16] are Seq2Seq

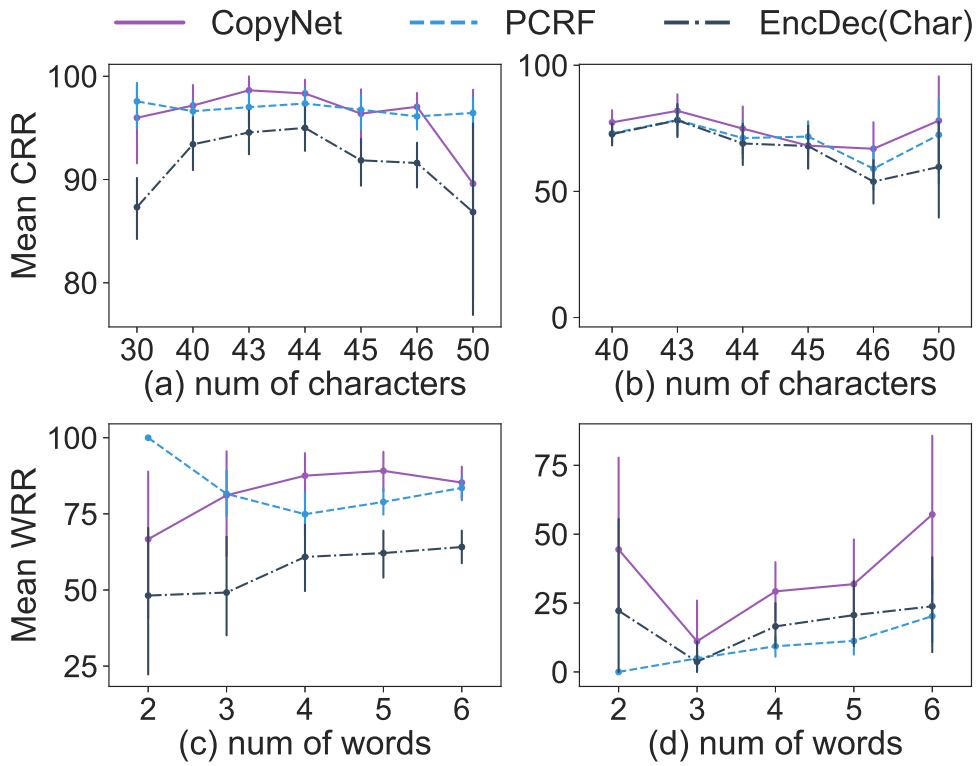


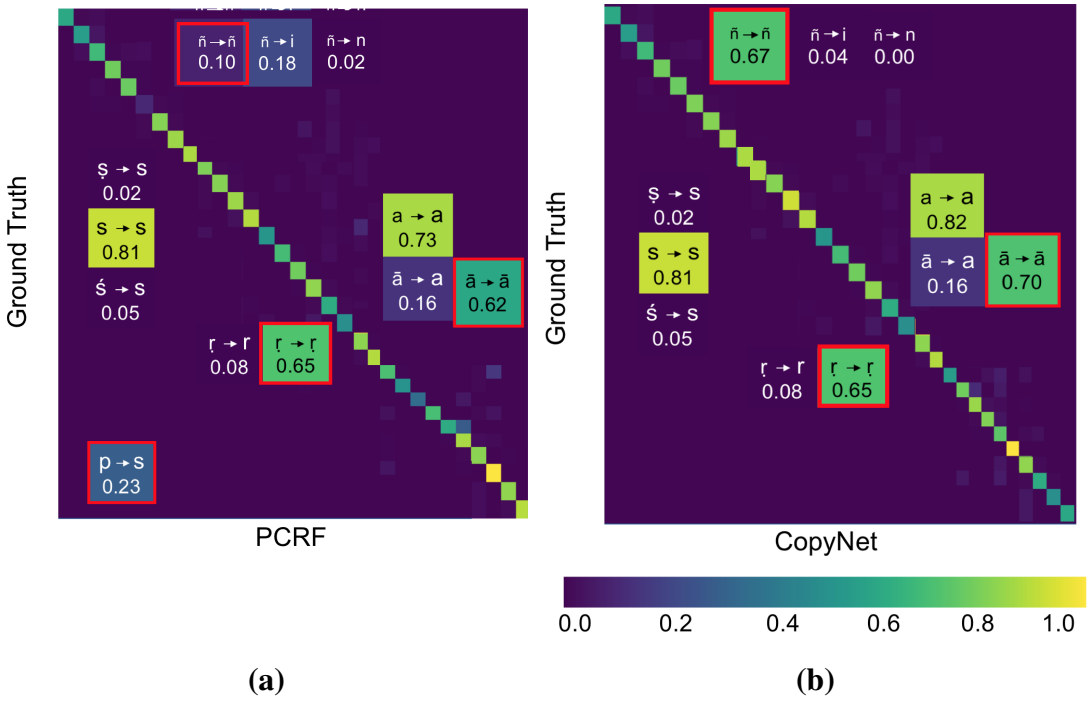
Figure 4.7 (a) and (b) show CRR for Gītā and Sahasranāma respectively, for the competing systems. (c) and (d) show WRR for Gītā and Sahasranāma, respectively. All the entries with insufficient data-points were merged to the nearest smaller number.

models and can handle varying length input and output. Both the systems perform well in handling substitution errors, the type which dominated the strings in *OCRTest*, though neither of the systems was able to correct the insertion errors. Insertion can be seen as a special case of 1-to-many insertion matches, which both systems are ideally capable of handling. We see that for Sahasranāma, CopyNet corrects about 17.24 % of the deletion errors as against <5% of the deletion errors corrected by PCRF.

Since there exist 14 graphemes in Sanskrit alphabet which are not present in the English alphabet, all 14 of them get substituted to a different grapheme by the OCR. While most of them get substituted to an orthographically similar character such as ā → a and ḥ → h, we find that ñ → i does not fit the scheme, as shown in Figure 4.5. In the majority of the cases, CopyNet predicts them to the correct grapheme. But PCRF still fails to correct the OCR induced confusion for ñ → i in the majority of the instances. Additionally, we find that PCRF introduces its own errors, for instance, it often mispredicts p → s. Figure 4.8 shows the overall variations in both the systems as compared to Figure 4.5 for OCR induced errors.

Model	Bhagavad Gītā			Sahasranāma			System errors		
	Ins	Del	Sub	Ins	Del	Sub	Ins	Del	Sub
OCR	23	63	1868	73	696	1596	–	–	–
PCRF	22	57	641	72	663	932	0	73	209
CopyNet	22	**45**	629	72	**576**	561	10	5	52

Table 4.7 Insertion, Deletion and Substitution errors for OCR, PCRF and CopyNet modes for both the datasets. The system errors are extra errors added by the respective systems.



(a) **(b)**

Figure 4.8 Heatmap for occurrences of majorly confusing character pairs between ground truth and predictions of (a) PCRF model (b) CopyNet model.

Copy or Generate? For the 14 graphemes, missing at the encoder (input) but present at the decoder side during training, those predictions have to happen with high values of the 'generate' probability in general. We find that not only the average 'generate' probability for such instances is high, but also the 'copy' probability is extremely low. The heatmap of the average copy and generate scores for 6 of the 14 graphemes are shown in Figure 4.9. For the remaining cases, we find that both the 'generate' and the 'copy' probability are higher. However, it needs to be noted that the prediction is made generally by using both the distributions and the distributions are not complementary to each other. A similar trend

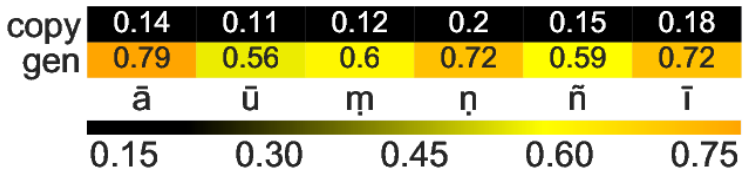


Figure 4.9 Heatmap of mean copy score (copy) and mean generate score (gen), respectively for 6 (of 14) graphemes not present in the English alphabet.

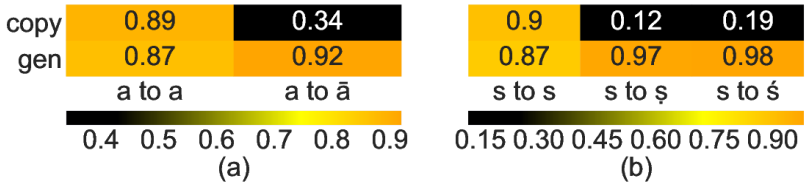


Figure 4.10 Mean copy and generate scores for different predictions from (a) 'a' and (b) 's'.

can be observed in Figure 4.10 as well. For example, in the case of a $\rightarrow$ ā, only the generate probability is high. But, for a $\rightarrow$ a, both the copy and generate probability scores are high.

Effect of BPE and Alphabet in the Vocabulary: We further investigate the effect of our vocabulary, which is the union of the alphabet in Romanised Sanskrit and what is learnt using BPE (BPE+Alphabet). For comparison, we train a model with only the alphabet as vocabulary and find the CRR and WRR for the combined test sentences to be 86.1% and 66.09%, respectively. Similarly, the BPE+Alphabet vocabulary setting performs better than a model that takes word level input. The word level model shows a drop in the performance with a CRR and WRR of 86.42% and 66.54%, respectively. When using the original BPE vocabulary, we report a CRR and WRR of 89.53% and 68.11%, respectively.

Image Quality: Our training set was generated with a quality of 60 DPI for the images. We generate images corresponding to strings in *OCRTrain* with DPI of 50 to 300 in step sizes of 50 for a sample of 500 images. We use noise settings, as shown in Figure 4.6. The OCR output of the said strings remained as is with that of the one generated with a DPI of 60. This experiment can be seen as a proxy in evaluating the robustness of the OCR to various scanning qualities of input. Our choice of DPI as 60 was based on the lowest setting we observed in digitisation attempts in Sanskrit texts.

Effect of Adding Distortions to the Synthetically Generated Images: Table 4.4 shows the system performance after training our model on the data which was synthetically generated, as per the procedure we discussed previously. Here, we make an implicit assumption that we can have access to a sample of textline images annotated with the corresponding text from the manuscript for which the Post-OCR text correction needs to be performed. This also mandates retraining the model for every new manuscript. We attempted for a more generalised version of our model, by using training data where the image generation settings are not inspired by the target manuscript for which the task needs to be performed. Using the settings from [CYJ+14] for inducing noise, we generated ten random noise configurations. Here the step sizes were fixed at values such that each parameter, except erosion (E), can assume five values each uniformly spread across the corresponding ranges considered. From a total of 2500 ($5\times5\times5\times5\times4$) configuration options, ten random settings were chosen. Every textline was generated with each of the ten different settings. The resulting model using CopyNet produced a CRR of 89.02% (96.99% for Gītā and 85.62% for Sahasranāma) on the test set, which is close to the reported CRR of 89.65 in Table 4.4. The noise ranges chosen are used directly from [CYJ+14] and are not influenced by the test data in hand.
We also experimented with a setting where no noise was added to the synthetically generated images and the images were fed to the OCR. We obtained a CRR of 80.12% from OCR, where the errors arose mostly from the missing graphemes in the alphabet getting mispredicted to a different grapheme. CopyNet after training with the text so generated reported a CRR of 86.81% (96.01% for Gītā, 75.78% for Sahasranāma) on the test data.

Human Judgement Survey: In this survey, we evaluate how often a human can recognise the correct construction by viewing only the prediction from one of the systems. We also evaluate how fast a human can correct them. We selected 15 constructions from Sahasranāma, the noisier dataset, and obtained the system outputs from the OCR, CopyNet and PCRF for each of these. The average length of a sentence is 41.73 characters, all ranging between 23 and 47 characters. A respondent is shown a system prediction (system identity anonymised) and is asked to type the corrected string without referring to any sources. A respondent gets 15 different strings altogether, five each from each of the three systems. We consider responses from 9 participants where all of them at least have an undergraduate degree in Sanskrit linguistics. Altogether from 3 sets of questionnaires, we have 45 strings (3 outputs for a given string). Every string obtained three impressions. We find that a participant on an average

could identify 4.44 sentences out of 5 from the CopyNet, while it was only 3.56 for PCRF and 3.11 for the OCR output. The average time taken to complete the correction of a string was 81.4 seconds, 106.6 seconds and 127.6 seconds for CopyNet, PCRF and OCR, respectively. In this work, we proposed a Seq2Seq model which uses the copying mechanism for the Post-OCR text correction task in Sanskrit. We find that the use of the copying mechanism in our model results in better performance than other Seq2Seq models for the task. From our experiments, we find that CopyNet performs stably even for OCR outputs with a CRR as low as 36%. Our model has shown consistent and significant improvement in its results across various experimental settings, as compared to the baseline models. It even outperformed the commercially available Google OCR on two different sets of test data, on which we tested both the systems.

4.3 Poetry to Prose Conversion in Sanskrit as a Linearisation Task

Prosody plays a key role in the word arrangement in Sanskrit Poetry. The word arrangement in a verse should result in a sequence of syllables which adhere to one of the prescribed meters in Sanskrit Prosody [SASG15]. As a result, the configurational information of the words in a verse is not aligned with its verbal cognition [Bha90, Den05]. Obtaining the proper word ordering, called as the prose ordering, from a verse is often considered a task which requires linguistic expertise [SKS$^+$16, KSSS15]. In this work, we use neural sequence generation models for automatic conversion of poetry to prose. Lack of sufficient poetry-prose parallel data is an impediment in framing the problem as a Seq2Seq task [GHDL18]. Hence, we formulate our task as that of a word linearisation task [HWGL09]. In linearisation, we arrange a bag of words into a grammatical and fluent sentence [LZCQ15]. This eliminates the need for parallel data, as the poetry order is not anymore relevant at the input. A neural-LM based model from Schmaltz et al. [SRS16] and a Seq2Seq model form Wiseman and Rush [WR16] are the current state of the art (SOTA) models in the linearisation task.
We first show that a Seq2Seq model with gated CNNs [GAG$^+$17], using a sequence level loss [EOA$^+$18] can outperform both the SOTA models for the Sanskrit poetry linearisation task. But using a Seq2Seq model brings non-determinism to the model as the final prediction of the system is dependent on the order at which the words are input at the encoder [VBK16].

We resolve this by using a pretraining approach [WCM18] to obtain an initial ordering of the words, to be fed to the final model. This approach consistently performs better than using the original poetry order as the input. Further, we find that generating multiple hypotheses[15] using this component [WCM18], to be fed to the final Seq2Seq component, results in improving the results by about 8 BLEU points. Additionally, we use a pretraining approach to learn task specific word embeddings by combining multiple word embeddings [KWC18]. We call our final configuration as *kāvyaguru* (काव्यगुरु). '*kāvyaguru*' is a compound word in Sanskrit, which roughly translates to 'an expert in prosody'.

4.3.1 Poetry to Prose as Linearisation

Given a verse sequence $x_1, x_2.....x_n$, our task is to rearrange the words in the verse to obtain its prose order. As shown in Figure 4.11, *kāvyaguru* takes the Bag of Words (BoW) S as the input to the system. We use two pretraining steps prior to the Seq2Seq component in our approach. The first step, 'DME', combines multiple pretrained word embeddings, say $\{w_{11}, w_{12}, w_{13}\}$ for a token $x_1 \in S$, into a single meta-embedding, w_1^{DME}. The second component, 'SAWO', is a linearisation model in itself, which we use to generate multiple hypotheses, i.e., different permutations of the tokens, to be used as input to the final 'Seq2Seq' component.

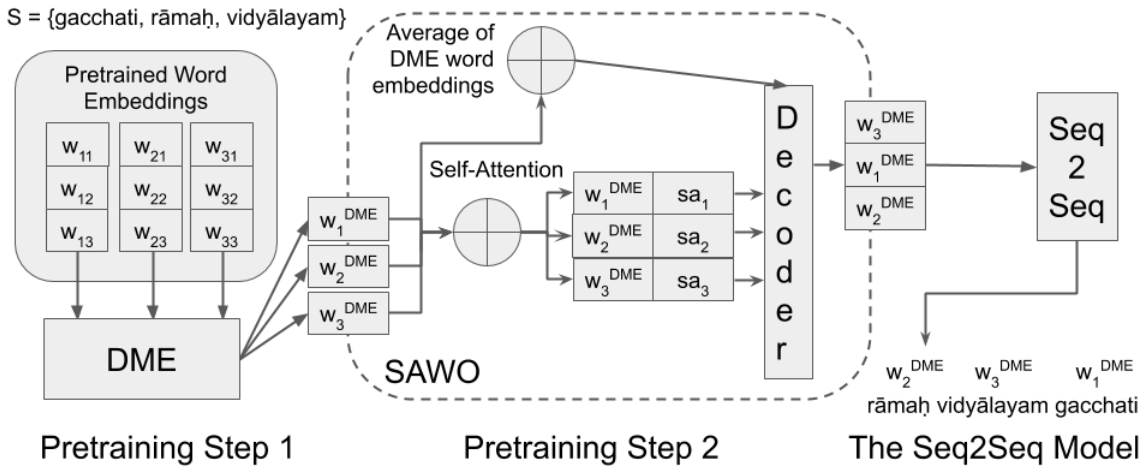


Figure 4.11 Configuration for *kāvyaguru*, demonstrated for a 3 word sentence with a prose order '*rāmaḥ vidyālayam gacchati*'. English translation: "Rāma goes to School". We show generation of only one hypothesis from SAWO.

[15]Empirically shown to be 10

Pretraining Step 1 – Dynamic Meta Embeddings (DME): Given a token $x_i \in S$, we obtain r different pre-trained word embeddings, represented as $\{w_{i1}, w_{i2}....w_{ir}\}$. Following Kiela et al. [KWC18], we learn a single task specific embedding, $\mathbf{w_i^{DME}}$ using weighted sum of all the r embeddings. The scalar weights for combining the embeddings are learnt using self-attention, with a training objective to minimise the negative log likelihood of the sentences, given in the prose order.

Pretraining Step 2 – Self-Attention Based Word-Ordering (SAWO): SAWO allows us to generate multiple permutations of words as hypotheses, which can be used as input to a Seq2Seq model. Here, we use a word ordering model itself as a pretraining step, proposed in Wang et al. [WCM18]. From step 1, we obtain the DME embeddings, $\{w_1^{DME}, w_2^{DME}, ...w_n^{DME}\}$, one each for each token in S. For each token in S, we also learn additional embeddings, $\{sa_1, sa_2, ...sa_n\}$, using the self-attention mechanism. These additional vectors are obtained using the weighted sum of all the DME embeddings in the input BoW S, where the weights are learned using the self-attention mechanism [WCM18, VSP+17]. As shown in Figure 4.11, the DME vector $\mathbf{w_i^{DME}}$ and the vector $\mathbf{sa_i}$ are then concatenated to form a representation for the token X_i. The concatenated vectors so obtained for all the tokens in S, form the input to the decoder.
We use an LSTM based decoder, initialised with the average of DME embeddings of all the tokens ($\{w_1^{DME}, w_2^{DME}, ...w_n^{DME}\}$) at the input. A special token is used as the input in the first time-step. Based on the predictions from the decoder, the concatenated vectors are input in the subsequent time-steps. The decoder is constrained to predict from the list of words in BoW, which are not yet predicted at a given instance. We use a beam-search based decoding strategy [SRS16] to obtain top-k hypotheses for the system.
For both the pretraining steps, the training objective is to minimise the negative log likelihood of the ground truth (prose order sentences), and both the components are trained jointly. The multiple hypotheses so generated are used as independent inputs to the Seq2Seq model, with the prose order as their corresponding ground truth for training. In Figure 4.11, we show only one hypothesis from SAWO. This helps us to obtain a k-fold increase in the amount of available training data.

The Seq2Seq Model: We use the Seq2Seq model comprising of gated CNNs [GAG+17] for the task. Our training objective is a weighted combination of the expected risk

minimisation ($RISK$) and the token level negative log likelihood with label smoothing ($TokLS$) [EOA+18]. Here, we use a uniform prior distribution over the vocabulary for label smoothing. $RISK$ minimises the expected value of a given cost function, BLEU in our case, over the space of candidate sequences.

$$\mathcal{L}_{Risk} = \sum_{\mathbf{u} \in \mathcal{U}} cost(\hat{\mathbf{y}}, \mathbf{u}) \frac{p(\mathbf{u}|\mathbf{x}')}{\sum_{\mathbf{u}' \in \mathcal{U}} p(\mathbf{u}'|\mathbf{x}')} \tag{4.1}$$

Here $\mathcal{U}$ is the candidate set, with $|\mathcal{U}| = 16$ and the sequences in $\mathcal{U}$ are obtained using Beam Search. The size for the beam search was determined empirically.[16] $\hat{\mathbf{y}}$ is the reference target sequence, i.e., the *prose*. $\mathbf{x}'$ is the input sequence to the model, which is obtained from SAWO. In $\mathcal{L}_{Risk}$, $cost(\hat{\mathbf{y}}, \mathbf{u}) = 1 - BLEU(\hat{\mathbf{y}}, \mathbf{u})$, where $0 \leq BLEU(\hat{\mathbf{y}}, \mathbf{u}) \leq 1$. Similar to Wiseman and Rush [WR16], we constrain the prediction of tokens to those available at the input during testing.

Majority Vote Policy: For an input verse, SAWO generates multiple hypotheses and Seq2Seq then predicts a sequence corresponding to each of these, of the same size as the input. To get a single final output, we use a 'Majority Vote' policy. For each position, starting from left, we find the token which was predicted the most number of times at that position among all the Seq2Seq outputs. We then choose it as the token in the final output.

4.3.2 Experiments

Dataset: We obtain 17,017 parallel poetry-prose data from the epic "*Rāmāyaṇa*".[17] Given that about 90 % of the vocabulary appears less than 5 times in the corpus, we use BPE to learn a new vocabulary [SHB16]. We add about 95,000 prose-order sentences from Wikipedia into our training data, as the poetry order input is irrelevant for linearisation. Here, we obtain sentences which are available only in its prose order from Wikipedia as auxiliary data for training. We obtain about 95,000 sentences from Wikipedia with an average of 7.63 words per sentence. We filter poetry verses from Wikipedia by matching them with the sentences in DCS[18], which is predominantly a poetry corpus. We also filter

[16] We experimented with beam sizes from 1 to 32, in powers of 2. Since the increase in beam size from 16 to 32 did not result in significant improvements in system performance, we set the beam size as 16.

[17] Filtered from 18,250 verses. The remaining were ignored due to corrupted word constructions.

[18] http://kjc-sv013.kjc.uni-heidelberg.de/dcs/index.php?contents=texte

the sentences (and adjacent 3 lines in either of the directions) which end with a double daṇda, an end marker specifically used for verses [Hel16b].

Data Preparation: With a vocabulary of 12,000, we learn embeddings for the BPE entries using Word2vec [MSC+13], FastText [BGJM17], and character embeddings from Nehrdich and Hellwig [HN18]. The embeddings were trained on 0.8 million sentences (6.5 million tokens) collected from multiple corpora, including DCS [Hel16a], Wikipedia and Vedabase[19]. Finally, we combine the word embeddings using DME [KWC18]. From the set of 17,017 sentences of parallel poetry-prose corpus, we use 13,000 sentence pairs for training, 1,000 for validation and the remaining 3,017 sentence pairs for testing. The sentences in test data are not used in any part of training or for learning the embeddings.

Evaluation Metrics: Linearisation tasks are generally reported using BLEU [PRWZ02] score [HST+17, BWE+11]. Additionally, we report Kendall's Tau (τ) and perfect match scores for the models. Perfect match is the fraction of sentences where the prediction matches exactly with the ground truth. Kendall's Tau (τ) is calculated based on the number of inversions needed to transform a predicted sequence to the ordering in the reference sequence. τ is used as a metric in sentence ordering tasks [Lap06], and is defined as $\frac{1}{m}\sum_{i=1}^{m} 1 - 2 \times inversions\ count/\binom{n}{2}$ [LLR18, Lap03]. In all these three metrics, a higher score always corresponds to a better performance of the system.

Baselines

LSTM Based Linearisation Model (LinLSTM): LinLSTM is an LSTM based neural language model (LM) proposed by Schmaltz et al. [SRS16]. Sequences in sentence/prose order are fed to the system for learning the LM. Beam search, constrained to predict only from the bag of words given as input, is used for decoding. The authors obtained SOTA results in their experiments on the Penn Treebank, even outperforming different syntax based linearisation models [ZC15, Zha13]. The best result for the model was obtained using a beam size of 512, and we use the same setting for our experiments.

