

Sanskrit Segmentation and Morphological Analysis: Normalized Datasets and a Ranking-based approach

*A dissertation submitted during 2024 to the University of Hyderabad
in partial fulfillment of the award of the degree of*

Doctor of Philosophy

in

Sanskrit Studies

by

Sriram K

19HSPH02

under the guidance of

Prof. Amba Kulkarni

Department of Sanskrit Studies



Department of Sanskrit Studies

School of Humanities

University of Hyderabad

Hyderabad

2024



Certificate

This is to certify that the thesis entitled ***Sanskrit Segmentation and Morphological Analysis: Normalized Datasets and a Ranking-based approach*** submitted by **Sriram K** bearing registration number **19HSPH02** in partial fulfilment of the requirements for the award of **Doctor of Philosophy** in the **School of Humanities** is a bonafide work carried out by him under my supervision and guidance.

The thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Further, the student has the following publications before submission of the thesis for adjudication and has produced evidence for the same in the form of acceptance letter in the relevant area of research:

1. **Normalized Dataset for Sanskrit Word Segmentation and Morphological Parsing**, Language Resources and Evaluation, Springer Nature, August 2024, ISSN #1574-0218. (Chapters 3, 4 and 5 of this dissertation contain the contents predominantly from this article.)

<https://link.springer.com/article/10.1007/s10579-024-09724-0>

and has made presentations in the following conferences, the presented papers being published in the conference proceedings,

1. **Validation and Normalization of DCS corpus and Development of Sanskrit Heritage Engine's Segmenter**, Computational Sanskrit & Digital Humanities Section of the 18th World Sanskrit Conference, Association for Computational Linguistics, Canberra, Australia (Online mode), January 2023.

<https://aclanthology.org/2023.wsc-csdh.3>

2. **Sanskrit Segmentation Revisited**, 16th International Conference on Natural Language Processing, NLP Association of India, International Institute of Information Technology, Hyderabad, India, December 2019.

<https://aclanthology.org/2019.icon-1.12>

Further, the student has passed the following courses towards fulfillment of the coursework requirement for Ph.D.

No.	Course-Code	Course Title	Credits	Pass/Fail
1.	SK830	Dissertation Related Readings	4.00	Pass
2.	SK831	Dissertation Related Readings II	4.00	Pass
3.	SK801	Research Methodology	4.00	Pass

Amber P
13.12.24

Supervisor

Sanskrit Studies
University of Hyderabad
P.O. Central University
Hyderabad-500 046, TS

Amber P
13.12.24

Head of the Department

HEAD
Department of Sanskrit Studies
School of Humanities
University of Hyderabad
Central University P.O.
HYDERABAD-500 046

Amber P
13.12.24

Dean of School

I/c
DEAN
SCHOOL OF HUMANITIES
UNIVERSITY OF HYDERABAD
HYDERABAD, T.S. INDIA

Declaration

I, **Sriram K**, hereby declare that the work embodied in this dissertation entitled “**Sanskrit Segmentation and Morphological Analysis: Normalized Datasets and a Ranking-based approach**”, submitted by me under the supervision of **Prof. Amba Kulkarni**, Professor, Department of Sanskrit Studies, University of Hyderabad, is a bonafide research work and has not been submitted for any degree in part or in full to this university or any other university.

A report on plagiarism statistics from the University Librarian is enclosed. The justification for the similarity index > 10% approved and signed by the supervisor is also enclosed.



Sriram K

19HSPH02

Date: 13/12/2024

Place: Hyderabad



Signature of the Supervisor

Professor
Department of Sanskrit Studies
University of Hyderabad
P.O. Central University
Hyderabad-500 046, TS

Acknowledgments

I would like to express my deepest gratitude to all those who have supported and guided me throughout my journey in completing this thesis.

First and foremost, I am deeply indebted to my supervisor, *Prof. Amba Kulkarni*, whose belief in me and guidance introduced me to the field of Sanskrit Computational Linguistics. Her constant encouragement, mentorship, and patience have played a crucial role in shaping my research.

I am also extremely grateful to *Prof. Gérard Huet* for his constant guidance, valuable discussions, and insightful suggestions, which significantly contributed to the direction and depth of my dissertation. And also to his *Sanskrit Heritage Platform* which forms the basis of this research work.

My heartfelt thanks go to the members of my Research Advisory Committee, *Prof. Kavi Narayana Murthi* and *Prof. Parameshwari*, for their valuable feedback, critical directions, and support that enriched my research.

I would like to acknowledge *Prof. J S R Prasad* for his academic support and encouragement throughout this journey.