Seq2Seq with Beam Search Optimisation (BSO): The Seq2Seq model uses a max-margin approach with a search based loss, designed to penalise the errors made during beam

[19] https://www.vedabase.com/en/sb

System	Augmentation	τ	BLEU	PM(%)
LinLSTM	Ramayaṇa dataset	61.47	35.51	8.22
	+ Wikipedia Prose	58.86	31.39	7.14
BSO	Ramayaṇa dataset	58.62	29.16	7.61
	+ Wikipedia Prose	65.38	41.22	12.97
	+ DME	68.45	44.29	19.69
	+ SAWO	72.89	52.37	24.56
kāvyaguru	Ramayaṇa dataset	59.27	31.55	8.62
	+ Wikipedia Prose	66.82	42.91	13.52
	+ DME	70.8	48.33	20.21
	+ SAWO	74.32	54.49	25.72
	+ Self-Attention	**75.58**	**55.26**	**26.08**

Table 4.8 Results for all the three competing models. The '+' sign indicates that the augmentation is added to the configuration in the row above it.

search [WR16]. Here scores for different possible sequences are predicted, and then they are ranked using beam search. The loss penalises the function when the gold sequence falls off the beam during training. For our experiments, we use a beam size of 15 for testing and 14 for training, the setting with the best reported scores in Wiseman and Rush [WR16].

Results

Table 4.8 provides the results for all the three systems under different settings. *kāvyaguru* reports the best results with a BLEU score of 55.26, outperforming the baselines. We apply both the pretraining components and the 'Majority Vote' policy (Section 4.3.1) to both the Seq2Seq models, i.e., 'BSO' and the proposed model '*kāvyaguru*'. From Table 4.8, it is evident that infusing prose-only training data from Wikipedia, and applying both the pretraining steps leads to significant[20] and consistent improvements for both the Seq2Seq models. LinLSTM shows a decrease in its performance when the dataset is augmented with sentences from Wikipedia. We obtain the best results for *kāvyaguru* when self-attention was added to the Seq2Seq component of the model [EOA$^+$18, PXS18] (final row in Table 4.8).

[20]For all the reported results, we use approximate randomisation approach for significance tests. All the reported values have a p-value < 0.02

Table 4.10 shows that the text-encoding/transliteration scheme in which a sequence is represented affects the results. *kāvyaguru* performs the best when it uses syllable level encoding of input, as compared to character level transliteration schemes such as IAST[21] or SLP1[22].

k	τ	BLEU	PM
1	71.14	48.26	20.15
5	74.15	53.74	25.02
10	75.58	55.26	26.08

Table 4.9 Results for *kāvyaguru* when trained (and at test-time) using different values of k at the SAWO pretraining step.

Encoding	τ	BLEU	PM
IAST	73.64	53.46	23.73
SLP1	73.79	53.91	24.16
Syllable	75.58	55.26	26.08

Table 4.10 Results for *kāvyaguru*, when using different sequence encoding schemes.

Effect of Increase in Training-set Size due to SAWO: Using SAWO, we can generate multiple word order hypotheses as the input to the Seq2Seq model. Results from Table 4.9 show that generating multiple hypotheses leads to improvements in the system performance.[20] It might be puzzling that *kāvyaguru* contains two components, i.e. SAWO and Seq2Seq, where both of them perform essentially the same task of word ordering. This might create an impression of redundancy in *kāvyaguru*. But, a configuration that uses only the DME and SAWO (without the Seq2Seq), results in a BLEU score of 33.8 as against 48.26 for *kāvyaguru* (Table 4.9, $k = 1$). Now, this brings the validity of SAWO component into question. To check this, instead of generating hypotheses using SAWO, we used 100 random permutations[23] for a given sentence as input to the Seq2Seq component. The first 3 rows of BSO and *kāvyaguru* in Table 4.8 show the results for non-SAWO configurations. These configurations do not outperform SAWO based configurations, in spite of using as many as 10 times the candidates than those used in SAWO based configuration. For SAWO (non-SAWO), we find that the system performances tend to saturate with number of hypotheses greater than 10 (100).

Effect of Using Word-order in the Verse During Inference: During inference, the test-set sentences are passed as input in the verse order to each of the *kāvyaguru* configurations

[21] https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

[22] https://en.wikipedia.org/wiki/SLP1

[23] Empirically decided from 1 to 100 random permutations with a step size of 10

in Table 4.8. *kāvyaguru*+DME configuration achieves the best result for this. But here also, the system performance drops to $\tau = 68.92$ and $BLEU = 45.63$, from 70.8 and 48.33, respectively. To discount the effect of majority vote policy used in SAWO, we consider predictions based on individual SAWO hypotheses. However, even the lowest τ score (70.61), obtained while using the 10th ranked hypothesis from SAWO, outperforms the predictions based on the verse order.[20]

In this work, we addressed the poetry to prose conversion problem by formalising it as an LM based word linearisation task. Our proposed model *kāvyaguru*, consists of a Seq2Seq model along with two pretraining approaches. In this task, *kāvyaguru* outperformed the state of the art models proposed originally for the word linearisation task. Through a series of experiments, we showed the effectiveness of each component of our model. Further, our experiments show that the use of our second pretraining component to generate a word ordering hypothesis as input to the Seq2Seq component results in an improved performance than using the original verse order as input to the Seq2Seq.

4.4 Summary

In this chapter, we investigated the efficacy of purely statistical neural sequence generation approaches for various sequence level structured prediction tasks in Sanskrit. We first showed that we could effectively train a Seq2Seq model for the word-segmentation task, which easily outperformed a linguistically involved setup such as Krishna et al. [KSS+16a]. We then experimented with two specific augmentations to a standard Seq2Seq model. For the Post-OCR Text correction, we augmented a standard Seq2Seq model to include copying mechanism into its architecture. For the poetry to prose linearisation task, we proposed gated-CNN based Seq2Seq model, which uses a sequence level loss. Since our solutions to these problems are language-agnostic, we hope to extend these models to other languages. For instance, the word-segmentation can be extended to handle Sandhi in other Indic languages such as Hindi, Marathi, Malayalam, Telugu, etc. In the case of Post-OCR text correction, our approach can easily be extended for other languages that use Roman Script. Currently, there are 135 languages which directly share the Roman alphabet, but only 35 of them have OCR system available.

A major challenge we face for the development of neural sequence models in Sanskrit is the

availability of task-specific labelled data for further downstream tasks.[24] To obtain sufficient training data for the post-OCR text correction task, we generated the data synthetically by generating images with distortions to replicate the real world scan settings. For the poetry to prose conversion task, we first formulated the problem as a linearisation task and then obtained prose only sentences as auxiliary training data. In future, we plan to explore transfer learning and multi task learning approaches that can be used to overcome the data scarcity for a low-resource morphologically rich language like Sanskrit. At the same time, in the next chapter, we propose a structured prediction framework using Energy based models for various sentence level tasks in Sanskrit. The framework facilitates to incorporate linguistic knowledge into its architecture. Using this, we were able to substantially reduce the task-specific training data requirements compared to neural sequence generation models.

[24] A probable exception to this will be the morphological tagging task

Chapter 5

A Graph Based Framework for Structured Prediction Tasks in Sanskrit

We propose a framework using energy based models (EBM) [LCH$^+$06] for solving multiple structured prediction tasks in Sanskrit. In this work, we consider the tasks, word segmentation, morphological parsing, dependency parsing, syntactic linearisation and poetry linearisation. Here, dependency parsing and syntactic linearisation are two syntax level tasks with word segmentation and morphologically parsing being their preliminary tasks. We also introduce the task of poetry linearisation, which is a prosody level task completely independent of the aforementioned morphosyntactic tasks. Here, the morphosyntactic information about a given text is ignored, and instead, its prosody level information is utilised. Figure 5.1 shows the hierarchy of the tasks.

The proposed framework is a search-based structured prediction framework for numerous NLP tasks in a free-word order language like Sanskrit. The models we design using this framework are essentially arc-factored models, similar to graph-based parsing models [MPRH05, Ish11]. Here, the system expects a graph as input with its edges featurised, irrespective of the task. We design suitable inference procedures to incorporate task-specific constraints, by which the search space for the possible solutions is considerably reduced. The task is then framed as the search for a task-specific structure. In principle, the graph based dependency parsing approaches such as McDonald et al. [MPRH05] or the lattice based morphological parsing approaches such as Kudo et al. [KYM04] can all be formalised as specific instances of this framework. To further elaborate, consider the case of dependency parsing. Here, the input graph will be a complete graph, with the (segmented) words in the sentence

forming the nodes of the graph. Here, the specific sub-structure to search for will be a spanning tree [Hir01]. The inference procedure searches for the minimum cost spanning tree, using a suitable algorithm such as Chu-Liu-Edmond [Edm67]. Summarily, training consists of learning an energy function that assigns lower scores to the ground-truth spanning tree than the other candidate spanning trees. All our models follow an arc-factored approach, where the energy of the structure is nothing but the sum of the energies of its edges [Ish11,LCH$^+$06]. The edges being featurised, the energy function is used to score these featurised edge vectors. The performance of a system depends highly on the choice of feature function used for the task. In Morphologically Rich Languages, hand-crafted features still form a crucial component in contributing to the performance of the state of the art systems for tasks such as morphological parsing and dependency parsing [MT16, MSBT19, SÇ15]. But Krishna et al. [KSB$^+$18] learn a feature function using the Path Ranking Algorithm (PRA) [LC10] for the joint task of word segmentation and morphological parsing.[1] PRA essentially maps the problem of learning a feature function to that of automatic learning of horn clauses [GTM15], where each clause is a morphological constraint. The domain knowledge required here confines to just defining the literals, the combinations of which will be used to form the clauses. In Krishna et al. [KSB$^+$18], morphological tags and grammatical categories form the literals, and the feature (clause) values are calculated using distributional information from a morphologically tagged corpus. We observe that the same feature function can be used effectively for all the standalone and joint tasks we experimented with, including both the downstream syntax-level tasks. In the case of poetry linearisation, prosody level information, instead of the morphological information, is used to define the literals. We further improve our feature function learning approach using Forward Stagewise Path Generation (FSPG) [MCM$^+$15]. In all the tasks we experimented with, FSPG based features consistently and significantly outperform our PRA based features and achieve state of the art results.
The contributions of our works are as follows:

1. We propose a general graph-based parsing framework for multiple structured prediction tasks in Sanskrit.[2] We achieve state of the art results in all the tasks we experimented with. In fact, this is the first work that introduces statistical models

[1] This chapter is a substantial extension of Krishna et al. [KSB$^+$18]. We include the relevant findings from the work in the chapter.

[2] The code and data for word segmentation and morphological parsing tasks as presented in Krishna et al. [KSB$^+$18] can be found here https://zenodo.org/record/1035413#.XTcrougzaM9

for performing dependency parsing and poetry word ordering in Sanskrit.

2. We automate the process of learning a common feature function to be used across all the syntax level tasks by using the FSPG approach. This avoids the need for feature engineering for each task separately and is completely automated. Further, this automated procedure simplifies feature extraction process, such that it can be directly used in domains outside of syntax level tasks. For instance, our poetry word-ordering task uses prosody level information instead of morphosyntactic information. Similar to Kiperwasser and Goldberg [KG16], this simplifies the process of choosing the feature function and is not constrained by the accessibility to domain expertise.

3. Sanskrit being a low-resource language, task-specific labelled data is particularly hard to come by. All our models use as low as 10 % of the training data as required by the current neural state of the art models for various tasks. We used around 9.26 % (10,000) and 1.5 % (8,200) of training data as against 108,000 and 0.5 million training sentences for state of the art neural models in syntactic linearisation [KSS+19] and word segmentation [HN18] for Sanskrit.

4. The models in our framework, in principle, are language agnostic. At the same time, the framework enable to incorporate language specific constraints which enables to prune the search space of solutions and to filter candidates during inference. We show that incorporating language specific knowledge improves the performance for such tasks. The LAS (UAS) for dependency parsing improved from 79.28 (82.65) to 83.93 (85.32), and the BLEU score for syntactic linearisation jumped by about 8 BLEU scores.

5.1 Task Descriptions

The tasks we consider in this framework are word segmentation, morphological parsing, dependency parsing, syntactic linearisation (word-ordering) and poetry linearisation. Figure 5.1 shows the hierarchy between the aforementioned tasks. First, we make a distinction between two types of tasks, namely syntax level tasks and prosody level tasks. Poetry linearisation, a prosody level task, is the task of ordering a bag of words into its corresponding verse order. Here, we use the syllable level information along with its prosodic properties

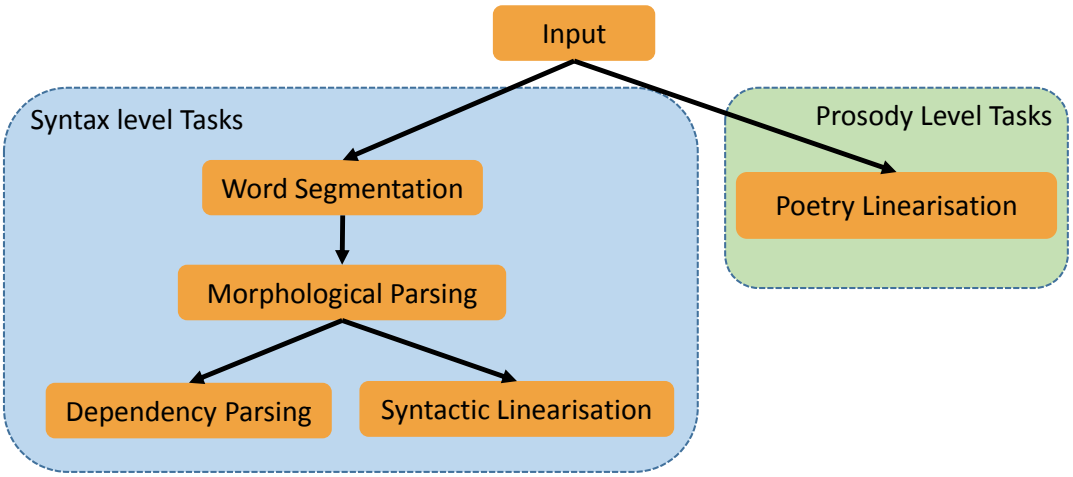


Figure 5.1 Hierarchy of the tasks.

of the input for modelling the task. The rest of the tasks clearly forms a hierarchy where the tasks expect the segmented word forms and their corresponding morphological analysis as part of the input. We first discuss each task as a standalone task, where we assume access to gold standard data to satisfy all of the input requirements for a task, including information from its upstream tasks. But often, this need not be true in the case of an end-to-end parsing scenario [MSBT19]. For instance, in the dependency parsing task, we might expect a segmented sequence as input which requires the word segmentation and the morphological analysis for the segmented word-forms. In such cases, we rely on predictions from the upstream tasks to fulfil the input requirements to the downstream tasks. The joint modelling of related tasks generally results in better performance than a pipeline based approach [SÇ15, MSBT19]. In this work, we jointly model the tasks of morphological parsing (MP) and word segmentation (WS), WS, MP and dependency parsing as well as WS, MP and syntactic linearisation. Summarily we model each of these tasks as a standalone task, with access to gold standard data and additionally, we report experiments with pipeline and joint modelling of the tasks in the hierarchy as shown in Figure 5.1. For our discussion throughout the section, we will be considering the verse, "*Śriyaḥ patiḥ śrīmati śāsituṁ jagajjagannivāso vasudevasadmani vasandadarśāvatarantamambarāddhiraṇyagarbhāṅgabhuvam munim hariḥ*"[3], from the literary work '*Śiśupālavadha*' (शिशुपालवधम्) for illustrative purposes.

[3]श्रियः पतिः श्रीमति शासितुं जगज्जगन्निवासो वसुदेवसद्मनि वसन्ददर्शावतरन्तमम्बराद्धिरण्यगर्भाङ्गभुवं मुनिं हरिः. The sentence translates to, Lakṣmi's consort,Viṣṇu, who is the source of the world, who was born in the house of Vasudeva to control the world, saw Brahma's son Nārada, descending from the sky.

Verse: Śriyaḥ patiḥ śrīmati śāsituṁ jagajjagannivāso vasudevasadmani
Segmentation: Śriyaḥ patiḥ śrīmati śāsituṁ jagat jagannivāsaḥ vasudevasadmani

vasandadarśāvatarantamambarāddhiraṇyagarbhāṅgabhuvam munim hariḥ
vasan dadarśa avatarantam ambarāt hiraṇyagarbhāṅgabhuvam munim hariḥ

Figure 5.2 A verse from 'Śiśupālavadha', with its corresponding segmentation. The instances of Sandhi are highlighted and are shown in boldface.

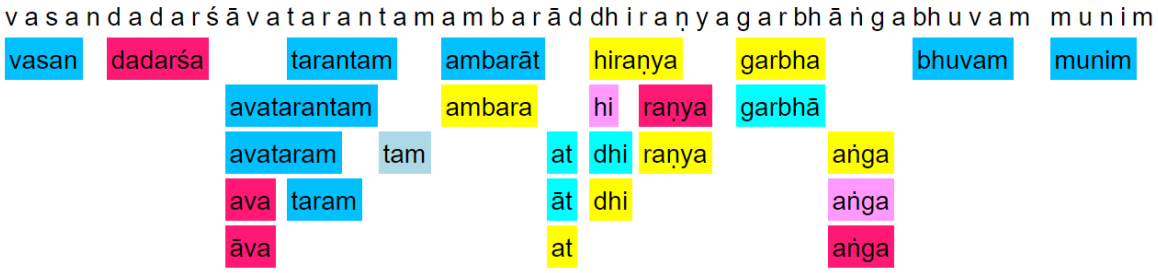


Figure 5.3 All the phonetically valid segmentation solutions from Sanskrit Heritage Reader for the string, '*vasandadarśāvatarantamambarāddhiranyagarbhāṅgabhuvam munim*'.

Word Segmentation: Word segmentation is the task of identifying the words in a given character sequence. Sandhi is defined as the euphonic assimilation of phones at the word boundaries. Figure 5.2 shows six instances of application of Sandhi in the verse from '*Śiśupālavadha*' along with its corresponding segmented sequence.[4] Here, four of the six instances of Sandhi result in phonetic transformations at the word boundaries, while the remaining two cases result in the concatenation of words without any phonetic transformations. The analysis of such a sequence can lead to ambiguity in identifying the original words in the sequence. Goyal and Huet [GH16] propose 'Sanskrit Heritage Reader (SHR)', a lexicon driven shallow parser which encodes all the rules of Sandhi as per traditional Sanskrit grammar.[5] SHR can enumerate all possible lexically valid segmentations for a given sequence. Figure 5.3 shows the possible analyses for a string (a substring of the original sentence from 'Śiśupālavadha'), 'vasandadarśāvatarantamambarāddhiraṇyagarbhāṅgabhuvam munim', as per SHR.[6] Here, we define a segmented solution that spans the entire input sequence as an 'exhaustive segmentation'. For the sentence in consideration, the correct solution

[4]For a complete list of transductions due to Sandhi, visit http://sanskrit.uohyd.ac.in/scl/table.html
[5]https://sanskrit.inria.fr/DICO/reader.fr.html
[6]The analysis for the entire sentence is available at http://bit.do/sandhisplit

is one among 56,160 possible exhaustive segmentations.[6] Given the possible word-splits, our task can be formalised as one of finding the semantically most valid exhaustive segmentation, among the candidate solutions. Here, along with word segmentation, we jointly perform the task of compound splitting, similar to previous word segmentation models in Sanskrit [HN18, RKS+18, KSS+16a].

Morphological Parsing: Morphological parsing is the task of identifying the morphemes of the words in a sentence. Sanskrit, similar to Czech [SST05], is a fusional language where a morpheme encodes multiple grammatical categories. Morphological parsing in Sanskrit is challenging, primarily due to two factors. Sanskrit has a rich tagset of about 1,635 possible tags. Table 5.2 shows the morphological classes in Sanskrit and the grammatical categories they comprise of. Additionally, the inflected surface-forms lead to ambiguity due to syncretism and homonymy. For instance, Table 5.1 shows the case of syncretism and homonymy for the inflected surface forms 'munim' (मुनिं) and 'bhuvam' (भुवं). Specifically, the word 'bhuvam' has three possible analyses, of which two are cases of syncretism for the inflections of the stem 'bhuva' (भुव) and the third analysis is an inflection of the stem 'bhū' (भू). This is a case of homonymy. Sanskrit Heritage Reader provides an exhaustive list of possible morphological analyses for the words in a sequence. We formulate our task as the disambiguation of the correct morphological analysis from the exhaustive list of solutions obtained from SHR.

Dependency Parsing: Given a sentence in Sanskrit, dependency parsing requires to find the syntactic relations between the words in the sentence, thereby predicting a labelled dependency tree as the final output. For the task, we follow the widely adopted dependency tagging scheme proposed for Sanskrit [KPS10, KR13]. The tagging scheme, consisting of 22 relations[7], is, in principle motivated from the traditional dependency analysis for Sanskrit, known as the '*kāraka*' (कारक) theory [Hel09a]. These relations are known to be syntactic-semantic in nature [BS93]. Figure 5.4 shows the dependency analysis for the sample sentence in 'Śiśupālavadha'. In the figure, the original *kāraka* tags are shown as edge labels, inside a shaded box, along with their corresponding English translation.[8] Using this scheme enables us to integrate our predictions into the pipeline of systems currently in use for linguistic

[7] http://sanskrit.uohyd.ac.in/scl/GOLD_DATA/Tagging_Guidelines/tag_proposal_consortium_28Oct2014.pdf

[8] For a more detailed understanding of the tagging scheme and *kāraka* theory in general, please refer to Bharati et al. [BKS19] or Kulkarni et al. [KPS10].

Word	Stem	Morpholgical Tag Case — Number — gender
munim	muni	acc. — 1 — feminine
		acc. — 1 — masculine
bhuvam	bhuva	acc. — 1 — masculine
		acc. — 1 — neuter
		nom. — 1 — neuter
	bhū	acc. — 1 — feminine

Table 5.1 Instances of homonyms and syncretism in the verse from 'Śiśupālavadha'. *munim* is a case of syncretism, where the lemma has the same inflection for different morphological classes. *bhuvam* is a case of both syncretism and homonymy, as it can be an inflection of two different stems.

Feature	**Values**	**Noun**	**Fin. verb**	**Participle**
Tense	18		✓	✓
Case	8	✓		✓
Number	3	✓	✓	✓
Gender	3	✓		✓
Person	3		✓	
Others	6			
Total	41	72	162	1296

Table 5.2 Grammatical features and their distribution over inflectional classes. 'Others' include forms such as infinitives, absolutives, Compound-component, Indeclinables, etc.

annotations and processing of Sanskrit texts [GHK+12, HK14, GH16, Das16]. The relations rely heavily on the case markers of the nouns and the valency of the verb [Pei97, Prz18] to infer the structural information of the sentence [Ram09]. The sentences in prose in Sanskrit are known to follow weak non-projectivity in their dependency analyses, but the same is not guaranteed for the word arrangements in verse order [KSSS15]. Nevertheless, the dependency tree is not dependent on the configurational information of the words in a sequence.

Syntactic Linearisation: In pedagogy, conversion of a verse into its corresponding prose is often performed by an expert [SKS+16], which facilitates easier verbal cognition of the sentence for others [Bha90]. Given that the arrangement of words in the verse order is influenced by the metrical constraints, it has little to offer in rearranging the words to a prose order [SASG15, KSSS15]. Hence we formulate the task as a syntactic linearisation task, where we consider the input as a bag of words and ignore any ordering information provided with the input. Syntactic linearisation is the task of ordering a bag of words into a grammatical and fluent sentence [LZCQ15]. When performed as a standalone task, it assumes access to a bag of words along with morphological analysis for each of the

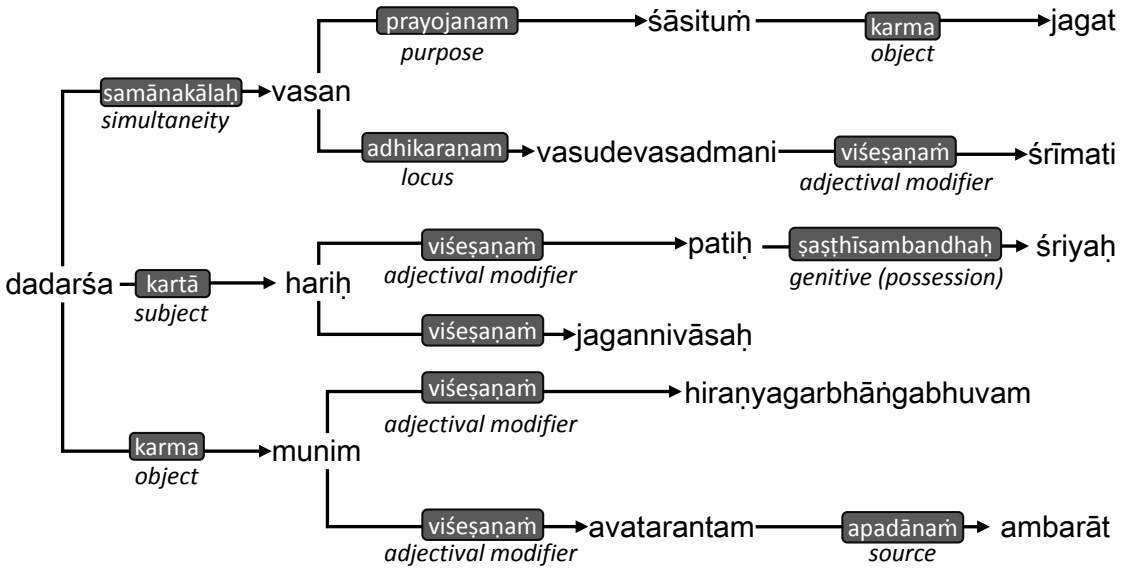


Figure 5.4 Dependency analysis for the verse in 'Śiśupālavadha'. The *kāraka* tags as per the dependency tagging scheme of Kulkarni et al. [KPS10] are shown as the edge labels (in shaded box). The corresponding English translation for the tags are given beneath each of these tags.

words as its input. Figure 5.5 shows an instance of syntactic linearisation from a bag of words. But a more realistic scenario is when the input is a sequence in its verse order with the words joined together due to Sandhi. Here, the task involves word segmentation and morphological parsing followed by linearisation, as shown in Figure 5.1.

A close inspection of the ordering of the sentence in its prose form and verse form reveals the differences between the two. Sanskrit sentences tend to follow SOV typology [Hoc15]. Here '*hariḥ*' (हरिः) is the subject (*kartā*), '*muniṃ*' (मुनिं) is the object (*karma*) and '*dadarśa*' (ददर्श) is the verb. While the prose ordering conforms to the SOV typology, the verse ordering does not. Dependency analysis of the prose constructions in Sanskrit tends to follow weak non-projectivity [KSSS15], which implies that the word arrangement follows Dependency locality [Gib98]. Here, the linear distance between words linked in dependencies should be as short as possible [GFP$^+$19], and uninterrupted if possible. Here the words '*patiḥ*' (पतिः) and '*jagannivāsaḥ*' (जगन्निवासः) are adjectives (denoted by relation *viśeṣaṇaṁ*) to the word '*hariḥ*'. Hence these words will be directly linked to *hariḥ* (Figure 5.4), and hence ideally should not be interrupted by other words in between. While the arrangement in prose order conforms to this, the verse order construction violates the principle of dependency locality.

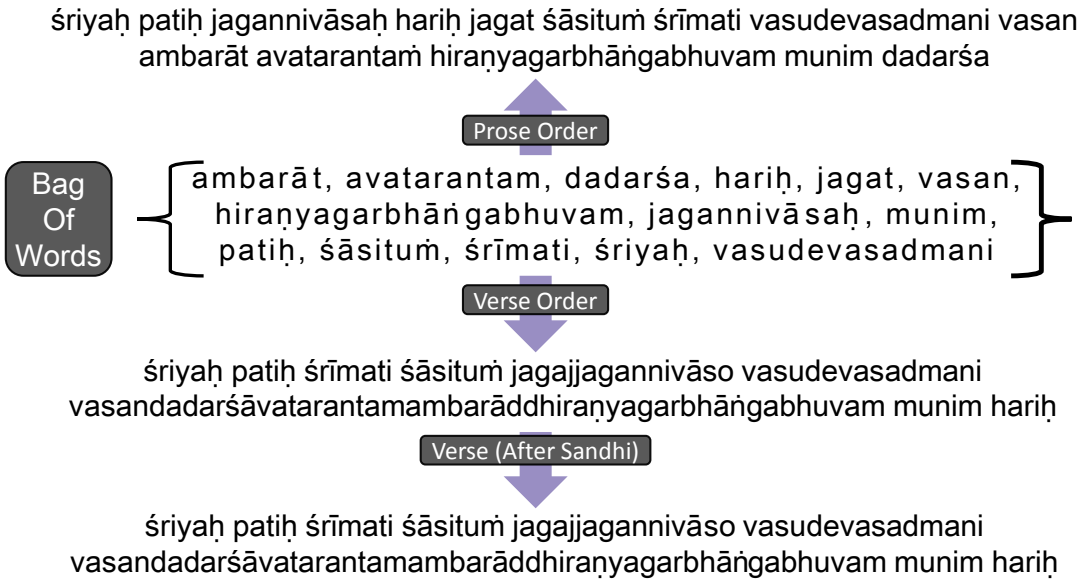


Figure 5.5 Instance of linearisation and poetry word-ordering from a bag of words. In case of poetry word ordering, the Sandhi is performed to adhere to metre constraints.

Poetry Linearisation: The task, similar to syntactic linearisation, is ordering a bag of words into the verse ordering. This essentially is a prosody level task, where the ideal ordering should adhere to one of the prescribed patterns of metre in Sanskrit prosody. The metre of a verse is determined only on the sequence after the Sandhi operations are performed. Figure 5.5 shows both these sequences, verse ordering from bag of words and the modified sequence after the Sandhi. First, Sandhi operation does not affect a sentence in any way other than at the phonetic level. The phonetic transformations can result in reducing the number of syllables in the sequence or might alter the syllable weight of one or more syllables. Syllable weight is a prosodic property, identified deterministically using rule-based systems [MGS13]. The weight of a syllable can either be light or heavy, and this pattern of syllable weights determines the metre of a sequence. So in order to obtain a verse to adhere to a particular metre, a poet would take liberties in both word arrangements and in performing Sandhi at appropriate boundaries. So in poetry linearisation, our task is not just confined to finding the correct permutation from a bag of words, but also to determine the junctures at which Sandhi needs to be performed.

5.2 Energy Based Framework for Structured Prediction in Sanskrit

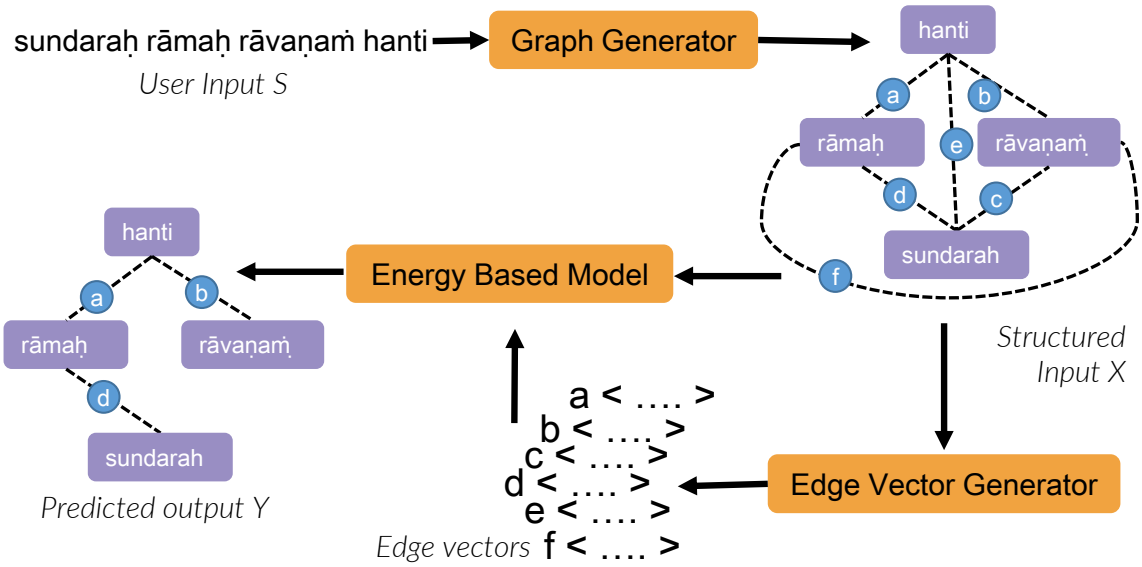


Figure 5.6 Overview of the Energy Based Model architecture. Dependency parsing for a four word sentence, sundaraḥ rāmaḥ rāvaṇam hanti (Handsome Rāma kills Rāvaṇa), is shown as the use-case.

We define a search-based structured prediction framework using Energy Based Models (EBM) for performing numerous sequence level NLP tasks in a free-word order language like Sanskrit. Simply put, training an EBM consists of finding an energy function that outputs a solution Y that best fits the observed variable X, where X is the input to the model [LCRH07]. But here, X and Y can be structured objects which can be decomposed into multiple variables. Such problems are challenging because the number of candidates in the output-space is exponential in the number of output variables that comprise the decomposed structure of Y [DFT14, Bel17]. Energy Based Models enable the design of architectures that can incorporate known properties about the language, or the properties beneficial for the task and then perform constrained optimisation over Y [BYM17]. Our framework at its core is an instance of non-probabilistic learning approaches [LCH$^+$06, Section 6], where it uses 'margin losses' to create an energy gap between the correct structured object and the incorrect structured objects in the search space. A structured prediction approach primarily consists of three components, namely, input

representation, learning and inference [Bel17]. While the learning procedure generally remains the same across the tasks in the framework, the input representation and the inference procedures are task-specific. An overview of the architecture of our framework is shown in Figure 5.6. In the figure, dependency parsing for a four word sentence, '*sundaraḥ rāmaḥ rāvaṇaṁ hanti*' (सुन्दरः रामः रावणं हन्ति, Handsome Rāma kills Rāvaṇa), is shown for illustrative purposes. In the figure, similar to other graph based parsing approaches, the four word sentence S is converted to a complete graph X. Here, the words form the nodes and the edges should capture the association between them. Each edge captures the distributional information between the pair of nodes it connects, in the form of an edge vector. The graph structure, complete graph in this case, and the edge vectors form the input to the Energy based model. The EBM searches for a suitable output structure Y with the minimum energy, using the inference procedure. In the case of dependency parsing, Chu-Liu-Edmond's algorithm becomes the inference procedure that searches for the minimum energy spanning tree.[9] Similarly for other tasks, Table 5.3 enumerates the setup for each of the tasks in terms of user given input S, the structured input to the framework X, the predicted structured output Y and the inference applied.

Formally, the user given input sequence S is converted into a structured object X. In our case, the structured object X will be generated using the graph generator module, and it will always be a graph $X(V_X, E_X)$.[10] The module not only generates the structured object, but it also decomposes the object into variables. The vertices in the graph, denoted by V_X, form the variables here. Each vertex will be encoded with relevant linguistic information about the component it represents. For syntax level tasks, the vertex set is a set of words, whereas syllables form the vertices for the prosody-level tasks. Similarly, the edges in the edge set E_X, capture the dependencies between the variables. The edge set E_X in the graph is then passed to the "edge vector generator" module. This module generates a feature representation $\vec{e}$ for every edge $e \in E_X$. The graph and the edge vectors form the input to the Energy Based Model. Using a suitable inference procedure, the model should output the

[9] We search for the minimum energy spanning tree, instead of the maximum spanning tree as in the case of McDonald et al. [MPRH05].

[10] We overload the notations to indicate both the structured object of the model as well as its corresponding graph representation.

answer Y^*, from a set of possible solutions $\mathcal{Y}$, which has the minimum energy [LCH+06].

$$Y^* = \underset{Y \in \mathcal{Y}}{\operatorname{argmin}} \mathcal{E}(Y, X)$$

Every element in $\mathcal{Y}$ will be a structured object consisting of a subset of variables in X. In graph-theoretic terms, every element in $\mathcal{Y}$ will be an induced subgraph of X. We design suitable inference procedures to incorporate task-specific constraints, by which the search space for the possible solutions is considerably reduced. For instance, in word segmentation, our solution space $\mathcal{Y}$ is the set of all maximal cliques of X [KSB+18]. The inference procedure searches for a $Y \in \mathcal{Y}$ which has the minimum energy. Here, we use the greedy maximal clique selection approach (Algorithm 1) that searches for a maximal clique $Y(V_Y, E_Y) \in \mathcal{Y}$ with the minimum energy. The graph $Y(V_Y, E_Y)$ will be an induced graph of X and hence has its vertex set $V_Y \subseteq V_X$ and the edge set $E_Y \subseteq E_X$.
The training involves learning an energy function that minimises the energy of ground-truth spanning tree Y^{GT} as compared to other candidate structures in $\mathcal{Y}$. Similar to arc-factored approaches in graphs [MPRH05, Car07], the energy of the structure produced by the inference procedure is decomposed as the summation of energies of its edges [Ish11]. The edges are vectorised and are obtained from the 'edge vector generator' module.

$$\mathcal{E}(Y) = \sum_{e \in E_Y} \mathcal{E}(\vec{e})$$

Here $\vec{e}$ is a non-negative real valued vector, for each edge $e \in E_X$. The energy function $\mathcal{E}(.) : [0, \infty)^{|\vec{e}|} \rightarrow (-\infty, \infty)$, takes non-negative real valued edge-vector and produces a scalar energy value. Our EBM model is trained with Multilayer perceptrons, with leaky ReLU activation function at the hidden layer. We apply Hinge Loss [TGK03, AJH03], a generalised margin loss, as the loss function. The loss function takes the following general form:

$$L = max(0, \mathcal{E}(Y^{GT}) - (\mathcal{E}(\bar{Y}) - m)),$$

where m is the positive margin. The difference between the energies of the ground-truth solution ($\mathcal{E}(Y^{GT})$) and that of the incorrect candidate with the lowest predicted energy ($\mathcal{E}(\bar{Y})$) is penalised when larger than $-m$ [LCRH07]. We essentially penalise those incorrect solutions that have low energy values, very close to the energy of the correct solution. Specifically, the energy of incorrect ones should be higher than Y^{GT} at least by a positive margin of m. Essentially, the margin is defined as a function that captures the structural divergences between

the ground truth and other candidate structures, such that larger the divergence of a candidate, the farther its energy should be. In the case of dependency parsing, the margin m is the number of nodes with an incorrect head attached to them [MPRH05, Car07, Boh10]. Similarly, in the case of word segmentation and also for the joint tasks, say the joint modelling of word segmentation and morphological parsing [KSB$^+$18], we predict a structure which contains a subset of nodes from the vertex set V_X of the input X. Here, the margin is basically the square of the number of nodes in the prediction which are not in the ground truth, i.e.

$$m = |V_{Y_T^-} - V_{Y^{GT}}|^2$$

We minimise the given loss function using gradient descent. The network parameters are updated per sentence using back-propagation. The hinge loss function is not differentiable at the origin. Hence, we use the subgradient method to update the network parameters [SMN10, RBZ07].

5.2.1 Graph Generator

The graph generator analyses the user input S, and transforms the input into a graph $X(V_X, E_X)$. The vertices represent the variables, and the edges represent the dependencies between these variables. An edge is added between every two vertices (variables) that can co-exist as variables in a predicted output structure. Hence the set E_X encodes the structural information between the vertices at the input side. The edge should capture the dependencies between the nodes it connects. Every edge has a vector representation which captures the distributional information between the pair of vertices it connects. This vector is used to calculate the scalar energy for the edge, which indicates the strength of the association between the node pairs. For syntax level tasks, we have word-level vertices in the vertex set V_X. Here, each vertex represents a word and its morphemes. Similarly, prosody-level tasks have syllable-level vertices, where each vertex encodes the syllable and its syllable-weight. Thus, a vertex v in the set V_X essentially represents some linguistic information encoded as a tuple of multiple attribute-value pairs.[11] Formally put, every (type of) task consists of a predetermined set of attributes, $A_1, A_2 ... A_n$. We define $\mathcal{A}$ as

[11] We additionally add unique identifiers to each of the vertices, in order to avoid confusion due to multiple instances of the same tuple in the input.

Task	User Input S	Structured Input X	predicted Structured Output Y	Vertex Attributes $\mathcal{A}$	Inference	Type
Word segmentation (WS)	A string with joint words	Graph V_X = set of words	Maximal Clique	A_1=Stem A_2=Inflection A_3=Morph. Tag	Greedy maximal clique selection heuristic. Section 5.2.3, Algorithm 1	Syntax level tasks
Morphological Parsing (MP)	A sequence of segmented words				Chu-Liu-Edmonds [CL65, Edm67]	
Dependency Parsing (DP)	A sequence of segmented words, along with their morphological tags	Complete Graph V_X = set of words	Spanning Tree		Beam Search	
Syntactic Linearisation (SL)	A Bag of (segmented) words, along with their morphological tags		Hamiltonian Path			
Joint WS + MP	A string with joint words	Graph V_X = set of words	Maximal Clique		Greedy maximal clique selection heuristic	
Joint WS + MP + DP			Steiner Tree		Approximation algorithm using Prm's algorithm Section 5.2.3 Algorithm 2	
Poetry to Prose Joint WS + MP + SL			Hamiltonian Path		Beam Search	
Prosodic Lineraisation (PL)	A bag of words	Complete Graph V_X = set of syllables		A_1=Syllable A_2=Syllable Weight		Prosody level task

Table 5.3 Task-wise description of user input S, the structured input to the framework X, the predicted structured output Y and the inference applied.

the Cartesian product of these sets, $\mathcal{A} = A_1 \times A_2 ... \times A_n$. Every vertex encodes a tuple $a \in \mathcal{A}$, where $a(i)$ indicates the value for the i^{th} attribute.

Syntax Level Tasks: As shown in Table 5.3, we consider seven different settings for the syntax level tasks, of which three are for the joint modelling of the tasks and four are for modelling the standalone tasks. Here, each node in the graph represents a word and consists of three attributes each, namely, the surface-from of the word, the stem and its morphological tag. For a given input sequence, we use a lexicon driven shallow parser, Sanskrit Heritage Reader (SHR) [GH16], as the graph generator to obtain all the linguistically valid candidate words. It relies on finite state methods to analyse sequences in Sanskrit. SHR enumerates all possible word splits, valid as per the rules of *Sandhi*, for a given sequence. For each such candidate word, SHR also provides the word-level morphological analysis which includes the surface-form of the word, the stem of the word and the morphological class of the word. Each such unique combination of the aforementioned three attributes forms a vertex.[12] Consider the case of joint dependency parsing, morphological parsing and word segmentation. Here, V_X represents every unique candidate word for the given sequence, as shown in Figure 5.3. Thus, the candidate-word 'bhuvam', as shown in Figure 5.3 will be represented by three different nodes in the input graph X. This is to accommodate for the homonymy and the syncretism expressed by the surface-form 'bhuvam' as per Table 5.1. We convert the collection of candidate words into an input graph.

The edges should connect those nodes that can co-occur in a solution. If two candidate words are proposed as alternatives, such as the words '*tarantam*' and '*avatarantam*' in Figure 5.3, then they are defined as '*conflicting nodes*'. In Figure 5.3, we can find that all the conflicting nodes do have overlapping spans in their word position with respect to the input sequence. An exception to this will be those overlaps valid under the sandhi rules, such as the pair of words '*garbhā*' and '*aṅga*' overlapping at the input character ā in the figure. The edge set E_X consists of edges between every pair of nodes which are not conflicting. This results in a dense graph. Now, for the joint task of dependency parsing along with morphological parsing and segmentation, we need to search for a tree that spans over a subset of nodes, called as Steiner tree. The subset of nodes so obtained using the inference should span over the entire input sequence, and is called as an '*exhaustive solution*' [KSB$^+$18].

[12] We additionally consider the span of the word-form w.r.t the input sequence (as a unique identifier) to distinguish between words that are used more than once in the sentence.

In the case of the standalone tasks of dependency parsing and syntactic linearisation, we expect that the user provided input S will contain the relevant information about the word-inflection, stem and the morphological tag. We form a complete graph in both the cases. For morphological parsing, we expect a sequence of segmented words as the input S. The words are passed to Sanskrit Heritage Reader to obtain possible stem and tag analyses. For all other tasks, we expect a sequence of characters in Sanskrit which may contain words in the sandhied form. Sanskrit Heritage Reader analyses the given sequence and provides the possible word-splits and other attribute information. The case of joint word segmentation, morphological parsing and syntactic linearisation expects specifically a verse as its input.

Prosody Level Task: In the prosody level tasks, we perform poetry linearisation as the sole task. Here, the input is a bag of words and we need to predict the original verse order of the sentence. The challenge for the task here is two-fold. Similar to syntactic linearisation, we have to identify the sequence of word arrangement. Since the sandhi operations may lead to a change in syllable weights, sandhi needs to be applied only at the relevant junctures, which facilitates obtaining a valid pattern of metre. Here, we tackle the word arrangement and decision to perform sandhi jointly. For the task, the user given input S, i.e. a bag of words, is converted into a syllable level graph $X(V_X, E_X)$. Here, each node encodes a tuple of two attributes, namely, the syllable and the syllable weight. The nodes are formed by identifying syllables and their weights using the 'Meter identification tool' by Melnad et al. [MGS13] for each word in S. We additionally add syllable nodes that may appear as a result of sandhi between every pair of words. Figure 5.7 shows the case for a pair of words '*jagajjagannivāsaḥ*' and '*vasudevasadmani*'. Here, the syllables in these words are identified and are created as nodes. Additionally, a node with the syllable '*so*' is created, which can appear in the verse, by virtue of sandhi between these two words ('*jagajjagannivāso vasudevasadmani*). Our task here is to find a valid permutation of the words to form a verse,[13] and hence the relative ordering of syllables in a word cannot be permuted. This gets reflected in our graph structure formation, which we discuss now.
First, every word is represented as a directed path, consisting of its syllables nodes. For a word, the path begins with the first syllable of the word and terminates at the last syllable of the word. Now, given that any two words can co-occur in the sequence, we form directed edges from the last syllable of each word to the first syllable of every other word. We also

[13] With an additional constraint to apply sandhi wherever relevant

add connections to handle syllable nodes which were added due to sandhi (Directed path from *vā* → *so* → *va*). Figure 5.7(Top) shows the graph formation discussed here. Now, this will lead to many nodes, especially the internal nodes of a word, where they have exactly one incoming and outgoing edges each. In such cases, there is no ambiguity, and hence the inference has nothing to search for in the search space. We merge such nodes together and represent those as a single node, as shown in Figure 5.7(Bottom). Hence, effectively a word is converted into three different nodes, the first syllable, the last syllable and the sequence of remaining syllables of a word. Additionally, syllables are added which may exist in a verse by virtue of sandhi.

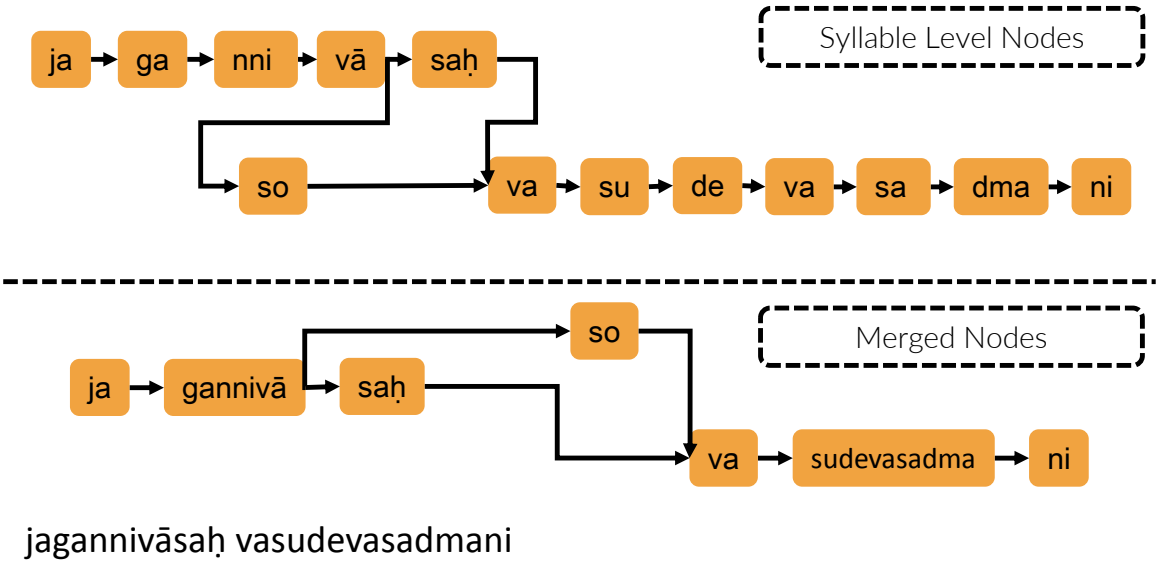


Figure 5.7 Input representation for the prosody level task. Here V_X is a set of syllable level nodes. Nodes where there is no ambiguity in traversal are merged to form a single node in the merged node representation.

5.2.2 Edge Vector Generator

The edges in the edge set E_X are passed onto the edge vector generator for obtaining a feature vector for each of them. Similar to McDonald et al. [MPRH05], we use these feature vectors to score the edges. The arc-factored approach we follow [SÇ15, KRC+10] uses the sum of these edge scores (energies) to predict the score (energy) for the predicted structures. Identifying a set of features that is beneficial for each task is a challenge in itself. For morphologically rich languages, the state of the art models for dependency parsing [MSBT19, SÇ15],

and even for morphological parsing [MT16], rely on hand-crafted features for obtaining the feature-function. In our case, we automate the learning of the feature function. Here, we map the problem of learning a feature function to that of automatic learning of arbitrary length horn clauses [GTM15]. For the syntax level tasks, we define morphological tags and grammatical categories as the literals from which the clauses are constructed, whereas syllables and syllable weights form the literals for the prosody level tasks. While the literals form a finite set, the clauses can be of arbitrary length, which makes the feature space an infinite set. From an infinite space of possible horn clauses, i.e. features, we need to find a finite subset of features which are relevant for the tasks we consider. Krishna et al. [KSB$^+$18] previously used Path Ranking Algorithm [LC10] for the automated learning of feature function. We instead utilise the Forward Stagewise Path Generation (FSPG) algorithm [MCM$^+$15] for this. For all the syntax level tasks, we learn a common feature function, and we learn a separate feature function for the prosody level task. The features we learn are for the edges in the input graph X. For a given edge, the feature value of a feature should essentially capture the strength of the association between the two nodes the edge connects, i.e. how likely are the two nodes to co-occur in a solution. For this, we make use of the linguistic information encoded in the node pair and calculate the strength of their association based on distributional information obtained from a corpus. Each feature acts as a means to calculate this information under specific linguistic constraints. Once the features are learnt, we use the Path Constrained Random Walks approach (PCRW) [LC10] for calculating the feature values.

The Feature Space: A feature in the feature-vector can be defined as a triple $\langle a(i), \psi, a'(j)\rangle$. Given an edge in E_X, let the corresponding attribute tuples for the pair of nodes it connects be denoted as $a, a' \in \mathcal{A}$. Now, $a(i), a'(j)$ are the i^{th} and j^{th} indexed attributes of a and a' respectively. The triple $\langle a(i), \psi, a'(j)\rangle$, should ideally capture the distributional information between the attributes $a(i)$ and $a'(j)$, under the presence of the constraint ψ. Every feature differs from others in two ways. One, each feature has got only partial information of the node. A feature, $\langle a(i), \psi, a'(j)\rangle$, that connects two nodes uses only one attribute each from these node pairs. For the syntax level tasks, the attribute tuple essentially is a collection of 3 attributes, namely, the surface-form, the stem of the word, and its morphological class. Second, it additionally uses a constraint ψ to limit the distributional context under which the association strength between $a(i)$ and $a'(j)$ should be measured. Here ψ is designed to capture the interaction between the individual grammatical categories

of the words in a sentence that is indicative of the structural information of a sentence.[14] For instance, the grammatical category for a noun, 'case' is indicative of its syntactic role in a sentence, i.e. dependency tag, while the 'gender' or 'number' of a noun helps in maintaining the agreement between other words. ψ essentially needs to capture the combination of such grammatical categories that is beneficial for the task. ψ can be seen as an arbitrary length tuple $(\psi(1), \psi(2), ...\psi(k))$, where k can be any non-negative integer. This creates a feature space with infinitely many possible features. Here, each entry in the tuple ψ brings in some morphological context. For a syntax level task (prosody level task), between two nodes in X it is possible to enumerate nine (four) different kind of features even without any additional constraints. We refer to this type of feature as ϵ-features, $\langle a, \epsilon, a' \rangle$. But, the total number of possible features can be infinite, as the morphological constraint ψ can be an arbitrary length tuple. Formally, a feature, $\langle a(i), \psi, a'(j) \rangle$, is a tuple $(a(i), \psi(1), \psi(2), ...\psi(k), a'(j))$.[15] Here, every feature starts and ends with an attribute from attribute tuple a. For syntax level tasks, this is, surface-form, stem and morphological tag and in the case of prosody level tasks, this is syllable and syllable weight. Every intermediate entry $\psi(1), \psi(2), ...\psi(k)$ is an entry in the constraint tuple ψ. In the case of syntax level tasks, we use morphological constraints and in the case of prosody level tasks, we use sub-patterns of a metre (sequences of syllable weights) as the constraints. We now define ψ as member of a countably infinite set $\mathcal{MC}$ where:

$$\mathcal{MC} = \bigcup_{i=1}^{\infty} M^i$$

Here, $\psi \in \mathcal{MC}$, is a tuple where each element in the tuple comes from a set M. M^k indicates the k-ary Cartesian product on the set M. $\mathcal{MC}$ is the union all such k-ary Cartesian products, where k can be any non negative integer. This enables $\mathcal{MC}$ to be a countably infinite set of tuples which contains tuples of any arbitrary length. In prosody-level task, M is nothing but a set consisting of all the syllable weights, which happens to be a set of cardinality 2 for Sanskrit. Here, rather than the size of the M, it is the length of tuples in $\mathcal{MC}$, i.e. the sequence of syllable weights, that matters. But in syntax level tasks, M is

[14] For the purpose of illustration we explain the scheme in terms of syntactic tasks. But these are applicable for prosodic tasks with a variation that the feature space consists of sub-sequences of meters patterns in Sanskrit Prosody.

[15] As a horn clause formulation, this can be denoted as $(a(i) \wedge \psi(1) \wedge \psi(2)..... \wedge \psi(k)) \rightarrow a'(j)$, where each entry in the tuple is a literal. We stick with the tuple representation, as the relative ordering of the literals is necessary for the feature value calculation.

a finite set formed by the union of the set of grammatical categories, morphological tags and any combination of categories that forms a part of a morphological tag. As previously mentioned, this definition of M is influenced by the role of specific grammatical categories, or that of a combination of grammatical categories, in syntactic disambiguation. The set M is defined such that an element in it is either a complete combination of categories, which leads to a valid morphological tag, or a partially-complete combination of grammatical categories. We define the complete combination and partial combination as follows:

1. *Complete combination*, i.e., a morphological class – In Sanskrit, similar to Czech [SST05], a morphological class represents a certain combination of grammatical categories. For instance, a noun is represented by case, gender and number. Hence, the combination 'genitive-masculine-singular' forms a morphological class.

2. *Partial combination* - A combination of grammatical categories, which can form a morphological class by adding one or more categories to it. For instance, 'genitive-masculine' is a partial combination that denotes all the possible (three) morphological classes which differ from each other only in terms of the category 'number'. However, 'genitive-present tense' is not a 'valid' combination as it can never form a valid morphological class. The evidence for a partial combination in the corpus $\mathcal{C}$ can be obtained by summing the evidence of all the morphological classes which it denotes.

We obtain a total of 528 different entries in set M, where we consider 240 complete combinations and a possible set of 288 different partial grammatical category combinations. Here, we consider all complete and partial combinations of a noun, verb and those denoted by 'Others' in Table 5.1.

Feature Score: For every edge in X, the feature function we learnt should produce a real valued feature vector. For calculating the feature values, we use distributional information from a corpus. Here, for the syntax level tasks, we assume we have access to a corpus, where the sentences in the corpus have undergone both word segmentation and morphological parsing. Using this corpus, we construct a graph, $\mathcal{C}$, henceforth to be referred to as the corpus graph. Here for every unique word in the corpus, we create three nodes, one each for the attributes in the attribute tuple (for the syntax level tasks)[16], i.e. surface-form,

[16]For the prosody level task, we assume the corpus to be segmented at a syllable level, and every syllable is tagged with its syllable weight. Also, the attribute tuple used here will be a tuple of two entries, i.e. syllable

the lemma of the word, and its morphological class. $\mathcal{C}$ is a typed graph where the type of a node can be one of the three attributes or any entry in the set M, i.e. a partial or complete combination of grammatical categories. For every type in M, we create one node each in $\mathcal{C}$.[17] An edge exists between every pair of nodes, irrespective of its type, if both nodes co-occur in a sentence in the corpus. Further for every node in $\mathcal{C}$, we calculate its frequency of occurrence in the corpus. Such a formulation for $\mathcal{C}$ enables us to use the Path Constrained Random Walks (PCRW) approach [LC10] for the calculation of feature values.

For a given edge in the sentence graph X, a feature can be denoted as a tuple $(a(i), \psi(1), \psi(2), ...\psi(k), a'(j))$. As per PCRW, this tuple forms a path in $\mathcal{C}$. To elaborate, every feature considers only an attribute each from the nodes involved and calculates the distributional information between them under a constrained context denoted by ψ. Here, the attributes $a(i)$ and $a'(j)$, each forms a node in $\mathcal{C}$. Further, since every entry in $\psi(1), \psi(2), ...\psi(k)$ is a member of M, each of it is present as a node in $\mathcal{C}$. Using the path so formed in $\mathcal{C}$, we calculate the feature score. Now, consider an ϵ-feature, $\langle a(i), \epsilon, a'(j)\rangle$. Here, since there exists no constraint ψ, the feature forms an edge between the attributes $a(i)$ and $a'(j)$ in $\mathcal{C}$. The score for an ϵ-feature, translates to the co-occurrence probability of both the attributes in sentences across the corpus. This is defined as P_{CO}:

$$P_{CO}(a(i)|a'(j)) = \frac{count(a(i), a'(j))}{count(a'(j))}$$

Now, the feature score calculation can be extended in the case of the presence of morphological constraints in $\psi \in \mathcal{MC}$. For $\langle a(i), \psi, a'(j)\rangle$, where $\psi = (\psi_1, \psi_2, ...\psi_k)$:

$$P_{\psi}(a(i)|a'(j)) = P_{CO}(a(i)|\psi_1) \times \left(\prod_{i=2...k} P_{CO}(\psi_{i-1}|\psi_i) \right) \times P_{CO}(\psi_k|a'(j))$$

The feature score for a given node-pair is calculated using the Path Constrained Random Walks (PCRW) [LC10], where the features are learnt using Forward Stagewise Path Generation (FSPG) Algorithm [MCM$^+$15].

Learning of Feature Function: We use the Forward Stagewise Path Generation (FSPG) Algorithm [MCM$^+$15] for the learning of feature function. Given the feature space, which,

and the syllable weight.

[17]The number of nodes of the types lemma, surface-form or morphological class is dependent on the corpus size.

in our case is infinite, FSPG uses greedy strategies to generate the most relevant subset of features [MCM+15]. It is a modified version of Least-Angle Regression approaches (LARS) [EHJ+04], which use greedy strategies for feature subset selection from a potentially infinite feature space. The approach is used as a pretraining step, with an auxiliary supervised objective for the regressor. We first define a regression model, where the ground truth is obtained based on a measure of binary association between a pair of words in a corpus, such as bigram probability, co-occurrence probability or PMI [KSB+18]. The method iteratively chooses the most relevant feature which can distinguish the strong associations from the weak associations. The choice for the most relevant feature is made greedily, where the method chooses a feature with the strongest correlation to the expected output [MCM+15]. For this purpose, a vector of residual values called as the residual vector $\vec{r}$ is constructed. Residual vector essentially is the element-wise difference between the values of the ground truth examples and the predictions from the current model, for the training data given as input [MCM+15]. Now, for a new feature to be added, we create a vector $\vec{m}$, where we calculate the feature score for the training data given as input. The cosine similarity between $\vec{r}$ and $\vec{m}$ for each of such feature is calculated, and the feature with the highest cosine similarity is chosen as the feature to be added. The algorithm terminates until the residual vector $\vec{r}$ is negligible i.e. $|\vec{r}| < \epsilon$. The FSPG framework essentially is an extension of LARS, where the framework is adapted to handle infinite feature spaces [MCM+15]. Instead of exhaustive enumeration of possible features, the authors propose a 'GreedyTree' Algorithm [MCM+15, Algorithm 2] for finding relevant features iteratively. Similar in spirit to the approaches used in Monte Carlo Tree Search, GreedyTree algorithm follows a best first search, which explores a graph by expanding the most promising node in the graph [GS10]. A feature in our framework is defined as a tuple $(a(i), \psi(1), \psi(2), ...\psi(k), a'(j))$. This tuple is represented as a path in the GreedyTree. Here, each element of the tuple is a node. A path is kept on expanding till a certain lower bound for a priority score is obtained. After a feature with the highest priority is expanded, the tree structure is preserved for subsequent feature generation passes [MCM+15]. Previously, Krishna et al. [KSB+18] used Path Ranking Algorithm for automated learning of the feature function. But, PRA performs exhaustive enumeration of the features and then perform feature selection using Path Ranking Algorithm. But in PRA the exhaustive enumeration was achieved in practice by limiting the maximum length of the feature tuple to an arbitrary length. In both PRA and FSPG, the feature score is calculated using PCRW.

Vector Formation in the Merged Representation for Prosody Level Tasks: In Section 5.2.1, we have already discussed graph formation in prosody level tasks. Here nodes are formed at a syllable level, but the internal syllables of a word are merged together to form a single node. This results in a substantial reduction in the number of vertices and edges in the graph. The edge vectors are expected to generate before the merging of nodes is performed, and naturally, this means that the vectors need to be updated to handle the merging operation. For instance, in Figure 5.7(Top), consider a path of 3 edges *ja* $\rightarrow$ *ga* $\rightarrow$ *nni* $\rightarrow$ *vā*. In Figure 5.7(Bottom), this was merged to form a single edge, *ja* $\rightarrow$ *gannivā*, between syllable *ja* and the merged syllable sequence '*gannivā*'. To obtain edge vector for the pair *ja* $\rightarrow$ *gannivā*, we perform element wise max operation of all the 3 edge vectors in the path *ja* $\rightarrow$ *ga* $\rightarrow$ *nni* $\rightarrow$ *vā*, which is similar to the max-pooling operation. It implies that the new vector will contain the maximum value at each dimension of the vector, among the three edge vectors in the path. Formally put, let the edge vectors in the merged path be represented as a matrix L of size $m \times t$, where m is the number of edges in the path and t is the size of each edge vector. Now, we generate a new vector $\vec{l}$ for the merged representation, where every k^{th} element of $\vec{l}$, $l_k = \max_{1 \leq i \leq m} L_{ik}$.

5.2.3 Inference Procedure

The inference procedure predicts a specific structure Y, an induced subgraph of the input graph X. Table 5.3 mentions the different inference procedures our framework uses and the specific structure they predict. Similar to the McDonald et al. [MPRH05], we use Edmonds-Chu-Liu algorithm for obtaining the directed spanning tree. Similarly, we use Beam search as our inference for all configurations of the linearisation task. But the task of poetry to prose (WS+MP+SL) involves taking a verse in its original form, including the joint words, and then finding the corresponding prose order. This involves word segmentation as one of the tasks, implying only a subset of the nodes from the input X will be in the predicted structure. Here, we encode the constraints in the inference such that once a node is selected, the 'conflicting nodes' to it are removed from the search space. We perform the same for other tasks involving segmentation. This can be observed in Step 4 of the Algorithms 1 and 2 as well. 'Conflicting' nodes are any pair of nodes which are not connected by an edge between them. This follows from the construction of the graph X, as the non-connectivity between the nodes implies that they are proposed as alternative candidate suggestions to

each other in X. The removal of such nodes guarantees that the structures predicted by these inference procedures will always result in an 'exhaustive segmentation'.

ALGORITHM 1

Greedy maximal clique selection heuristic

1: **for** each node v_i in V_X **do**
2: Initialise a graph $K_i(V_{K_i}, E_{K_i})$ with $K_i = G$ such that $V_{K_i} = V_X$ and $E_{K_i} = E_X$. Initialise a vertex set V_{T_i} with v_i as the only element in it. Remove all the vertices which are conflicting with v_i from K_i.
3: Add the vertex $v_j \in (V_{K_i} - V_{T_i})$ to V_{T_i}, such that in K_i, the sum of edge weights for the edges starting from v_j to all other vertices in V_{T_i} is minimum.
4: Remove all the vertexes which are conflicting with v_j from V_{K_i}.
5: Repeat steps 3-4 till $V_{K_i} - V_{T_i} = \varnothing$
6: **end for**

ALGORITHM 2

Approximation algorithm for finding directed Steiner Tree using Prim's algorithm originally proposed for Minimum Spanning Tree

1: **for** each node v_i in V_X **do**
2: Initialize a tree $T_i(V_{T_i}, E_{T_i})$ with v_i as the only vertex and remove all the vertexes in V_X which are conflicting with v_i.
3: Add the vertex $v_j \in V_X - V_{T_i}$ to V_{T_i}, where v_j forms the minimum weighted directed edge with any of the vertexes in the V_{T_i}.
4: Remove v_j and all the vertexes conflicting with v_j from V_X.
5: Repeat Steps 2 - 3 till $V = \varnothing$
6: **end for**

Algorithm 2 is used for the joint task of word segmentation, morphological parsing and dependency parsing. The procedure is basically a slightly modified version of Prim's algorithm, used here as an approximation algorithm for finding directed Steiner trees [Tak80]. Algorithm 1 is used for the tasks of word segmentation, morphological parsing and the joint task of word segmentation and morphological parsing [KSB$^+$18]. Here, we start the clique selection with a single node. At any given instance, we loop through the nodes in the graph which are not yet part of the clique. We add a vertex v to the clique if the cumulative score of all the edges from v to every vertex that is already in the clique is the minimum. We discard all the nodes which are conflicting with vertex v. The procedure terminates, when there exist no more vertices to loop through. We then obtain the maximal clique (*exhaustive segmentation*), as guaranteed by our structured input graph construction. We perform this

for every node in the graph X. From all the cliques so obtained, we select the maximal clique with the least score. The approach does not guarantee enumeration of all the cliques, but it is guaranteed that every node will be covered by at least one maximal clique. In fact, the inference procedures for most of the tasks shown are essentially approximation algorithms. These procedures can be seen as a means of sampling some potential minimum energy maximal cliques for the learning task. Energy based models do not require proper normalisation of the solution space [LCH$^+$06], a choice that enables the use of the heuristic. During inference, the greedy clique selection heuristic is performed for every node in X. Though the run-time for this inference is polynomial, it can still be computationally expensive. But, in practice, we find that our inference procedure results in faster output for graphs with > 19 nodes in comparison to the exponential time Bron-Kerbosch algorithm [TTT06, BK73] for clique enumeration [MPK$^+$05]. We further improve the run time of our inference procedure by paralleling the clique selection procedure for each node on a separate thread.

5.2.4 Design Decisions

The decision of ours to design an architecture that is computationally expensive as compared to the standard sequence labelling approaches, generally employed for similar tasks, deems justification. Our motivation for this comes from the linguistic characteristics and the nature of the texts that we generally deal with in Sanskrit. Sanskrit is a morphologically rich language, and hence the grammatical information encoded in a sentence relies more on morphology rather than the configuration of a sentence. This makes Sanskrit, a relatively free word order language. Additionally, in works of poetry, the poets have a constraint that their constructions follow certain prescribed metre or rhythm. Here, the poets exploit the free word-orderness inherent in the language and often permute the words to fit the metrical pattern even at the cost of verbal cognition of the reader [SKS$^+$16, SASG15]. Currently, there exist digitised versions of texts which spans over a period of 3000 years categorised into pre-classical literature (1500 BCE - 100 BCE), classical literature (300 CE - 800 CE) and modern literature (900 CE to now). Hellwig [Hel09a] asserts that the assumption that Sanskrit syntax has remained unchanged over an interval of over 3000 years is not valid. Kulkarni et al. [KSSS15] note that the constructions in prose generally follow weak non-projectivity [Hav07, ML11]. Kulkarni et al. [KSSS15] also observe that constructions in verses violate weak non-projectivity, especially with the adjectival and

genitive relations. A large number of texts are written in verses or to complicate things further, they are written as a combination of prose and verse. A lack of consensus among the experts on a common set of rules for works across the different time spans, and the enormous effort in categorising constructions based on their writing styles motivated us to use this graph construction scheme which is agnostic to word order.

The configurations of the Energy Based Model we have discussed so far can be seen as language agnostic. The language specific components come from the Graph generator and the Edge vector generator modules. The framework treats the input as a graph without incorporating any language specific constraints. But, the architecture enables one to incorporate known properties about the language and then perform constrained optimisation over the structure to be predicted [BYM17]. We experiment with EBM configurations where we incorporate such language specific constraints for the Dependency parsing and syntactic linearisation. The linguistic information is made to use both in pruning the search space and in filtering the candidates during the inference. Essentially, these constraints can be seen as higher order features that capture constraints that span beyond a pair of words. For dependency parsing, we rely on the linguistic information from rule based dependency analysers proposed for Sanskrit [KPS10, KR13, Kul13]. Using these works, we utilise information from morphological markers to explicitly prune grammatically invalid edges in the search space wherever possible, and not rely on the model to disambiguate those. We also incorporate additional knowledge that lies outside of morphological markers, such as roles of specific indeclinable words, as well as cases where an indeclinable acts like a noun by taking a declension and forming noun-noun relations [KPS10, Section 2.2]. We add numerous constraints which mandate or invalidate the presence of certain word pair connections and relations in the final solution, by virtue of the presence of one or more relations in the solution [KR13, Section 4]. Such constraints are actively used in the linguistic tradition of dependency analysis in Sanskrit [KPS10, Section 2.1]

We incorporate additional linguistic information for syntactic linearisation as well [KSSS15, SASG15]. Similar to the dependency parsing task, we make use of the rule based constraints for dependency analysis [KPS10] to obtain a pruned search space. Further, on the pruned search space, we apply the principles of dependency locality [Gib98, GPB$^+$13] and weak non-projectivity [KSSS15, Section 3.2 and Section 4.3] to filter out invalid partial candidate sequences. Dependency locality is a theory which states that the linear distance between words linked in dependencies should be as short as

possible [GFP+19]. This is equivalent to the principle of *sannidhi* (सन्निधि) or proximity as used in principles of verbal cognition in Sanskrit [KSSS15].

5.3 Experiments

We present the results for the experiments for all the tasks that we have described so far. We first look into the performance of standalone tasks and compare them with other state of the art baselines whenever possible and show the effectiveness of our model. Then we further test the hypothesis, that whether formulating the setting of joint formulation of the tasks is beneficial as compared to a pipeline based approach. We further evaluate settings where linguistic characteristics specific to Sanskrit are taken into consideration.

5.3.1 Dataset

We use the Digital Corpus of Sanskrit (DCS) [Hel16a] for all our experiments. Krishna et al. [KSS+16a] identify a subset of 350,000 such constructions from the DCS for their tasks.[18] This dataset was used as the corpus $\mathcal{C}$ for the generation of edge vectors in Krishna et al. [KSB+18]. We re-use most of the same corpora for our edge vector generation. In the dataset, we replace some of these textlines, which form the training or test data for the tasks introduced in this work. Now, we specify the training and the test data for each of the tasks, most of which come from various parts of the DCS. A possible exception to this will be for the task of dependency parsing, which we describe below.

Word Segmentation and Morphological Parsing: Krishna et al. [KSB+18] used a training set of 8,200 textlines from DCS for training the joint model for word segmentation and morphological parsing. In Krishna et al. [KSB+18], the segmentation results were reported on a dataset of 4,200 sentences (termed as DCS4k), and results for morphological parsing were reported on a dataset of 9,576 sentences called as the DCS10k. Here, we will report the results of morphological parsing on the same DCS10k dataset, but word segmentation on a subset of DCS10k with 8925 sentences. The subset was chosen to accommodate the mismatches with various baselines we compare with.

[18] This is the same model we discussed in Section 3.4

Dependency Parsing: DCS is a morphologically tagged corpus but does not contain annotations for dependency tagging in the dataset. We obtain 4,000 dependency tagged sentences from Kulkarni [Kul13] and the corpus of Śiśupālavadha[19]. From the collection, we select about 1,300 sentences as our test dataset. The test data were filtered such that every stem in those sentences, though not necessarily the inflected surface-from, should be present in our corpus $\mathcal{C}$. From the remaining 2,700 sentences, we choose to expand the training set by data augmentation approaches. We relied on standard approaches generally followed in text classification, Semantic Role labelling and in dependency parsing for data augmentation. We perform synonym replacement, of at most one word per sentence [ZZL15], such that the stem for new word is found in DCS. We additionally perform sentence simplifications [VK08], originally proposed for SRL, and sentence cropping [SS18] proposed for dependency parsing. In sentence cropping, unlike Sahin and Steedman [SS18], we crop non subject/object words which, targeting other words that form other dependency relations. While augmentations may lead to sentence-level semantic shifts, it will still preserve local syntactic tags [SS18]. We altogether obtain a set of 20,000 augmented sentences. We further apply linguistic constraints based on the principles of verbal cognition [Bha90] and Kāraka (Sanskrit specific dependency analysis) [KR13] and obtain a total of 12,320 sentences. We use the filtered set of sentences as our training data, and each sentence has an average length of 6.42 words per sentence. This is comparable to the average number of words in a sentence in the test data, which is 7.03. The test data is devoid of any augmentations and in its original form.

Syntactic Linearisation and Poetry Linearisation: From Table 5.3, it can be observed that we essentially perform three linearisation tasks. The joint task of word segmentation, morphological parsing and the syntactic linearisation, requires a pair of sequences, both being permutations of each other. The input is a text construction in the verse order, and the output is a text construction in its corresponding verse order.[20] But for the standalone task of syntactic linearisation (Poetry linearistion), the input is the bag of words in the construction, and the prediction is the sequence in prose (verse) order.

We use the verses and its corresponding prose constructions from the epic Rāmāyaṇa (रामायण), which we obtain from Aiyaṅkār [Aiy10] .[21] We filter 17,017 pair of text

[19] http://sanskrit.uohyd.ac.in/scl/e-readers/shishu/#/home

[20] The input will retain the sandhi information, while the predicted prose order will be in the segmented form.

[21] All the verses and their prose form is available at https://www.sanskritdocuments.org/sites/valmikiramayan/

constructions from the work, both in its verse and prose word order.[22] Krishna et al. [KSS+19], the current state of the art in prose word ordering in Sanskrit[23], use a test set of 3,017 text constructions from this dataset. We maintain the same test data for all the three tasks. From the remaining data, we use the 10,000 text constructions as training data. Since Rāmāyaṇa is already included in DCS, we obtain the gold standard word segmentations and the morphological analysis of the words, for the syntactic level tasks. For poetry linearisation, we identify the syllables and the weight of the syllables using the meter identification tool by Melnad et al. [MGS13].[24]

We compare our EBM configurations with the state of the art neural baselines for the standalone tasks. These baselines are provided with additional data for their training. For syntactic linearisation, we use 95,000 prose order sentences crawled from Wikipedia by Krishna et al. [KSS+19]. Similarly, for poetry linearisation, we use 100,000 verses from DCS and Vedabase.

5.3.2 Baselines

Table 5.4 shows the list of baseline systems and the various configurations of our EBM framework, along with the tasks for which those systems are used for. We provide a brief description of each of the baseline systems.

Word Segmentation

SupervisedPCRW (Sup-PCRW) Krishna et al. [KSS+16a]: Similar to the EBM-framework, the model uses the graph output from SHR. It treats the problem as an iterative query expansion problem, where a greedy heuristic approach was used to select candidate nodes based on the edge weights. The edges are featurised with hand-crafted morphological features. The feature values are calculated using PCRW [LC10].[25]

EdgeGraphCRF (EG-CRF): This is a second order CRF Model [MB14, Ish11] which also uses the SHR output graph X as the input to the system. Every node is represented with fastText [BGJM17] word embeddings. The edges are featurised with the PRA vectors. We

[22] Filtered from 18,250 verses. The remaining were ignored due to problems in aligning the poetry and the corresponding prose at verse level.

[23] We discussed this model in Section 4.3

[24] The process of identifying syllable and syllable weights is a deterministic process.

[25] This is the same model we discussed in Section 3.4

System	Tasks							
	WS	MP	DP	SL	WS+MP	WS+MP+DP	WS+MP+SL	PL
EG-CRF	✓	✓			✓			
Sup-PCRW [KSS+16a]	✓							
rcNN-SS [HN18]	✓							
FCRF [MGN18]		✓						
SeqGen [TS18]		✓						
NeurDP [DM17]			✓					
YAP [MSBT19]			✓					
EeasyFirst [GE10]			✓					
LinLSTM [SRS16]				✓				✓
BSO [WR16]				✓				✓
Transformer [VSP+17]				✓				✓
kāvyaguru [KSS+19]				✓				✓
Energy Based Model (EBM) Configurations								
VL-EBM	✓	✓			✓			
BL-EBM	✓	✓			✓			
Prim-EBM-P	✓	✓			✓	✓		
Prim-EBM-F						✻		
Tree-EBM-P			✓					
Tree-EBM-F			✓					
Tree-EBM*-F			✻					
Cliq-EBM-P	✓	✓			✓			
Cliq-EBM-F	✻	✻			✻			
Beam-EBM-F				✓			✻	✻
Beam-EBM*-F				✻				
ATSP-EBM-F				✓			✓	✓

Table 5.4 Different baseline systems and configurations of EBM used in various tasks. The tick mark (✓) indicates that the system is used for the task as a baseline. Star (✻) indicates that the system reports the best score for the task. The naming for EBM configurations is as follows: The inference procedure used, the term 'EBM' which is followed by the type of the edge vector generation approach marked by its first character (PRA/FSPG). Here 'Tree-EBM*-F' and 'Beam-EBM*-F' are the two EBM configurations with language specific augmentations as described in Section 5.2.4.

used 1-slack structured SVM for training. QPBO [RKLS07] inference approach provided the best results for the model.

Char RNN-CNN Seq-Tagger (rcNN-SS): Hellwig and Nehrdich [HN18] propose a sequence tagger that uses character level recurrences and convolutions to train a Sanskrit word segmenter. The best performing model of theirs passes characters of the sequence through a bidirectional RNN and then convolution operations on the resulting embedding. Additionally, shortcut connections [Bis95] are applied both from the input character embeddings and the RNN output to the resulting embeddings.

Lattice-EBM: This is an energy based sequence labelling model, where the input is a lattice [WW77] similar to that of Kudo [Kud06]. The model can be seen as a special case of Graph Transformer Networks [LBBH98, LCRH07]. In the lattice structure, the candidate links only to its adjacent nodes in an exhaustive segmentation. We also generate edge vectors for the dummy nodes that act as the start and end markers in the lattice. We use the PRA vectors as the edge-vectors for the model. During prediction, we have to find the best path from the lattice, which minimises the sentence score. Here, we consider two variants of Lattice-EBM. *VL-EBM* uses the discriminative forward training approach [CWB$^+$11] with the standard hinge loss. The second variant *BL-EBM*, uses multi-margin loss [EOA$^+$18], instead of the hinge loss. Here, we employ beam search to generate multiple candidates as required by the loss.

Prim-EBM-P: This is also a configuration of the energy based model, where the model uses the input graph X from SHR, PRA vectors for the edges and the modified Prim's algorithm as the inference. The inference procedure searches for a Steiner Tree [Tak80] from the input graph X. Prim's algorithm acts as an approximation algorithm to find the Directed Steiner tree [Vos93]. The Steiner tree essentially spans over a subset of nodes, and by graph construction, it is guaranteed that the inference procedure will produce an exhaustive segmentation.

Clique-EBM (Cliq-EBM-P/Cliq-EBM-F): This is the proposed EBM configuration for the word segmentation (and morphological parsing) problem. Similar to other Prim-EBM-P, the configuration uses the input X obtained from SHR. Here we use our greedy maximal clique selection heuristic, as shown in Algorithm 1, as the inference procedure. In Krishna et al. [KSB$^+$18] the configuration used PRA edge vectors, and is denoted as Cliq-EBM-P. Here, we additionally introduce a variant Cliq-EBM-F which uses FSPG based edge vectors for the task. The model works exactly the same as Prim-EBM-P but differs only in the inference procedure used.

Morphological Parsing

Neural Factorial CRF (FCRF): Malaviya et al. [MGN18] proposed a Factorial CRF model [SMR07] that relies on an LSTM based sequence model for its feature representation. The model predicts a composite label for each word in the sequence, where a label is a set of tags, one for each of the grammatical category. Here, co-temporal factors are defined to capture the dependencies between various tags of the same word. To capture the dependencies across the words, there exist factors between tags of the same type for the adjacent words, thereby forming a linear chain.

Sequence Generation Model (SeqGen): The model, proposed in Tkachenko and Sirts [TS18] also treats the label as a composite label. Here, a char-BiLSTM is used to obtain word-embeddings, which are then passed on to a word-level BiLSTM as the input features [LBS$^+$16, HNvG17]. The model inspired by Seq2Seq models, uses an LSTM decoder to predict the grammatical categories one after the other, based on the previous predictions for each word.

Energy Based and CRF Based Configurations: All the configurations of the Energy based models, in addition to Edge Graph CRF, which is used for the word segmentation task are used in the morphological parsing task as well. These models predict all the morphemes of a given word, including the stem of a word. But the aforementioned neural baselines only predict the morphological label of a word and not the stem. In these models, we use segmented sentences obtained from the gold standard data in DCS, as input to the SHR. SHR does not analyse for sandhi and only performs morphological analysis of the words involved. This forms the input graph X for these models.

Dependency Parsing

Yet Another Parser (YAP): The model is a language agnostic dependency parser for Morphologically rich languages [MSBT19]. The transition based framework currently reports the best score for dependency parsing and for the joint task of morphological analysis and dependency parsing in Hebrew. Further, the framework was experimented on multiple languages for dependency parsing [SMT18].

Easy-First Parser (EasyFirst): The model proposed by Goldberg and Elhadad [GE10] is a non-directional greedy search procedure used for dependency parsing. The parsing procedure is initialised with all the words in a sentence. The system iteratively forms partial

structures by merging existing partial structures, where only the head partial structure is retained in the list of structures available for merging. The procedure terminates when there is exactly one structure remaining, which corresponds to the root of the sentence.

Neural Graph Based Dependency Parsers (NeurDP): We experiment with two neural dependency parsers, the deep biaffine attention based model of Dozat and Manning [DM17] and the neural graph based dependency parser based on Kiperwasser and Goldberg [KG16]. Both the models rely on LSTM based feature extractors for the parsing.

EBM Configurations: Our energy based model configurations are basically graph based parsing framework for dependency parsing. Here, similar to McDonald et al. [MPRH05], we use Edmond-Chu-Liu's algorithm for finding arborescence (directed spanning tree) of minimum weight.[26] We experiment with three variations. Tree-EBM-P, Tree-EBM-F and Tree-EBM*-F. Tree-EBM-P uses PRA vectors as the edge vectors, while Tree-EBM-F uses the edge vectors using FSPG. Here Tree-EBM*-F incorporates the language specific augmentations as discussed in Section 5.2.4. For all our EBM models, the label prediction is fully integrated with our parsing procedure, similar to Carreras [Car07].

Syntactic Linearisation and Poetry Word-ordering

LSTM Based Linearisation Model (LinLSTM): The model [SRS16] essentially is a neural language model [BDVJ03] with a beam search based decoder. First, a language model is learned by feeding an LSTM based neural network with ground truth sequences, i.e., the sequence with desired word order. Beam search is employed for decoding from the language model. We use a beam size of 512, as reported in Schmaltz et al. [SRS16], for all our experiments. This model currently reports the best result in the word ordering task for English [HST+17], even outperforming other syntax based linearisation models.

Seq2Seq Beam Search Optimisation (BSO): This is a Seq2Seq model [WR16], where the training procedure defines a loss function in terms of errors made during beam search. The beam always keeps multiple top-k possible sequence hypotheses as possible solutions. The loss penalises the function when the gold sequence falls off the beam during training. The model during inference is constrained to predict only from the words which are present in the input at the encoder side. Additionally, the model is further constrained to predict only from the entries in the input which are yet to be predicted. While this model outperforms

[26] While McDonald et al. [MPRH05] used the approach to find the maximum weighted arborescence, we use this to find the minimum weighted arborescence

LinLSTM under comparable conditions, LinLSTM still reports the state of the art results in linearisation in English. We use a beam size of 15 and 14 for testing and training respectively, the setting with the best reported scores in Wiseman and Rush [WR16].

Kāvyaguru (काव्यगुरु):[27] This is a Seq2Seq model [KSS+19] with gated CNNs [GAG+17], using a sequence level loss [EOA+18]. The model uses a weighted combination of expected risk minimisation and the token level log likelihood with label smoothing as loss. Additionally, the model uses two pretraining components. One for learning task-specific embeddings, formed by a weighted combination of pretrained word embeddings [KWC18]. Second, a component for generating multiple ordering hypotheses [WCM18] to be used as input to the Seq2Seq component. The model currently reports the state of the art results for syntactic linearisation in Sanskrit.

EBM based Configurations (Beam-EBM-F/ATSP-EBM-F): Here, for both the syntax level and prosody level linearisation tasks, we use the exact same EBM configurations. The only difference will be in the input graph representation. For the tasks, we experimented with two different inference procedures. Horvat and Byrne [HB14] proposed to solve the task of linearisation as that of solving the Asymmetric Travelling Salesman Problem (ATSP) over a graph. We incorporate the same as our inference procedure in the ATSP-EBM-F configuration. Inspired from the works of Schmaltz et al. [SRS16] along with Wiseman and Rush [WR16], we use beam search as the approximate inference in the Beam-EBM-F procedure. We also report the performance score for Beam-EBM*-F, which incorporates the language specific augmentations as discussed in Section 5.2.4. This configuration is used only for the syntactic linearisation task.

Edge Vector Generation

Path Ranking Algorithm (PRA): In Krishna et al. [KSB+18], the authors use the Path Ranking Algorithm (PRA) [LC10] for learning the feature function Here, PRA is performed as a pipeline where the features are first enumerated exhaustively and are then filtered. In Krishna et al. [KSB+18], we performed the filtering using recursive feature elimination (RFE) [KJ97, GWBV02] and Mutual Information Regression (MIR) [KSG04]. As labels to these regressors, we experimented with point-wise mutual information (PMI) and also with word-pair co-occurrence probability (Co-Pr). Though PRA, in principle, can have a morphological

[27] This is the same model we discussed in Section 4.3

constraint tuple ψ of any cardinality, the cardinality is typically restricted by an arbitrary upper bound. We experimented with $|\psi| \leq 1$ and $|\psi| \leq 2$. Altogether, this resulted in eight different configurations of feature sets using PRA. Of the eight, the one which used Mutual Information regression with co-occurrence probability as the label on a set of features with an upper bound of $|\psi| \leq 1$ reported the best performance. We use this as our default configuration for the PRA. The PRA based edge vectors are generated only for syntax level tasks. Hence all the aforementioned settings are valid only for the edge vectors used in syntax level tasks.
Forward Stagewise Path Generation (FSPG): This is the feature learning approach we elaborated in Section 5.3.2.

5.3.3 Results

Evaluation Metrics: For the standalone tasks, we report the standard metrics generally used in reporting the performance of those tasks. For dependency parsing, we use unlabelled Attachment Score (UAS) and labelled attachment scores (LAS). Similarly, for linearisation tasks, we follow Krishna et al. [KSS$^+$19] and report the performance of the systems using, BLEU [PRWZ02], Kendall's Tau (τ) score [Lap03] and perfect match score, i.e. the percentage of sentences with an exact match to the corresponding ground-truth. For word segmentation (WS), morphological parsing (MP) and the joint task of WS and MP, we use macro-averaged Precision, Recall and F-Score [KSB$^+$18]. We adopt the same metric for the joint task of WS, MP and dependency parsing. Due to the possibility of a structural mismatch between the prediction and the ground-truth in the joint setting, the use of standard metrics for dependency parsing, i.e. UAS and LAS, need not be feasible here [MSBT19]. For the same reason, we do not use the Kendall's Tau score for reporting the performance of the joint task of WS, MP and syntactic linearisation. We instead, use the longest common sub-sequence ratio for this joint task, along with BLEU and Perfect match scores.
Tables 5.5-5.9 show the results for each of the five standalone tasks, as discussed in Section 5.1. In all the tasks, we can find that our models using the EBM framework, report the best performance for the tasks. We elaborate the results for each of the tasks and compare the system performance with the baseline models.

Word Segmentation: Table 5.5 shows the result for the word segmentation task. The proposed system of ours, Cliq-EBM-F, reports the best score in terms of macro average Precision, Recall and F-Score for the task. Here rcNN-SS [HN18] scores the best when

System	P	R	F	PM
Sup-PCRW	73.37	82.84	77.82	31.32
EG-CRF	77.66	79.81	78.72	40.46
L-EBM-V	84.11	86.94	85.5	56.52
L-EBM-B	87.12	87.37	87.24	63.17
Prim-EBM-P	88.79	91.37	90.06	64.22
Cliq-EBM-P	94.48	96.52	95.49	81.57
rcNN-SS	97.15	97.41	97.23	**87.75**
Cliq-EBM-F	**98.04**	**98.67**	**98.35**	85.25

Table 5.5 Results for word segmentation.

System	P	R	F
SeqGen	81.79	81.79	81.79
FCRF	80.26	80.26	80.26
EG-CRF	79.48	82.4	80.91
L-EBM-V	79.67	79.67	79.67
L-EBM-B	81.42	81.42	81.42
Prim-EBM-P	85.11	85.11	85.11
Cliq-EBM-P	93.52	93.52	93.52
Cliq-EBM-F	**95.33**	**95.33**	**95.33**

Table 5.6 Results for morphological parsing.

System	τ	BLEU	PM (%)
Transformer	53.72	20.18	6.13
LinLSTM	61.47	35.51	8.22
BSO	65.38	41.22	12.97
ATSP-EBM-F	72.79	48.03	20.18
Beam-EBM-F	73.62	51.81	25.16
Kāvya Guru	75.58	55.26	26.08
Beam-EBM*-F	**78.22**	**59.38**	**27.87**

Table 5.7 Results for syntactic linearisation.

System	τ	BLEU	PM (%)
Transformer	38.71	12.12	2.48
LinLSTM	42.96	15.42	5.18
BSO	48.28	20.12	6.8
Kāvyaguru	58.27	36.49	12.59
ATSP-EBM-F	63.24	42.86	17.67
Beam-EBM-F	**70.4**	**46.88**	**21.16**

Table 5.8 Results for poetry linearisation.

System	UAS	LAS
EasyFirst	73.23	-
YAP	76.69	73.02
Tree-EBM-P	81.07	77.73
Tree-EBM-F	82.65	79.28
Tree-EBM*-F	85.32	83.93

Table 5.9 Results for dependency parsing.

System	P	R	F	PM
EG-CRF	76.69	78.74	77.7	31.82
L-EBM-V	76.88	74.76	75.8	27.49
L-EBM-B	79.41	77.98	78.69	31.57
Prim-EBM-P	82.35	79.74	81.02	32.88
Pipe-Cliq-EBM-P	86.57	83.74	85.13	43.17
Joint-Cliq-EBM-P	91.35	89.57	90.45	55.78
Pipe-Cliq-EBM-F	86.39	85.15	85.77	45.00
Joint-Cliq-EBM-F	**94.04**	**91.47**	**92.74**	**58.21**

Table 5.10 Results for joint task of word segmentation and morphological parsing.

System	P	R	F
Pipe-PEBM-P	69.82	71.95	70.87
Joint-PEBM-P	72.98	74.42	73.69
Pipe-PEBM-F	75.17	76.58	75.87
Joint-PEBM-F	77.65	77.92	77.78
Joint-PEBM*-F	78.68	79.72	79.2

Table 5.11 Joint task of word segmentation, morphological parsing and dependency parsing. Here 'PEBM' denotes 'Prim-EBM'.

System	LCS	BLEU	PM (%)
Pipe-ATSP-EBM-F	38.51	32.78	10.14
Pipe-Beam-EBM-F	40.22	33.73	10.37
Joint-ATSP-EBM-P	46.75	36.67	11.83
Joint-Beam-EBM-P	51.44	40.18	13.92
Joint-ATSP-EBM-F	49.72	38.04	12.53
Joint-Beam-EBM-F	53.82	43.39	16.87
Joint-Beam-EBM*-F	**55.06**	**47.27**	**18.66**

Table 5.12 Joint task of word segmentation, morphological parsing and syntactic linearisation.

it comes to the perfect match score. Cliq-EBM-F reports a percentage improvement of 1.15 % over rcNN-SS in F-Score.[28] The system named as Cliq-EBM-P was the system which was proposed in Krishna et al. [KSB$^+$18]. Previously, the word segmentation systems rcNN-SS and Cliq-EBM-P showed that they both outperform the then state of the art, a Seq2Seq model[29] from Reddy et al. [RKS$^+$18]. But, a comparison between both these systems was not available. Here, we find that the rcNN-SS outperforms Cliq-EBM-P with a percentage improvement of 1.82 %.[28] But, both Cliq-EBM-P and the best performing model Cliq-EBM-F were trained on 8,200 sentences which is just about 1.5% of the training data (> 0.55 million) used in rcNN-SS.

All the competing systems for word segmentation, except for rcNN-SS, use the linguistically refined output from SHR as their search space to predict the final solution. rcNN-SS uses a training data of more than half a million sentences, while all our EBM configurations use less than 10,000 training data. EG-CRF was trained on 10,000 sentences, whereas VL-EBM and BL-EBM were trained on 9,000 sentences after which the models got saturated. The Prim-EBM-P, Cliq-EBM-P/F all use a training data of 8,200 sentences. EdgeGraphCRF being a second order CRF model [Ish11], does not take the entire sentence context into account. Also, the QBPO inference does not guarantee the prediction of exhaustive segmentation. In fact, about 85.16% of the sentences predicted by the model from DCS10K do not correspond to an '*exhaustive segmentation*'. Prediction of an '*exhaustive segmentation*' is guaranteed in

[28] Following Dror et al. [DBSR18], we perform pairwise t-tests between the reported scores for these systems, and find that the scores are statistically significant ($p < 0.05$).

[29] This is the same model we discussed in Section 4.1

all our EBM models (also in supervisedPCRW) by virtue of the inference procedure we use. Cliq-EBM-P and Cliq-EBM-F differ only in the vector generation module used. Between these two configurations, the use of FSPG based vectors result in a percentage improvement of about 3 % in the results.[28]. In the lattice based configurations, VL-EBM and BL-EBM, the edges are formed only to position wise immediate nodes in an exhaustive segmentation. Also, the inference procedures process the input sequentially, where it considers the input from left-to-right. The approach substantially reduces the performance of the system. This justifies the additional cost incurred due to use of our non-sequential inference procedure. Similarly, the Prim-EBM-P performs substantially worse as compared to the Cliq-EBM-P. Here, the latter considers all the pairwise potentials between the inputs, while the former does not.

Morphological Parsing: Table 5.6 shows the results for the morphological parsing in Sanskrit. We find that similar to word segmentation, our proposed EBM configuration, Cliq-EBM-F provides the state of the art results for morphological parsing. Currently, there exists no morphological parser that performs both analysis and disambiguation of morphemes in Sanskrit, leaving aside the Cliq-EBM-P configuration reported in Krishna et al. [KSB$^+$18]. Instead, as baselines, we employ two widely used neural sequence taggers which reported state of the art results on multiple morphologically rich languages, the FCRF [MGN18] and SeqGen [TS18]. But they predict only the morphological tag of each word. For all the other models, we predict all the morphemes of a word including its stem. All the EBM configurations we use for morphological parsing, are used for the word segmentation task as well. Among the models with a sequential processing of input, SeqGen reports the best F-Score of 81.79. Our lattice based EBM configuration reports an F-Score of 81.42 %, close to SeqGen. Cliq-EBM-F, our best performing model reports a percentage improvement of 16.55 % over FCRF. All the EBM-Configurations with graph based input representation and a non-sequential inference procedure, in general, report a score which is significantly greater than the sequential models. Further, Cliq-EBM-F reports a percentage improvement of about 2 % in comparison to Cliq-EBM-P, where the former uses FSPG based edge vector generation approach. All our EBM configurations used the same amount of training data as used in Word-segmentation, which is less than 10,000. At the same time, the neural sequence taggers were trained on a dataset of 50,000 sentences. Given we use the segmented sequence as input, the precision and recall will be the same for all the systems as the number of predictions will match with the number in the ground-truth. As the QBPO inference used for EG-CRF does

not guarantee the prediction of an exhaustive segmentation, this need not be true for EG-CRF.

Dependency Parsing: Table 5.9 shows the results for dependency parsing in Sanskrit. Here, our energy based model configurations outperform the other baseline models for the task. Among the EBM configurations, Tree-EBM*-F reports the best score of 85.32 UAS and 83.9 LAS. Tree-EBM*-F is an EBM configuration where we incorporate language specific knowledge as constraints. We find that the configuration reports percentage improvement of 3.23 % from the reported performance of Tree-EBM-F (UAS of 82.65). Further, we find performance improvements in LAS are much more substantial in Tree-EBM*-F, with 5.86 %, as compared to LAS of 79.28 of Tree-EBM-F. We hypothesis that incorporating linguistic constraints in Tree-EBM*-F helped much more in disambiguating between the edge labels. YAP [MSBT19], reports UAS score of 76.69 and LAS score of 73.02, lower than the least performing EBM configuration, Tree-EBM-P. YAP currently reports the state of the art score for dependency parsing in Hebrew, outperforming previous models, including Seeker and Çetinoğlu [SÇ15]. It may not be fair to compare YAP with Tree-EBM*-F due to the latter's advantage in explicit linguistic information. But as a fair comparison of two language agnostic dependency parsing models, Tree-EBM-F reports a performance improvement of about 7.77 % as compared to the YAP. We experimented with neural models for dependency parsing such as Dozat and Manning [DM17] and Kiperwasser and Goldberg [KG16], but both the models had UAS/LAS scores below 50. All of the models used a training data of 12,000 sentences, as described in Section 5.3.1. Given that the neural models rely on the network architecture for automatic feature extraction, we hypothesise the lack of sufficient gold standard training data may have resulted in their poor performance.

Syntactic Linearisation: Table 5.7 shows the result for syntactic linearisation. Here, Beam-EBM*-F, our model that incorporates additional linguistic knowledge about Sanskrit syntax, performs the best in comparison to the other models. The model improves the performance by more than 4 BLEU points (percentage improvement of 7.45 %), in comparison to *kāvyaguru*, the current state of the art for linearisation in Sanskrit. Altogether we report performance of three different configurations of EBM for the task, of which *kāvyaguru* outperforms two of the configurations. But, all the neural sequence generation models, including *kāvyaguru* were trained with additional training data of 95,000 from Wikipedia amounting to a total of 108,000 training sentences. Our EBM models were trained only on 10,000 sen-

tences. LinLSTM, performed worse when additional data was used. Hence we report score of linLSTM where the additional data was not used. We experiment with two different inference procedures, ATSP-EBM-F and Beam-EBM-F. We find that the beam search inference with a beam size of 128 works the best for the task. Larger beams result in diminishing returns, with improvements not being significant but incurring computational overheads at the same time.

Poetry Linearisation: Table 5.8 shows the result for poetry linearisation. Here, we find a significant difference between both the EBM configurations presented and other baselines. *kāvyaguru* reports the best score among the non-EBM configurations. But, for Beam-EBM-F the score jumps by about 10 BLEU as compared to *kāvyaguru*. Similarly, *kāvyaguru* performs significantly better than other neural sequence generation models. We observe that the improvement appeared only after we incorporated prosody information, i.e., explicitly incorporating syllable weight information and then filtering the sequence that adheres to valid patterns of syllable weights. This was incorporated in one of the pretraining steps used to generate multiple hypotheses for the final Seq2Seq component of *kāvyaguru*. The BLEU score improved from 23.16 to 36.49 with this additional information. For LinLSTM, we experimented with filtering sequences with only valid patterns of syllable weights, during its beam decoding. But this does not lead to any improvements. We find that larger beam sizes with a size of 512, as against 128 for the syntax level task, lead to better results for the task. Probably, this difference can be attributed to larger path lengths of the final solution in a poetry task as compared to a syntax level task due to the graph construction (Figure 5.7, merged Nodes). While the EBM configurations were trained on a dataset of 13,000 verses from Rāmāyaṇa, the neural sequence generation models were trained on a dataset of 113, 000 verses. The additional 100,000 verses were taken from DCS and also from vedabase[30].

Joint and Pipeline Modelling of Tasks: Tables 5.10-5.12 show the results for the three joint tasks we perform using the framework. For all the three tasks, we observe similar trends to what is observed in the previous research [SÇ15,MSBT19], that the joint modelling of the tasks outperforms the pipeline based models. In Table 5.10 we report the task of joint word segmentation and morphological parsing as reported in Krishna et al. [KSB+18]. Table 5.6 shows the results for morphological parsing when using the gold standard segmented sentence as input. Since the EBM models used for the segmentation task have access to

[30] https://vedabase.io/

the morphological information, we use word segmentation predictions from rcNN-SS for the pipeline model. The predictions from rcNN-SS are then passed onto SHR as input for the morphological parsing task. The results are shown in Table 5.10. The pipeline models perform far inferior to the corresponding joint models for the task. For Cliq-EBM-F the joint version reports a percentage improvement of 8.13 % over its pipeline based counterpart. Similarly, the results for the joint task of word segmentation, morphological parsing and dependency parsing are shown in Table 5.11. It is evident that the pipeline system performs inferior to the joint modelling. The joint model of Prim-EBM-F reports an F-Score of 77.92 as against an F-score of 75.87 for the pipeline model. Table 5.12 shows the performance of pipeline and joint models for the joint task of Word-segmentation, Morphological parsing and Syntactic Linearisation. Here, the pipeline based model is outperformed by the joint model with a significant margin of about 10 BLEU points, in the case of Beam-EBM-F. The EBM* configurations for the dependency parsing and syntactic linearisation, which use additional constraints, may lead to disconnected components in the input of the pipeline model. This can happen due to the errors in predictions from the word segmentation or morphological parsing. Hence this configuration cannot be used in the pipeline setting, and we, therefore, do not compare the pipeline vs joint setting for this configuration. In the pipeline configurations of both these tasks, we perform the word segmentation and morphological parsing jointly using Cliq-EBM-F/P models. For both the tasks, use of rcNN-SS for segmentation in the pipeline resulted in inferior performances, as compared to joint modelling of the upstream tasks using EBM.

Training Specifics

Effect of Feature Generation Approach: FSPG, the feature function learning approach we introduce in this task, consistently outperforms the previously employed [KSB$^+$18] PRA based feature generation approach for all the tasks we experimented with. Table 5.13 reports the Precision, Recall and F-score on the word segmentation task, for all the different sets of vectors generated. We find that the best performing set is using the FSPG with an F-Score 98.35. Among the eight different configurations for PRA based vectors (Section 5.3.2), the best configuration (Table 5.13, Row 3), by design, considers a feature space where all the features have $|\psi| \leq 1$. This implies all the features learnt will have a constraint tuple of length at most 1. We observe a performance drop when we change this setting to $|\psi| \leq 2$ (Table 5.13, Row 4). On the contrary, FSPG does not require any such upper bound on

Upper bound on $\|\tau\|$	Feature Selection	Label	P	R	F
1	MIR	PMI	92.86	95.27	94.05
2	MIR	PMI	82.18	86.17	84.13
1	MIR	Co-Occ	**94.48**	**96.52**	**95.49**
2	MIR	Co-Occ	93.65	95.52	94.57
1	RFE	PMI	80.93	87.65	84.16
2	RFE	PMI	84.85	86.90	85.86
1	RFE	Co-Occ	83.69	89.11	86.31
2	RFE	Co-Occ	86.34	89.47	87.88
-	FSPG	Co-Occ	**98.04**	**98.67**	**98.35**

Table 5.13 Effect of feature selection approach tested on the word segmentation model.

the length of the constraint tuple ψ. This results in a quite diverse set of features. FSPG has only 54.94 % of its features with a tuple size $|\psi| \leq 1$ for the syntax level tasks. Interestingly, 17.06 % of paths have $|\psi| \geq 3$, and the longest path was of size 4. In the case of prosody level FSPG features, we find that all the features have a tuple size $|\psi|$ between three and seven. In fact, more than 50 % of the paths have a tuple size $|\psi| \geq 5$. Further, we learn feature sets of different sizes using both FSPG and PRA. For FSPG, we experimented with feature sets of sizes starting from 300 till 1,000, in step sizes of 50. We find that the model performs the best with a feature set size of 850 for syntax level tasks, and 500 for the prosody level task. Similarly, for PRA, we vary the feature set size from 400 to 2000 in steps of 100, and we achieve the best results for a feature set with a size of 1500. The feature generation process takes about 9 to 11 hours when using the PRA approach [KSB$^+$18, supplementary]. We could reduce the feature generation time to about 4 hours for syntax level tasks, and 3 hours for the prosody level tasks for the FSPG vectors. This is primarily due to the use of 'GreedyTree' structure in FSPG, which avoids the need for an exhaustive enumeration of all the features which was a bottleneck in PRA. The generation of feature vectors at runtime is performed using PCRW, irrespective of the feature generation approach. But, owing to the lesser number of features in the feature set when using FSPG, a feature vector for syntax level tasks is generated in 38 ms as compared to 66 ms for PRA. For prosody level tasks

the time is about 23 ms (with 500 vector components).[31]

k	WS			WS+MP		
	P	R	F	P	R	F
5	90.46	92.27	91.36	83.52	80.48	81.97
10	92.92	95.07	93.98	85.32	84.4	84.86
15	94.85	96.14	95.49	87.67	86.38	87.02
20	**95.23**	**96.49**	**95.86**	**89.25**	**88.62**	**88.93**

Table 5.14 Performance of Cliq-EBM with pruned edges in G.

Limiting the Context by Pruning Edges in the Input Graph X: From the tasks of Word-Segmentation, Morphological Parsing and their joint task formulation, we make two important observations. In these tasks, the EBM configurations which use inference procedures with non-sequential processing of input outperform those which perform the input sequentially. Between Prim-EBM and Cliq-EBM configurations, the two configurations with non sequential processing, Cliq-EBM which considers pairwise potentials between all the pairs of non-conflicting words leads to significant performance improvements than Prim-EBM. But, to get a better understanding of the impact of pairwise potentials between every non-conflicting words, we perform experiments where the edges are pruned from the original input graph X, if two words are separated by more than k characters between them. For any two words appearing in an exhaustive segmentation, we keep an edge only if both the words overlap within a distance of k characters. We experiment with k = 5, 10, 15 and 20. Hence, for $K = 20$, a word will form edges with all the words that fall within 20 characters to the left and 20 characters to the right. The average length of an input sequence in the test data is 40.88 characters. We do not modify our inference procedure in Cliq-EBM-P other than to take care of the possibility that a clique need not always be returned. Table 5.14 shows the results for different values of k. Interestingly, the results show a monotonic increase with the increase in context window, and the results with the entire context are still better than those with $k = 20$, even though only marginally. It is interesting to note that, keeping the entire context does not adversely affect the predictions as none of the pruned models outperform the original configuration in Cliq-EBM-P. The lattice structure can be seen as an extreme case of pruning. We modify VL-EBM-P to use a non-sequential inference procedure, adapted from Prim-

[31] The vector generation can be performed in parallel, and we use 40 threads in our experiments

Type	WS Recall			WS+MP Recall		
	Prim-EBM	Cliq-EBM		Prim-EBM	Cliq-EBM	
	-P	-P	-F	-P	-P	-F
Nominals	93.06	96.87	98.07	86.14	89.0	91.33
Verb	89.14	95.91	96.84	87.38	94.42	95.14
Compound	89.35	93.52	95.69	86.01	91.07	92.86
Indeclinable	95.07	97.09	98.26	94.93	96.47	97.95

Table 5.15 System performance on the inflectional classes (Table 5.2) for the competing systems Clique-EBM-P, Cliq-EBM-F and Prim-EBM-P.

EBM-P. Here, the start and end markers were removed. Additionally, a given connected node pair has 2 edges, each in either of the directions. We find that the model gives an F-Score of 87.4, which outperforms the original VL-EBM-P configuration 4 by more than three points.

Morphological Class Specific Assessment: Table 5.15 presents the micro-averaged recall for the words grouped based on their inflectional classes (Table 5.2) for Clique-EBM-F, Clique-EBM-P and Prim-EBM-P. For word segmentation (WS), the surface-forms belonging to indeclinables and nominals have the highest recall. In the case of joint word segmentation and morphological parsing (WS+MP), the recall over nominals (surface-form and tag) shows the highest decline among the inflectional classes. This difference in recall for nouns is mostly arising due to mispredictions in the morphological tag of the word, rather than the surface-form. We find that considering the pairwise potential between all the words in a sentence in Clique-EBM-P/F led to an improved morphological agreement between the words in comparison to Prim-EBM-P. The difference is particularly visible in the case of improvements in the verb prediction.

Error Propagation in the Pipeline and Joint Models for Word-segmentation, Morphological Parsing and Dependency Parsing: Table 5.16 shows the error analysis over the entire test data of dependency parsing. We analyse the pipeline model and both the Joint models Prim-EBM-F and Prim-EBM*-F, for the task. As we can expect, the largest gap in errors between the pipeline model and the joint model is due to the mispredictions in word segmentation and in morphological parsing. At the same time, the reduction in mispredictions in the remaining categories of errors between these models is lower. Now,

Prim-EBM-F	Pipe	Joint	
		EBM	EBM*
WS	425	322	297
MP	703	547	520
Label mismatch	404	372	294
Others	886	827	777
Total Errors	2418	2068	1888

Table 5.16 Error propagation in the pipeline (pipe) and joint models for Word segmentation (WS), Morphological parsing(MP) and dependency parsing.

Beam-EBM-F	Pipe	Joint	
		EBM	EBM*
WS	318	233	204
MP	356	241	226
Dep. Locality	213	197	144
Others	486	408	397
Total Errors	1373	1079	971

Table 5.17 Error propagation in the pipeline (pipe) and joint models for word segmentation (WS), morphological parsing(MP) and syntactic linearisation. Here Dep. locality denotes dependency locality.

Among the joint models, Prim-EBM*-F (marked as EBM*) shows considerable reduction in errors under 'label mismatch' category as compared to Prim-EBM-F. Prim-EBM*-F is the EBM configuration with language specific augmentations as described in Section 5.2.4. Here, 'Label mismatch' category specifically looks into those mispredictions that occur due to the many-to-many mapping between the case markers of the words and the dependency tags (kāraka). In this category, we consider those mispredictions, where the reference tag and the mispredicted tag are applicable for the case markers of the words and hence are more susceptible to be mispredicted.

Error Propagation in the Pipeline and Joint Models for Word-segmentation, Morphological Parsing and Syntactic Linearisation: Table 5.17 shows the performance difference between pipeline based model and the joint models for the poetry to prose linearisation task. Here, we perform a qualitative analysis on a sample of 300 sentences of the test data for linearisation.[32] It can be observed that the pipeline model suffers due to errors from the word segmentation and morphological parsing, compared to either of the joint models. Among the joint models, Beam-EBM*-F considerably makes fewer errors in the dependency locality category as compared to the errors made by Beam-EBM-F. This is due to the language

[32]We restricted ourselves to a sample of the test data due to the requirement of manual inspection of data to detect the errors in the 'dependency locality' [GFP+19] category. One of the authors and two other annotators with a graduate level degree in Sanskrit linguistics performed this. The 300 ($\times$ 3, as there are three systems to be considered) sentences we filter have an inter annotator agreement of at least 2 of the three annotators.

specific augmentations (Section 5.2.4), made to Beam-EBM*-F mitigate the errors from dependency locality. We observe that about 39.62 % of the reduction in error in this category is achieved specifically in the case of placement of adjectives near to the word it modifies.

Merging of Syllable-level Nodes in Poetry Linearisation: For the graph generation in the prosody level tasks (Section 5.2.1), we define the vertices at a syllable level. At the same time, we have made a modification by merging the nodes which have only one outgoing edge (Figure 5.7). This node merging substantially reduces the number of nodes and edges in the input graph X, as well as the edge vectors generated. Empirically we find that this modification results in an overall improvement of system performance. While the best performing model of ours using the modified structure performs with a BLEU Score of 46.88 and a Kendall's Tau score of 68.4, the original syllable-level representation results in a BLEU of only 41.93 and a Kendall's Tau score of 64.21.
Summarily, FSPG significantly and consistently outperforms PRA, in learning effective feature functions for all the tasks. Our non-sequential method of inference results in better performance in comparison to the sequential models even for low level tasks such as word segmentation and morphological parsing. The performance gain of Clique-EBM over Prim-EBM illustrates the effectiveness of this approach. Joint modelling of the tasks mitigate the error propagation from upstream tasks, compared to a pipeline based approach. Language specific augmentations in pruning the search space and filtering the candidates during inference reduces errors due to label mismatch and dependency locality for dependency parsing and syntactic linearisation respectively.

5.4 Conclusion

Our work presents a generic search based structured prediction framework for numerous structured prediction tasks in Sanskrit. We either achieve state of the art results, or ours is the only data-driven model for all the tasks we experiment with. In fact, we introduce the task of poetry linearisation. While the framework is language agnostic, it still enables to encode language specific constraints which enable to prune the input search space as well as in filtering the candidates during inference. It also facilitates joint modelling of multiple tasks, which we plan to extend to further downstream semantic tasks as well. But more importantly, we achieve a substantial reduction in task specific training data requirements for all the tasks

by incorporating rich linguistic information, both in the form of the search space as well as in our feature function. This is particularly important for a low resource language like Sanskrit. Another novel aspect of this work is the automated learning of feature function. For one, this shifts the burden of designing effective features for each task from human domain experts and automates it. The only domain information required here is to define the literals required for the generation of horn clauses. This makes it adaptable to different scenarios, as witnessed in the case of syntactic and prosodic tasks in our case. The use of FSPG based feature generation not only improved the system performance, but it also substantially reduced the time taken for this, as we do not require an exhaustive enumeration of the features.

Chapter 6

Conclusion

The thesis focused on developing data-driven solutions for processing Sanskrit texts by addressing language specific characteristics. We argue for the harmonious integration of linguistic and distributional knowledge into data-driven pipelines for the processing of texts. We not only show that these approaches are effective but are also crucial for the development of NLP solutions for a language like Sanskrit.

In the first contributing chapter (Chapter 3), we focused primarily on integrating the traditional linguistic sources, such as the grammar and lexicons that a human learner would often rely on, into a statistical learner, i.e. a data-driven model. Chapter 4 focused on providing scalable, language agnostic, purely statistical solutions to three of the sequence level tasks. The motivation here was to obtain scalable performance driven solutions so as to catalyse the efforts in digitising Sanskrit texts. Here, we focused on post-OCR text correction, as a use-case of a task that has limited applicability of linguistic knowledge in solving it. Further, we considered the linearisation task, which can be seen as a task that is dependent on outputs from multiple preliminary tasks to address it linguistically. Instead, we proposed a scalable data-driven solution that directly addresses the problem without relying on the outputs from its preliminary tasks. The final chapter (Chapter 5) focuses entirely on a common framework for multiple structured prediction tasks in Sanskrit. Here, we focus on integrating powerful learning mechanisms that can use rich linguistic structures and features to provide effective NLP solutions to various tasks in Sanskrit. Here, our linguistic integration has substantially reduced the data requirements for our tasks, as compared to their neural counterparts.

6.1 Summary of the Contributions

- Computational treatment of *Aṣṭādhyāyī* (अष्टाध्यायी) mostly centred around formal language theory analysis and its applications in rule based systems [PK12, Hym09, GHK+12]. In this thesis, we successfully show that the rules from the grammar can be directly used as effective features for data-driven modelling of tasks in Sanskrit.

- For identifying derivational nouns, we do not just confine to the surface level pattern differences between the source and derived words, but we also consider the string transductions that occur during the derivation process. Here, we again make use of the derivation rules from *Aṣṭādhyāyī* to identify such transductions, especially the stem-internal transductions.

- We introduced data-driven solutions for both the word formation tasks, i.e. compound type identification and identification of derivational nouns. Currently, these are the state of the art models in these tasks.

- It was previously observed in Schnober et al. [SEDDG16] that neural Seq2Seq models could not outperform traditional sequence labelling models in monotone sequence labelling tasks, including the Post-OCR text correction task. On the contrary, we showed that incorporating the copying mechanism into a standard Seq2Seq model equips such models to achieve state of the art results in the post-OCR text correction task.

- To combat the limitations with the availability of training data, we synthetically generate images from digitised texts as training data for the Post-OCR text correction task. Here, we systematically incorporate various distortions to emulate the noise settings in real world scanned images.

- We achieve state of the art results for the Post-OCR text correction task, where we outperform the higher-order pruned-CRF model used by Schnober et al. [SEDDG16]. Our model outperformed the commercial offering from Google OCR as well. Further, a human judgement survey corroborated the results as the human subjects found it easier and faster to recognise the original text from the predictions from our system.

- We introduce the task of converting a verse into its corresponding prose order, which facilitates easier verbal cognition. We propose two solutions, one a purely statistical

solution using a Seq2Seq model with a sequence level loss and another a linguistically involved solution, as part of the structured prediction framework. The latter currently reports the state of the art performance for the task.

- The verse to prose order conversion task was formulated as a linearisation task, or more specifically a syntactic linearisation task, so as to avoid the need for parallel verse and prose order sequences. Further, this enabled us to incorporate prose only sentences as auxiliary training data for our Seq2Seq based solution.

- We introduce the task of poetry linearisation. The task here is to arrange a given bag of words into a sequence such that it adheres to a prescribed metre pattern in Sanskrit Prosody. The task is computationally more challenging than the syntactic linearisation task. In both the linearisation tasks, we need to identify a permutation from a search space that is exponential to the number of words in the input. But, in poetry linearisation, we additionally need to consider the possibility of *Sandhi* between every adjacent pairs in a permutation as Sandhi operation may alter the metre pattern.

- We propose a general graph-based parsing framework for multiple structured prediction tasks in Sanskrit. The framework we propose facilitates to model multiple related tasks jointly as well as independently. Consistent with observations in other morphologically rich languages, we find that joint modelling of the tasks, wherever applicable, is more effective than a pipeline configuration of the same set of tasks. We perform both syntax level and prosodic level tasks using the framework. Using the framework, we find that non-sequential processing of input outperforms sequential processing approaches even for low level tasks such as word segmentation and morphological parsing for Sanskrit.

- The framework by default is language agnostic, as it just expects a graph as input. The domain information is encoded in the nodes, and the dependencies between them are captured using the edges. But, the framework also facilitates to encode language specific information by pruning of search space or filtering the candidates during inference. This shows a marked increase in the performance of our models for the dependency parsing and syntactic linearisation tasks.

- We automate the learning of feature function for all the tasks. In fact, we use the exact same features for all the syntax level tasks, in both joint and pipeline based

implementations as well. Here, we learn the features as arbitrary length horn clauses, from an infinite space of possible clauses defined by a finite set of literals. The literals suffice to be the only domain information we require. We use only the knowledge of syllables and syllable weights for obtaining features for the prosody level task. In the case of syntax level tasks, we incorporated additional constraints that filtered some horn clause formations. This is particularly a novel aspect we introduce, especially when morphologically rich languages still rely on hand-crafted features for morphosyntactic tasks [MSBT19, Mor16].

- The framework, by its design, requires considerably less training data as compared to a standard neural sequence level model. All our models use as low as 10 % of the training data as required by the current neural state of the art models for various tasks. To be more precise, we use just 1.5 % of the training data as compared to the state of the art sequence labeller for word segmentation [HN18]. We also use just 9.26 % of training data for the syntactic linearisation as compared to the Seq2Seq model we propose.

- For all the tasks we proposed, we either achieve the state of the art results, or ours is the only data-driven solution for the task.

6.2 Future Work

The thesis focused on designing NLP solutions for lexical, syntactic and prosodic level tasks in Sanskrit. In the course of our investigations, we identify several new research directions that can be seen as a natural extension to our efforts here. We discuss a few of those possibilities.

Downstream Semantic and Information Extraction Tasks in Sanskrit In this thesis, we developed a structured prediction framework that performs core NLP tasks such as word segmentation, morphology and syntax level tasks for Sanskrit. Here, we list a few downstream tasks which we believe are important and challenging in the processing of texts in Sanskrit, but are often constrained by lack of available data resources. To begin with, the tagging scheme that we used for the dependency analysis is based on the *kāraka* theory, the traditional framework for syntactic analysis of sentences in Sanskrit [BKS19]. The relations as per the *kāraka* theory are known to be of syntactic-semantic in nature [BS93]. This was

a conscious design decision we made, which leaves us with potential scope for expanding the framework for a semantic labelling task. Given the link between the morphological case markers to *kāraka* tags as per traditional Sanskrit grammar, we hypothesise that modelling of semantic labelling tasks such Semantic role labelling might benefit from jointly modelling it with its preliminary upstream tasks. Though currently, we address the problem of compound splitting as well as compound type identification, the compound type identification is not fully integrated into the structured prediction framework. The major challenge here is to identify the headword of the compound. The headword is relevant to the sentence context, especially in the case of multi-component compounds. Since any compound can be analysed by decomposing it into two immediate component nouns, a multi-component compound can be seen as the outcome of a recursive compounding procedure. The head and the semantic type of the final step in recursive compounding procedure in such cases might be of relevance for understanding the sentence.

As a consequence of language usage, texts in Sanskrit pose a novel challenge pertaining to co-reference resolution. In literary works of Sanskrit, a person, i.e. a named entity, need not be addressed by their given name, but often by a quality, trait or lineage of the person which is relevant to the context. For instance, in *Mahābhārata* (महाभारत), the warrior-prince *Arjuna* (अर्जुन) is addressed using 10 different names, if not more. Here each name either represents his skills, appearance or his lineage. For instance, if the context demands *Arjuna* to be referred to as a warrior, he is addressed as *bībhatsu*[1] (बीभत्सु), which means "one who never commits loathsome or horrible acts on the battlefield". Similarly to invoke or remind Arjuna about his lineage, he might be addressed as '*kaunteya*' (कौन्तेय) or '*pāṇḍava*' (पाण्डव) implying son of *Kunti* (कुन्ति) or son of *Pāṇḍu* (पाण्डु) respectively. Now many of these are compounds, often non compositional, or are derivational nouns. These instances need not be effectively resolved by a direct application of existing techniques currently used in anaphora resolution or in knowledge based approaches. Disambiguation of such co-references might require more semantically involved processes for Natural Language Understanding. Nevertheless, literature in Sanskrit is filled with such instances, but with a caveat that availability of task specific labelled datasets will be the major challenge here. In general, development of task specific labelled data will be a challenge for all the aforementioned tasks.

[1] https://www.sanskrit-lexicon.uni-koeln.de/scans/PEScan/2014/web/webtc/servepdf.php?page=145-b

User-feedback, Co-active Learning and Cross-lingual Learning A major recurring source of concern in processing Sanskrit texts is probably the availability of annotated data. Acquiring annotations can be linguistically involved and time consuming, making it a challenge in itself. To catalyse such annotation efforts, we plan to explore the possibility of relying on humans to provide feedback to the predictions made by our existing systems, instead of relying on them to provide complete annotations. This is particularly relevant for the sentence level predictions for tasks such as word segmentation and morphological parsing from our structured prediction framework. The predictions often have considerable overlap with the ground-truth, but differ only on the basis of few mispredicted components. Further, the structured prediction framework can be trivially modified to output ranked list of predictions, instead of a single prediction. In such scenarios, the perfect match score for the joint word segmentation and morphological parsing jumps from 58.21 % to 76.51 %, if we consider top 5 predictions from our system, instead of exactly one prediction. Under such a case, we find that developing an interactive learning system as a post-editing tool might be of utility. Here we expect that the proposed system inputs ranked predictions from an offline structured prediction framework like ours, and then learns to improve upon the predictions based on user feedback. We plan to incorporate online structured prediction approaches such as co-active learning [SJ12b], which can be used to develop such interactive learning systems. Such approaches would greatly alleviate the burden on the annotator and yet, aid in developing robust learning frameworks. We leave this for future work.

This thesis focused entirely on solutions for processing Sanskrit texts, where we incorporate grammatical and linguistic information from the grammatical tradition of Sanskrit. Several indic languages are influenced by Sanskrit grammar. In fact, the Kāraka scheme for dependency analysis was enriched and extended for other languages, primarily for other Indic languages [BS90, GPMS14, CSS13]. But the applicability of Sanskrit grammar in other core-NLP tasks is yet to be explored. For instance, the Dravīdian language 'Malayalam' shares considerable overlap with Sanskrit in morphological derivation, compounding as well as in rules of Sandhi [Var96]. Similarly, other Indic languages such as Telugu, Marathi and Kannada have considerable overlap both lexically and grammatically with Sanskrit [BCSR95, Bha89]. Given that most of these Indic languages are low-resource in nature, we could exploit this considerable overlap in linguistic regularities not just to share the linguistic knowledge but also to facilitate parameter sharing across trained models in these languages.

Synthesising Logic Programs *Aṣṭādhyāyī*, the grammatical treatise of Sanskrit, is generally accepted to carry many traits that would qualify it as a generative grammar [Blo29, Kip94]. In fact, the rule descriptions for *Sandhi* (सन्धि) may be unambiguously decoded as algebraic rewrite rules which can often be related to contemporary morphophonetic rules in computational linguistics [Hue15]. To quote Huet [Hue15], 'Indeed, *Pāṇinīan* rules may be directly fed into the finite state toolkits implementing this paradigm". This would leave any information scientist to wonder, how could someone come up with a complete description of a natural language (spoken at the time), and fit the entire grammar description into merely 4,000 rules. Inspired by the algebraic structure expressed in the Pāṇinian framework of grammar for Sanskrit [PS92, Kip10, Pet05], we attempt to obtain similar grammars for other languages. As a subtask, we have been focusing on learning rules for the morphological (re)inflection task for 44 languages. The work essentially attempts for feature subset selection, followed by learning of constraints for building a hierarchical ordering of the rules. Summarily, the proposed approach essentially relies on a complete lattice (Formal Concept Analysis) to learn valid bijections between maximal subsets of inputs (intents) and maximal subsets of features (extents). In Formal Concept Analysis, such pairs are collectively called as concepts. We score each concept using the information gain it provides and then compares it to its immediate general concept(s). This is followed by the construction of a logic program, using denotations from the concepts, where conflicting concepts are scored using Energy-Based Models. The program so formed is validated using SAT Solvers. The whole process is iteratively performed with an objective to minimise the description length of the program so obtained using the Minimum Description Length. Our initial investigations show promising outcomes as compared to standard grammar induction techniques such as Adaptor grammars on the task for all the 44 languages we have evaluated.

Bibliography

[AAAK04] Shlomo Argamon, Navot Akiva, Amihood Amir, and Oren Kapah. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of Coling 2004*, pages 1058–1064, Geneva, Switzerland, aug 23–aug 27 2004. COLING.

[AB06] V Sheeba Akshar Bharati, Amba P Kulkarni. Building a wide coverage morphological analyser for sanskrit: A practical approach. First National Symposium on Modeling and Shallow Parsing of Indian Languages,IIT Mumbai, 2006.

[ABCC06] David L. Applegate, Robert E. Bixby, Vašek Chvatál, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[ABN+10] Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, Dipak Kumar Basu, and Mahantapas Kundu. Recognition of non-compound handwritten devnagari characters using a combination of mlp and minimum edit distance. *International Journal of Industrial Electronics and Electrical Engineering*, 2010.

[AG07] Galen Andrew and Jianfeng Gao. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.

[AGP+18] Rahul Aralikatte, Neelamadhav Gantayat, Naveen Panwar, Anush Sankaran, and Senthil Mani. Sanskrit sandhi splitting using seq2(seq)2. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4909–4914, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[Aiy10] Śrīnivāsa Aiyaṅkār. *The Ramayana of Valmiki*. Madras : Little Flower Co, 1910.

[AJH03] Yasemin Altun, Mark Johnson, and Thomas Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan, 2003. Association for Computational Linguistics.

[ASA+18] Devaraj Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K Ramasubramaniam, and Malhar Kulkarni. Improving the learnability of classifiers for sanskrit ocr corrections. In *The 17th World Sanskrit Conference, Vancouver, Canada*. IASS, 2018.

[BCB15] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representation (ICLR)*, San Diego, US, 2015.

[BCSR95] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi, 1995.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[Bel17] David Belanger. *Deep Energy-Based Models for Structured Prediction*. PhD thesis, University of Massachusetts Amherst, 2017.

[BGE16] Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 490–500, Austin, Texas, November 2016. Association for Computational Linguistics.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[Bha89] Saroja Bhate. *Panini's taddhita rules*. University of Poona, 1989.

[Bha90] Vinayak P. Bhatta. Theory of verbal cognition (Śābdabodha). *Bulletin of the Deccan College Research Institute*, 49:59–74, 1990.

[Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[BK73] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[BK03] Kenneth R Beesley and Lauri Karttunen. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*, 2003.

[BKS19] Akshar Bharati, Amba Kulkarni, and Dipti Misra Sharma. Pāṇinian syntactico-semantic relation labels. In *Proceedings of the Fifth International Conference on Dependency Linguistics, DepLing 2019 (Accepted)*, 2019.

[Blo29] Leonard Bloomfield. *Language*, 5(4):267–276, 1929.

[BMK+04] Ajay S Bhaskarabhatla, Sriganesh Madhvanath, MNSSKP Kumar, A Balasubramanian, and CV Jawahar. Representation and annotation of online handwritten data. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 136–141, Tokyo, Japan, 2004. IEEE.

[BMT02a] Marco Baroni, Johannes Matiasek, and Harald Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57. Association for Computational Linguistics, July 2002.

[BMT02b] Marco Baroni, Johannes Matiasek, and Harald Trost. Wordform-and class-based prediction of the components of german nominal compounds in an aac system. In *Proceedings of the 19th international conference*

on Computational linguistics-Volume 1, pages 1–7. Association for Computational Linguistics, 2002.

[BN18] Jacob Buckman and Graham Neubig. Neural lattice language models. *Transactions of the Association for Computational Linguistics*, 6:529–541, 2018.

[BNB+13] Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428, 2013.

[BNG18] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis. Random walks with restarts for graph-based classification: Teleportation tuning and sampling design. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2811–2815, April 2018.

[Boh10] Bernd Bohnet. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[BS90] Akshar Bharati and Rajeev Sangal. A karaka based approach to parsing of Indian languages. In *COLNG 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*, 1990.

[BS93] Akshar Bharati and Rajeev Sangal. Parsing free word order languages in the paninian framework. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 105–111, Columbus, Ohio, USA, June 1993. Association for Computational Linguistics.

[BS98] Ken Barker and Stan Szpakowicz. Semi-automatic recognition of noun modifier relationships. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 96–102, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics.

[BWE+11] Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France, September 2011. Association for Computational Linguistics.

[BYM17] David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 429–439. JMLR. org, 2017.

[Car65] George Cardona. On translating and formalising pāṇinian rules. *Journal of the Oriental Institute of Baroda*, 14:306–14, 1965.

[Car07] Xavier Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 957–961, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[Car09] George Cardona. On the structure of pāini's system. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 1–32, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[CL65] Yoeng-Jin Chu and T H Liu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

[CL92] Keh-Jiann Chen and Shing-Huan Liu. Word identification for mandarin chinese sentences. In *Proceedings of the 14th conference on Computational linguistics-Volume 1*, pages 101–107. Association for Computational Linguistics, 1992.

[Cou92] Michael Coulson. Sanskrit: An introduction to the classical language. 1992.

[CP97] BB Chaudhuri and U Pal. An ocr system to read two indian language scripts: Bangla and devnagari (hindi). In *Proceedings of the Fourth International*

Conference on Document Analysis and Recognition, volume 2, pages 1011–1015, Ulm, Germany, 1997. IEEE.

[CS07] Shay B. Cohen and Noah A. Smith. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 208–217, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[CS17] Ryan Cotterell and Hinrich Schütze. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics*, 2017.

[CSS13] Himani Chaudhry, Himanshu Sharma, and Dipti Misra Sharma. Divergences in English-Hindi parallel dependency treebanks. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 33–40, Prague, Czech Republic, August 2013. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

[CWB+11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[CYJ+14] Guang Chen, Jianchao Yang, Hailin Jin, Jonathan Brandt, Eli Shechtman, Aseem Agarwala, and Tony X Han. Large-scale visual font recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3598–3605, 2014.

[Das16] Monali Das. discourse analysis of sanskrit texts: first attempt towards computational processing. 2016.

[DBSR18] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[DE11] Markus Dreyer and Jason Eisner. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 616–627, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[Den05] Simon Dennis. A memory-based theory of verbal cognition. *Cognitive Science*, 29(2):145–193, 2005.

[Deo07] Ashwini Deo. Derivational morphology in inheritance-based lexica: Insights from pāini. *Lingua*, 117(1):175–201, 2007.

[DFT14] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Hc-search: a learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, 50(1):369–407, 2014.

[DGG16] Eva D'hondt, Cyril Grouin, and Brigitte Grau. Low-resource ocr error detection and correction in french clinical texts. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 61–68, Auxtin, TX, November 2016. Association for Computational Linguistics.

[DH15] Corina Dima and Erhard Hinrichs. Automatic noun compound interpretation using deep neural networks and word embeddings. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 173–183, London, UK, April 2015. Association for Computational Linguistics.

[DM17] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Dow77] Pamela Downing. On the creation and use of english compound nouns. *Language*, 53(4):810–842, 1977.

[Edm67] Jack Edmonds. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1967.

[EHJ+04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[Eis02] Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[EOA+18] Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364. Association for Computational Linguistics, 2018.

[Fin80] Timothy Wilking Finin. *The Semantic Interpretation of Compound Nominals*. PhD thesis, Champaign, IL, USA, 1980. AAI8026491.

[FMS16] Manaal Faruqui, Ryan McDonald, and Radu Soricut. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16, 2016.

[FOV18] Murhaf Fares, Stephan Oepen, and Erik Velldal. Transfer and multi-task learning for noun–noun compound interpretation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[GAG+17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*, 2017.

[GE10] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, June 2010. Association for Computational Linguistics.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[GFP+19] Edward Gibson, Richard Futrell, Steven T. Piandadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389 – 407, 2019.

[GGJ06] Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics, 2006.

[GH13] Pawan Goyal and Gérard Huet. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd*, pages 130–171, 2013.

[GH16] Pawan Goyal and Gerard Huet. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182, 2016.

[GHDL18] Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[GHK+12] Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. A distributed platform for Sanskrit processing. In *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[Gib98] Edward Gibson. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68(1):1 – 76, 1998.

[Gil91] Brendan Gillon. Sanskrit word formation and context free rules. *Toronto Working Papers in Linguistics*, 11, 1991.

[Gil09] Brendan S. Gillon. Tagging classical sanskrit compounds. In *Proceedings of the 3rd International Symposium on Sanskrit Computational Linguistics*, pages 98–105, Hyderabad, India, 2009. Springer-Verlag.

[GJ04] Sharon Goldwater and Mark Johnson. Priors in Bayesian learning of phonological rules. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 35–42, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[GLLL16] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics.

[GM15] Matt Gardner and Tom Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[GM18] Thomas Graf and Connor Mayer. Sanskrit n-retroflexion is input-output tier-based strictly local. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 151–160, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[Gol01] John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

[GPB+13] Edward Gibson, Steven T Piantadosi, Kimberly Brink, Leon Bergen, Eunice Lim, and Rebecca Saxe. A noisy-channel account of crosslinguistic word-order variation. *Psychological science*, 24(7):1079–1088, 2013.

[GPMS14] Sai Kiran Gorthi, Ashish Palakurthi, Radhika Mamidi, and Dipti Misra Sharma. Identification of karaka relations in an English sentence. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 146–149, Goa, India, December 2014. NLP Association of India.

[GS90] VK Govindan and AP Shivaprasad. Character recognition—a review. *Pattern recognition*, 23(7):671–683, 1990.

[GS09] Venu Govindaraju and Srirangaraj Setlur. *Guide to OCR for Indic Scripts*. Springer, 2009.

[GS10] Romaric Gaudel and Michele Sebag. Feature selection as a one-player game. In *International Conference on Machine Learning*, pages 359–366, 2010.

[GT08] Yoav Goldberg and Reut Tsarfaty. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[GTM15] Matt Gardner, Partha Talukdar, and Tom Mitchell. Combining vector space embeddings with symbolic logical inference over open-domain text. In *2015 AAAI spring symposium series*, volume 6, page 1, Palo Alto, California, 2015.

[GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[Hal73] Morris Halle. Prolegomena to a theory of word formation. *Linguistic inquiry*, 4(1):3–16, 1973.

[Hav07] Jiri Havelka. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[HB14] Matic Horvat and William Byrne. A graph-based approach to string regeneration. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 85–95, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[Hel09a] Oliver Hellwig. Extracting dependency trees from sanskrit texts. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics*, pages 106–115, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Hel09b] Oliver Hellwig. Sanskrittagger: A stochastic lexical and POS tagger for sanskrit. In *Sanskrit Computational Linguistics*, pages 266–277. Springer, 2009.

[Hel15a] Oliver Hellwig. ind. senz–ocr software for hindi, marathi, tamil, and sanskrit, 2015.

[Hel15b] Oliver Hellwig. Using recurrent neural networks for joint compound splitting and sandhi resolution in sanskrit. In *4th Biennial Workshop on Less-Resourced Languages*, 2015.

[Hel16a] Oliver Hellwig. *DCS - The Digital Corpus of Sanskrit*. Berlin, 2010-2016.

[Hel16b] Oliver Hellwig. Detecting sentence boundaries in sanskrit texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 288–297. The COLING 2016 Organizing Committee, 2016.

[Hel16c] Oliver Hellwig. Improving the morphological analysis of classical Sanskrit. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 142–151, 2016.

[Hir01] H Hirakawa. Semantic dependency analysis method for japanese based on optimum tree search algorithm. *Proceedings of the PACLING 2001*, 2001.

[HK14] Gérard Huet and Amba Kulkarni. Sanskrit linguistics web services. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 48–51,

Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[HMMT12] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[HN18] Oliver Hellwig and Sebastian Nehrdich. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763. Association for Computational Linguistics, 2018.

[HNvG17] Georg Heigold, Guenter Neumann, and Josef van Genabith. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 505–513, 2017.

[Hoc15] Hans Henrich Hock. Some issues in sanskrit syntax. *Sanskrit syntax*, pages 1–34, 2015.

[HR06] Gérard Huet and Benoît Razet. *The Reactive Engine for Modular Transducers*, pages 355–374. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[HST+17] Eva Hasler, Felix Stahlberg, Marcus Tomalin, Adrià de Gispert, and Bill Byrne. A comparison of neural models for word ordering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 208–212, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics.

[HTOT00] Diiek Z. Hakkani-Tur, Kemal Oflazer, and Gokhan Tur. Statistical morphological disambiguation for agglutinative languages. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[Hue03] Gérard Huet. Zen and the art of symbolic computing: Light and fast applicative algorithms for computational linguistics. In Veronica Dahl and Philip Wadler, editors, *Practical Aspects of Declarative Languages*, pages 17–18, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Hue09] Gérard Huet. Sanskrit Segmentation, South Asian Languages Analysis Roundtable xxviii, Denton, Texas. 2009.

[Hue15] Gérard Huet. Sanskrit signs and Pāṇinian scripts. *Sanskrit and the IT World, The 16th World Sanskrit Conference, Bangkok, Thailand*, pages 53–76, 2015.

[HWGL09] Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. Dependency based chinese sentence realization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 809–816, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[Hym09] Malcolm D. Hyman. Sanskrit computational linguistics. chapter From PāNinian Sandhi to Finite State Calculus, pages 253–265. Springer-Verlag, Berlin, Heidelberg, 2009.

[Ish11] Hiroshi Ishikawa. Transformation of general binary mrf minimization to the first-order case. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1234–1249, 2011.

[JG09] Mark Johnson and Sharon Goldwater. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[JGG06] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, 2006.

[JKS07] Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, April 2007. Association for Computational Linguistics.

[KB05] Su Nam Kim and Timothy Baldwin. Automatic interpretation of noun compounds using wordnet similarity. In *International Conference on Natural Language Processing*, pages 945–956. Springer, 2005.

[KB06] Su Nam Kim and Timothy Baldwin. Interpreting semantic relations in noun compounds via verb semantics. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 491–498, Sydney, Australia, July 2006. Association for Computational Linguistics.

[KB13] Su Nam Kim and Timothy Baldwin. Word sense and semantic relations in noun compounds. *ACM Transactions on Speech and Language Processing (TSLP)*, 10(3):9, 2013.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *In 3rd International Conference on Learning Representations (ICLR) 2015*, 2015.

[KBK+17] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain, April 2017. Association for Computational Linguistics.

[KCS16] Katharina Kann, Ryan Cotterell, and Hinrich Schütze. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967, Austin, Texas, November 2016. Association for Computational Linguistics.

[KDK+10] Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya. Introducing sanskrit wordnet. In *Proceedings on the 5th Global Wordnet Conference (GWC 2010), Narosa, Mumbai*, pages 287–294, 2010.

[KG15] Amrith Krishna and Pawan Goyal. Towards automating the generation of derivative nouns in sanskrit by simulating panini. *arXiv preprint arXiv:1512.05670*, 2015.

[KG16] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

[Kip94] Paul Kiparsky. Paninian linguistics. *The Encyclopedia of Language and Linguistics*, 6:2918–2923, 1994.

[Kip09] Paul Kiparsky. *On the Architecture of Pāini's Grammar*, pages 33–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Kip10] Paul Kiparsky. Dvandvas, blocking, and the associative: The bumpy ride from phrase to word. *Language*, 86(2):302–331, 2010.

[KJ97] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[KJ07] Anand Kumar and CV Jawahar. Content-level annotation of large collection of printed document images. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 799–803, Parana, Brazil, 2007. IEEE.

[KK81] Ronald M Kaplan and Martin Kay. Phonological rules and finite-state transducers. In *Linguistic Society of America Meeting Handbook, Fifty-Sixth Annual Meeting*, pages 27–30, 1981.

[KK94] Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20,3:331–378, 1994.

[KK11] Amba Kulkarni and Anil Kumar. Statistical constituency parser for sanskrit compounds. In *Proceedings of Ninth International Conference on Natural Language Processing*, Anna University, Chennai, Dec 2011.

[KK13] Amba Kulkarni and Anil Kumar. Clues from aṣṭādhyāyī for compound type identification. In *5th International Sanskrit Computational Linguistics Symposium (SCLS)*, 2013.

[KMN09] Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009.

[Knu92] Donald E Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992.

[KPS10] Amba Kulkarni, Sheetal Pokar, and Devanand Shukl. Designing a constraint based parser for sanskrit. In *International Sanskrit Computational Linguistics Symposium*, pages 70–90. Springer, 2010.

[KR13] Amba Kulkarni and KV Ramakrishnamacharyulu. Parsing sanskrit texts: Some relation specific issues. In *Proceedings of the 5th International Sanskrit Computational Linguistics Symposium. DK Printworld (P) Ltd*, 2013.

[KRC+10] Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October 2010. Association for Computational Linguistics.

[KS09] Amba Kulkarni and Devanand Shukl. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177, 2009. in the Festscrift volume of Bh. Krishnamoorty.

[KSB+18] Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, and Pawan Goyal. Free as in free word order: An energy based model for word segmentation and morphological tagging in sanskrit. In *Proceedings of the 2018 Conference*

on Empirical Methods in Natural Language Processing, pages 2550–2561. Association for Computational Linguistics, 2018.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[KSG17] Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. A dataset for sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[KSK09] Anil Kumar, V Sheebasudheer, and Amba Kulkarni. Sanskrit compound paraphrase generator. In *Proceedings of Seventh International Conference on Natural Language Processing (ICON)*, IIIT-H and University of Hyderabad, Dec 2009.

[KSP+17] Amrith Krishna, Pavankumar Satuluri, Harshavardhan Ponnada, Muneeb Ahmed, Gulab Arora, Kaustubh Hiware, and Pawan Goyal. 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 2017. Association for Computational Linguistics.

[KSS+16a] Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 494–504, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[KSS+16b] Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 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 2016. The COLING 2016 Organizing Committee.

[KSS+19] Amrith Krishna, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, and Pawan Goyal. Poetry to prose conversion in Sanskrit as a linearisation task: A case for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1160–1166, Florence, Italy, July 2019. Association for Computational Linguistics.

[KSSJ14] Praveen Krishnan, Naveen Sankaran, Ajeet Kumar Singh, and CV Jawahar. Towards a robust ocr system for indic scripts. In *Eleventh IAPR International Workshop on Document Analysis Systems (DAS)*, pages 141–145, Tours, France, 2014. IEEE.

[KSSS15] Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. How Free is 'free' Word Order in Sanskrit. In *The Sanskrit Library, USA*, pages 269–304, 2015.

[Kud06] Taku Kudo. Mecab: Yet another part-of-speech and morphological analyzer. Retrieved october 9, 2019 from https://taku910.github.io/mecab/, 2006.

[Kud18] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Kul13] Amba Kulkarni. A deterministic dependency parser with dynamic programming for sanskrit. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 157–166, 2013.

[Kum12] Anil Kumar. *An automatic Sanskrit Compound Processing*. PhD thesis, University of Hyderabad, Hyderabad, 2012.

[KWC18] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the*

2018 Conference on Empirical Methods in Natural Language Processing, pages 1466–1477, Brussels, Belgium, 2018. Association for Computational Linguistics.

[KYM04] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[Lap03] Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[Lap06] Mirella Lapata. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484, 2006.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBS+16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics.

[LC10] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67, October 2010.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[LCL15] Jey Han Lau, Alexander Clark, and Shalom Lappin. Unsupervised prediction of acceptability judgements. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1618–1628, Beijing, China, July 2015. Association for Computational Linguistics.

[LCRH07] Yann LeCun, Sumit Chopra, Marc'Aurelio Ranzato, and Fu-Jie Huang. Energy-based models in document recognition and computer vision. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pages 337–341, Curitiba, Paraná, Brazil, 2007. IEEE Computer Society.

[Lev89] Judith N Levi. *The syntax and semantics of complex nominals*. Academic Press New York, 19789.

[Lie09] Rochelle Lieber. A lexical semantic approach to compounding. In *The Oxford handbook of compounding*. 2009.

[Lig96] Marc Light. Morphological cues for lexical semantics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 25–31, Santa Cruz, California, USA, June 1996. Association for Computational Linguistics.

[LLR18] Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. Sentence ordering using recurrent neural networks. In *Proceedings of the 32nd National Conference on Artificial Intelligence*, AAAI'18, ew Orleans, Louisiana, USA, 2018. AAAI Press.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[LMZB13] Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. Compositional-ly derived representations of morphologically complex

words in distributional semantics. In *ACL (1)*, pages 1517–1526. Citeseer, 2013.

[Low15] John J Lowe. The syntax of sanskrit compounds. *Language*, 91(3):e71–e115, 2015.

[LPM15] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[LWZ09] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[LZ15] Jiangming Liu and Yue Zhang. An empirical comparison between n-gram and syntactic language models for word ordering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 369–378, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[LZCQ15] Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. Transition-based syntactic linearization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 113–122, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[Man15] Christopher D. Manning. Last words: Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707, December 2015.

[Mar69] Hans Marchand. *The categories and types of present-day English word-formation: A synchronic-diachronic approach*. Beck, 1969.

[MB14] Andreas C Müller and Sven Behnke. Pystruct: learning structured prediction in python. *Journal of Machine Learning Research*, 15(1):2055–2060, 2014.

[MBT+] Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. Models for the semantic classification of noun phrases. In *Proceedings of the Computational Lexical Semantics Workshop at HLT-NAACL 2004*.

[MCM+15] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 754–764, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[Mel88] Igor A Mel'čuk. *Dependency syntax: theory and practice*. SUNY press, 1988.

[MGN18] Chaitanya Malaviya, Matthew R. Gormley, and Graham Neubig. Neural factor graph models for cross-lingual morphological tagging. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2653–2663, Melbourne, Australia, 2018. Association for Computational Linguistics.

[MGS13] Keshav S Melnad, Pawan Goyal, and Peter Scharf. Meter identification of sanskrit verse. *The Sanskrit Library, USA*, 2013.

[Mit10] Vipul Mittal. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90. Association for Computational Linguistics, 2010.

[ML10] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[ML11] Wolfgang Maier and Timm Lichte. Characterizing discontinuity in constituent treebanks. In Philippe de Groote, Markus Egg, and Laura Kallmeyer, editors, *Formal Grammar*, pages 167–182, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[MMG04] Andrew Cameron Morris, Viktoria Maier, and Phil Green. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*, 2004.

[Moh97] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[Mor16] Amir More. *Joint Morpho-Syntactic Processing of Morphologically Rich Languages in a Transition-Based Framework*. PhD thesis, MSc thesis, The Interdisciplinary Center, Herzliya, 2016.

[MPK+05] Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 491–498, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[MPRH05] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[MS07] Ryan McDonald and Giorgio Satta. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[MS18] Andrew McLeod and Mark Steedman. Meter detection and alignment of midi performance. In *ISMIR*, pages 113–119, 2018.

[MSBT19] Amir More, Amit Seker, Victoria Basmova, and Reut Tsarfaty. Joint transition-based models for morpho-syntactic parsing: Parsing strategies

for MRLs and a case study from modern Hebrew. *Transactions of the Association for Computational Linguistics*, 7:33–48, March 2019.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., Stateline, Nevada, United States, 2013.

[MSJ16] Minesh Mathew, Ajeet Kumar Singh, and CV Jawahar. Multilingual ocr for indic scripts. In *Document Analysis Systems (DAS), 2016 12th IAPR Workshop on*, pages 186–191. IEEE, 2016.

[MSS13] Thomas Mueller, Helmut Schmid, and Hinrich Schütze. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[MT16] Amir More and Reut Tsarfaty. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 337–348, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[Muñ13] Molina Muñoz. *Sanskrit compounds and the architecture of the grammar*, pages 33–94. Beech Stave Press, Ann Arbor/New York, 2013.

[MW99] Monier Monier-Williams. A sanskrit-english dictionary, 1899.

[NBJ15] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167, 2015.

[NC11] Abhiram Natarajan and Eugene Charniak. S3-statistical saṃdhi splitting. In *Proceedings of the 5th International Joint Conference on Natural Language*

Processing, pages 301–308. Association for Computational Linguistics, 2011.

[NK10] Sivaja S Nair and Amba Kulkarni. The knowledge structure in amarakośa. In *Sanskrit Computational Linguistics*, pages 173–189. Springer, 2010.

[NSSSS06] Vivi Nastase, Jelber Sayyad-Shirabad, Marina Sokolova, and Stan Szpakowicz. Learning noun-modifier semantic relations with corpus-based and wordnet-based features. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 781. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[O'D15] Timothy J O'Donnell. *Productivity and reuse in language: A theory of linguistic computation and storage*. MIT Press, 2015.

[ON93] M. Oerder and H. Ney. Word graphs: an efficient interface between continuous-speech recognition and language understanding. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 119–122, April 1993.

[Pav16] Satuluri Pavankumar. *Sanskrit Compound Generation: With a Focuson the Order of Operations*. PhD thesis, University of Hyderabad, Hyderabad, 2016.

[Pei97] Charles S. Peirce. The Logic of Relatives. *The Monist*, 7(2):161–217, 01 1897.

[Pet05] Wiebke Petersen. How formal concept lattices solve a problem of ancient linguistics. In *International Conference on Conceptual Structures*, pages 337–352. Springer, 2005.

[PK12] Gerald Penn and Paul Kiparsky. On Panini and the generative capacity of contextualized replacement systems. In *Proceedings of COLING 2012: Posters*, pages 943–950, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings*

of 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Prz18] Adam Przepiórkowski. The origin of the valency metaphor in linguistics. *Lingvisticæ Investigationes*, 41(1):152–159, 2018.

[PS92] Alan Prince and Paul Smolensky. Optimality: Constraint interaction in generative grammar. In *12th West Coast Conference on Formal Linguistics, Los Angeles*, 1992.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PWK09] Umapada Pal, Tetsushi Wakabayashi, and Fumitaka Kimura. Comparative study of devnagari handwritten character recognition using different feature and classifiers. In *Tenth International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE, 2009.

[PXS18] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018.

[PZS17] Ratish Puduppully, Yue Zhang, and Manish Shrivastava. Transition-based deep input linearization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 643–654, Valencia, Spain, April 2017. Association for Computational Linguistics.

[Raj] Shreevatsa Rajagopalan. A user-friendly tool for metrical analysis of sanskrit verse. *Computational Sanskrit & Digital Humanities, 17th World Sanskrit Conference*, pages 113–142.

[Ram09] KV Ramkrishnamacharyulu. Annotating sanskrit texts based on śābdabodha systems. In *International Sanskrit Computational Linguistics Symposium*, pages 26–39. Springer, 2009.

[RBZ07] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. (online) subgradient methods for structured prediction. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, pages 380–387, San Juan, Puerto Rico, March 2007. JMLR.org.

[RH01] Barbara Rosario and Marti Hearst. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 2001.

[RKLS07] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary mrfs via extended roof duality. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8, Minneapolis, Minnesota, USA, 2007. IEEE Computer Society.

[RKS+18] Vikas Reddy, Amrith Krishna, Vishnu Sharma, Prateek Gupta, Vineeth M R, and Pawan Goyal. Building a Word Segmenter for Sanskrit Overnight. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA).

[SAC+17] Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. Error detection and corrections in indic ocr using lstms. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 17–22, Kyoto, Japan, 2017. IEEE.

[SAL09] Ray Smith, Daria Antonova, and Dar-Shyang Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, page 1. ACM, 2009.

[Sap04] Edward Sapir. Language: An introduction to the study of speech, 2004.

[SASG15] Peter Scharf, Anuja Ajotikar, Sampada Savardekar, and Pawan Goyal. Distinctive features of poetic syntax preliminary results. *Sanskrit syntax*, pages 305–324, 2015.

[SB08] Benjamin Snyder and Regina Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[SB19] Rico Sennrich and Zhang Biao. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019. Association for Computational Linguistics.

[SC13] Diarmuid O Séaghdha and Ann Copestake. Interpreting compound nouns with kernel methods. *Natural Language Engineering*, 19(03):331–356, 2013.

[SÇ15] Wolfgang Seeker and Özlem Çetinoğlu. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373, 2015.

[Sch05] Anne Schiller. German compound analysis with wfsc. In *International Workshop on Finite-State Methods and Natural Language Processing*, pages 239–246. Springer, 2005.

[Séa09] Diarmuid Séaghdha. Semantic classification with wordnet kernels. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 237–240. Association for Computational Linguistics, 2009.

[SEDDG16] Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, 2016.

[SG16] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany, 2016. Association for Computational Linguistics.

[Sha87] Ram Nath Sharma. *The Astadhyayi of Panini, volume I–VI*. Munshiram Manoharlal Publishers Pvt. Ltd, 1987.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SHN18] Yan Shao, Christian Hardmeier, and Joakim Nivre. Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435, 2018.

[SJ00] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000.

[SJ12a] Naveen Sankaran and CV Jawahar. Recognition of printed devanagari text using blstm neural network. In *21st International Conference on Pattern Recognition (ICPR)*, pages 322–325, Tsukuba Science City, JAPAN, 2012. IEEE.

[SJ12b] Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. In *Proceedings of the 29th International Coference*

on International Conference on Machine Learning, ICML'12, pages 59–66, Edinburgh, Scotland, 2012. Omnipress.

[SJ15] Ajeet Kumar Singh and CV Jawahar. Can rnns reliably separate script and language at word and line level? In *13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 976–980. IEEE, 2015.

[SK13] Wolfgang Seeker and Jonas Kuhn. Morphological and syntactic case in statistical dependency parsing. *Computational Linguistics*, 39(1):23–55, 2013.

[SKS+16] Preeti Shukla, Amba Kulkarni, Devanand Shukl, Maharshi Sandipani, and Rashtriya Vedavidya Pratisthan. Revival of ancient sanskrit teaching methods using computational platforms. In *Bridging the gap between Sanskrit Computational Linguistics tools and management of Sanskrit Digital Libraries, ICON*, IIT BHU, 2016.

[SLZ+17] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2017.

[Smi87] Raymond W Smith. *The Extraction and Recognition of Text from Multimedia Document Images*. PhD thesis, University of Bristol, 1987.

[Smi07] Ray Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition, (ICDAR)*, volume 2, pages 629–633. IEEE, 2007.

[SMN10] Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

[SMR07] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723, 2007.

[SMT18] Amit Seker, Amir More, and Reut Tsarfaty. Universal morpho-syntactic parsing and the contribution of lexica: Analyzing the ONLP lab submission to the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 208–215, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[SN12] M. Schuster and K. Nakajima. Japanese and korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, Kyoto, Japan, March 2012.

[SO15] Radu Soricut and Franz Och. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[SP06] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1419–1424, Boston, Massachusetts, 2006. AAAI Press.

[SPS08] Bikash Shaw, Swapan Kumar Parui, and Malayappan Shridhar. Offline handwritten devanagari word recognition: A holistic approach based on directional chain code feature and hmm. In *International Conference on Information Technology (ICIT)*, pages 203–208, Bhubaneswar, India, 2008. IEEE.

[SRB06] Manish Sinha, Mahesh Reddy, and Pushpak Bhattacharyya. An approach towards construction and application of multilingual indo-wordnet. In *3rd Global Wordnet Conference (GWC 06), Jeju Island, Korea*, 2006.

[Sri17] Vivek Srikumar. An algebra for feature extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1891–1900, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[SRS16] Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. Word ordering without syntax. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2319–2324, Austin, Texas, November 2016. Association for Computational Linguistics.

[SS18] Gozde Gul Sahin and Mark Steedman. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[SSGC94] Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. A stochastic finite-state word-segmentation algorithm for chinese. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 66–73, Las Cruces, New Mexico, USA, June 1994. Association for Computational Linguistics.

[SSGC96] Richard W. Sproat, Chilin Shih, William Gale, and Nancy Chang. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404, 1996.

[SST05] Noah A. Smith, David A. Smith, and Roy W. Tromble. Context-based morphological disambiguation with random fields. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 475–482, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[ST03] Stuart M Shieber and Xiaopeng Tao. Comma restoration using constituency information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 142–148. Association for Computational Linguistics, 2003.

[Sta67] Johan Frederik Staal. *Word order in Sanskrit and universal grammar*, volume 5. Springer Science & Business Media, 1967.

[Sta08] Frits Staal. *Discovering the Vedas : origins, mantras, rituals, insights*. Penguin Books India, 2008.

[Sun10] Weiwei Sun. Word-based and character-based word segmentation models: Comparison and combination. In *International Conference on Computational Linguistics (Coling) 2010: Posters*, pages 1211–1219, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., Montreal, Canada, 2014.

[SW18] Vered Shwartz and Chris Waterson. Olive oil is made *of* olives, baby oil is made *for* babies: Interpreting noun compounds using paraphrases in a neural model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 218–224, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Tak80] Hiromitsu Takahashi. An approximate solution for steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[TC09] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Proceedings of the 2009th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECMLPKDD'09, pages 442–457, Bled, Slovenia, 2009. Springer-Verlag.

[TGK03] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 25–32, Cambridge, MA, USA, 2003. MIT Press.

[TH10] Stephen Tratz and Eduard Hovy. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual*

Meeting of the Association for Computational Linguistics, pages 678–687, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[Tie05] Jörg Tiedemann. Improving passage retrieval in question answering using nlp. In *Portuguese Conference on Artificial Intelligence*, pages 634–646. Springer, 2005.

[TS18] Alexander Tkachenko and Kairit Sirts. Modeling composite labels for neural morphological tagging. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 368–379, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[Tsa06] Reut Tsarfaty. Integrated morphological and syntactic disambiguation for modern hebrew. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 49–54, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[Tsv16] Yulia Tsvetkov. *Linguistic Knowledge in Data-Driven Natural Language Processing*. PhD thesis, PhD thesis, Carnegie Mellon University, 2016.

[TTT06] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

[UHB13] Adnan Ul-Hasan and Thomas M. Breuel. Can we build language-independent ocr using lstm networks? In *Proceedings of the 4th International Workshop on Multilingual OCR*, MOCR '13, pages 9:1–9:5, New York, NY, USA, 2013. ACM.

[Var96] A.R.R. Varma. *Keralapanineeyam*. DC Books (9th Edition reprint), 1896.

[VBK16] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *Proceedings of the Fourth International Conference on Learning Representation (ICLR)*, Vancouver, BC, Canada, April 2016.

[VDVZVH14] Ben Verhoeven, Walter Daelemans, Menno Van Zaanen, and Gerhard Van Huyssteen. Automatic compound processing: Compound splitting and semantic analysis for afrikaans and dutch. *ComAComA 2014*, page 20, 2014.

[VK08] David Vickrey and Daphne Koller. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Vos93] Stefan Voss. Worst-case performance of some heuristics for steiner's problem in directed graphs. *Information Processing Letters*, 48(2):99–105, 1993.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., California, United States, 2017.

[WCM+14] Marion Weller, Fabienne Cap, Stefan Müller, Sabine Schulte im Walde, and Alexander Fraser. Distinguishing degrees of compositionality in compound splitting for statistical machine translation. In *Proceedings of the First Workshop on Computational Approaches to Compound Analysis (ComAComA 2014)*, pages 81–90, 2014.

[WCM18] Wenhui Wang, Baobao Chang, and Mairgup Mansur. Improved dependency parsing using implicit word connections learned from unlabeled data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2857–2863. Association for Computational Linguistics, 2018.

[Whi89] William Dwight Whitney. *Sanskrit grammar: Including both the classical language, and the older dialects, of Veda and Brahmana*, volume 2. Harvard University Press, 1889.

[Wic04] Richard Wicentowski. Multilingual noise-robust supervised morphological analysis using the WordFrame model. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current*

Themes in Computational Phonology and Morphology, pages 70–77, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[WR16] Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas, November 2016. Association for Computational Linguistics.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144. Retrieved from https://arxiv.org/abs/1609.08144, 2016.

[WVM14] Mengqiu Wang, Rob Voigt, and Christopher D. Manning. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 193–198, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[WW77] J. Wolf and W. Woods. The hwim speech understanding system. In *ICASSP '77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 784–787, May 1977.

[WWRK16] David Weir, Julie Weeds, Jeremy Reffin, and Thomas Kober. Aligning packed dependency trees: A theory of composition for distributional semantics. *Computational Linguistics*, 42(4):727–761, 2016.

[Xue03] Nianwen Xue. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48, 2003.

[Yu18] Lei Yu. *Tackling Sequence to Sequence Mapping Problems with Neural Networks*. PhD thesis, Mansfield College, University of Oxford, 2018.

[YW00] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 207–216, Hong Kong, October 2000. Association for Computational Linguistics.

[YZL19] Jie Yang, Yue Zhang, and Shuailong Liang. Subword encoding in lattice LSTM for Chinese word segmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2720–2725, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[ZBGC14] Ke Zhai, Jordan Boyd-Graber, and Shay B. Cohen. Online adaptor grammars with hybrid inference. *Transactions of the Association for Computational Linguistics*, 2:465–476, 2014.

[ZBL+] Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 321–328. Vancouver, Canada.

[ZC11] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151, 2011.

[ZC15] Yue Zhang and Stephen Clark. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538, 2015.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD, 2002.

[Zha13] Yue Zhang. Partial-tree linearization: Generalized word ordering for text synthesis. In *Proceedings of the Twenty-Third International Joint*

Conference on Artificial Intelligence, IJCAI '13, pages 2232–2238, Beijing, China, 2013. AAAI Press.

[ZLP+17] Boliang Zhang, Di Lu, Xiaoman Pan, Ying Lin, Halidanmu Abudukelimu, Heng Ji, and Kevin Knight. Embracing non-traditional linguistic resources for low-resource language name tagging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 362–372, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[ZN11] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[ZNR19] Zining Zhu, Jekaterina Novikova, and Frank Rudzicz. Detecting cognitive impairments by agreeing on interpretations of linguistic features. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1431–1441, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[ZvdP16] Patrick Ziering and Lonneke van der Plas. Towards unsupervised and language-independent compound splitting using inflectional morphological transformations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–653, San Diego, California, June 2016. Association for Computational Linguistics.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc., Montreal, Canada, 2015.

All Publications

The following is a list of publications during my tenure as a student in IIT Kharagpur. The publications (including communicated) are listed in chronological order.

1. **Amrith Krishna**, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, Pawan Goyal. 2019. A Structured Prediction Framework Using Energy Based Models for Sanskrit. Computational Linguistics, MIT Press (Communicated)

2. **Amrith Krishna**, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, Pawan Goyal. 2019. Poetry to Prose Conversion in Sanskrit as a Linearisation Task: A case for Low-Resource Languages. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers, ACL 2019). Florence, Italy 1160–1166.

3. **Amrith Krishna**, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, Pawan Goyal. 2018. Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018). Brussels, Belgium, 2550-2561.

4. **Amrith Krishna**, Bodhisattwa P. Majumder, Rajesh Bhat, Pawan Goyal. 2018. Upcycle Your OCR: Reusing OCRs for Post-OCR Text Correction in Romanised Sanskrit. In Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018). Brussels, Belgium, 345-355.

5. Vikas Reddy[*], **Amrith Krishna**[*], Vishnu Sharma, Prateek Gupta, M R Vineeth, Pawan Goyal. 2018. Building a Word Segmenter for Sanskrit Overnight. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Languages Resources Association (ELRA). Miyazaki, Japan.

6. **Amrith Krishna**, Bodhisattwa P. Majumder, Anilkumar Boga, Pawan Goyal. 2018. An 'Ekalavya' Approach to Learning Context Free Grammar Rules for Sanskrit Using Adaptor Grammar. In Proceedings of the 17th World Sanskrit Conference (WSC 2018). D K Publishers Distributors Pvt. Ltd. Vancouver, BC, Canada, 83-112.

[*]The first two authors contributed equally

7. **Amrith Krishna**, Pavankumar Satuluri, Pawan Goyal. 2017. A Dataset for Sanskrit Word Segmentation. In Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature Workshop, 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017). Vancouver, Canada, 105-114.

8. **Amrith Krishna**, Pavankumar Satuluri, Harshavardhan Ponnada, Muneeb Ahmed, Gulab Arora, Kaustubh Hiware, 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 Workshop, 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017). Vancouver, Canada, 66-75

9. **Amrith Krishna**, Pavankumar Satuluri , Shubham Sharma, Apurv Kumar, Pawan Goyal . 2016. Compound Type Identication in Sanskrit: What Roles do the Corpus and Grammar Play? In Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), The 26th International Conference on Computational Linguistics. Osaka, Japan, 1-10.

10. **Amrith Krishna**, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu , Yajuvendra Singh, Pawan Goyal. 2016. Word Segmentation in Sanskrit Using Path Constrained Random Walks. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan, 494-504.

11. **Krishna, Amrith**; Mallick, Madhumita; Mitra, Bivas. 2016. SleepSensei - An automated sleep quality monitor and sleep duration estimator. In Proceedings of the First Workshop on IoT-enabled Healthcare and Wellness Technologies and Systems, The 14th ACM International Conference on Mobile Systems, Applications, and Services. Singapore, 29-34

12. Chakraborty, Tanmoy; **Krishna, Amrith**; Singh, Mayank; Ganguly, Niloy; Goyal, Pawan and Mukherjee, Animesh. 2016. FeRoSA: A Faceted Recommendation System for Scientific Articles. In Proceedings of The 20th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD). Springer International Publishing. Auckland, New Zealand, 528-541.

13. **Krishna, Amrith** and Goyal, Pawan. 2015. Towards automating the generation of derivative nouns in Sanskrit by simulating Panini. In Computational Sanskrit and Digital Humanities, 16th World Sanskrit conference, Bangkok, Thailand.

14. **Krishna, Amrith**; Bhowmick, Plaban; Sahu, Archana; Ghosh, Krishnendu; Roy, Subhayan. 2015.Automatic Generation and Insertion of Assessment Items in Online Video Courses. In Proceedings of the 20th International Conference on Intelligent User Interfaces Companion. Atlanta, USA, 1-4.