I would like to extend my sincere appreciation to *Dr. Pavan Kumar Satuluri*, *Dr. Arjuna S R*, *Dr. Sanjeev Panchal*, *Dr. Anil Kumar*, *Dr. Madhusudan*, *Dr. Shailaja* and *Dr. Gayatri* for their continuous guidance and advice on various aspects of my research.

I would like to sincerely thank *Dr. Oliver Hellwig* for his *Digital Corpus of Sanskrit* as it forms one of the basis of this research work. I also thank *Dr. Amrith Krishna* and *Dr. Jivnesh Sandhan* for providing valuable insights on various dimensions of the work including the discussions on the datasets and algorithms for segmentation and morphological analysis.

My fellow researchers, *Sanal Vikram*, *Malay Maity*, *Amruta Malvade*, *Sae Vaze*, *Amruta Barbadikar*, and *Anagha Pradeep*, have provided invaluable academic, intellectual, and emotional support throughout this journey, for which I am deeply grateful. *Prasanna Venkatesh*, my friend, has been a constant source of strength and guidance.

Our numerous discussions have not only shaped my research but also contributed greatly to my personal growth. I would also like to thank *Anil Kumar Sanmukh, Samar Kumar Parhi, Abirlal, and Deepak Garasangi* for their support and encouragement.

I would also like to extend my sincere gratitude to the *office staff of the Department of Sanskrit Studies, the IGM Library, and the Administration Office* for providing me with all sorts of infrastructural support and the scholarship.

A special thanks to *Dr. V Raghavendran and Sri S Ranganathan* for introducing me to the world of Sanskrit and nurturing my curiosity and understanding of it.

I am grateful to *Prof. N Kannan* for introducing me to the various dimensions of Sanskrit, particularly in the realms of mathematics and astronomy, which greatly broadened my perspective.

Sri Suprabatha Sabha deserves my heartfelt thanks for shaping my interest in Sanskrit, both as a language and a culture, which has been fundamental to my journey.

To my *parents, Malathi and Krishnan*, who have always supported me with love and trust, providing me the space and time to pursue my dreams, I owe my deepest gratitude. Their belief in me was a source of strength throughout this process.

I would also like to express my gratitude to my *brother, sisters, cousins, friends and relatives* for their emotional support during the course and beyond.

To my dear wife, *Vaishnavi*, who has been my pillar of strength, being patient, with her unwavering belief, and standing by me during the most difficult times, I am eternally thankful.

Lastly, to *Srirangan Srinivasan*, my best friend, whom I fondly call my *Ātmasakhā*, I dedicate this work. For always being there for me, understanding me like no one else, and accompanying me throughout this journey and beyond from wherever you are, I am truly grateful.

Finally, I would like to thank *God* for giving me everything.

Contents

Acknowledgments	i
List of Figures	vi
List of Tables	vii
Dissertation Related Articles	ix
1 Introduction	1
1.1 Motivation	3
1.2 Problem Statement	6
1.3 Existing approaches	7
1.4 Salient features of the current work	7
1.4.1 Differences between DCS and SH annotations	7
1.4.2 Normalized Dataset Generation from DCS and SH	8
1.4.3 Ranking SH segmentations	8
1.4.4 Joint task of Word Segmentation and Morphological Analysis using SH	8
1.4.5 Compound Boundary Indication	8
1.5 Organization of the Thesis	9
2 A Review on <i>Sandhi</i>, Datasets and Segmentation Approaches	10
2.1 Traditional Grammarian's perspective	12
2.1.1 Structure of a word	12
2.1.2 <i>Sandhi</i> and Segmentation (<i>sandhi-viccheda</i>)	13
2.1.3 Generation vs Analysis	16
2.2 Datasets and Annotations for Segmentation and Morphological Analysis	20
2.2.1 SHMT	21
2.2.2 Sandhi-Kosh	21
2.2.3 DCS - raw	22
2.2.4 DCS - segmentation	24
2.2.5 SIGHUM - A Dataset for Word Segmentation	25
2.3 Recent Implementations for Segmentation	26
2.3.1 Segmentation in Sanskrit	28
2.4 Sanskrit Heritage Segmenter (SH)	41
2.4.1 SH methodology	42
2.4.2 Observations on SH	47
2.5 Necessity for aligning DCS and SH analysis	49
2.5.1 SIGHUM Alignment	50
2.5.2 Why to align again?	52

3	Overcoming Linguistic Issues in Alignment	54
3.1	DCS-SH differences	55
3.1.1	Marking Sentence Boundaries	55
3.1.2	Chunk boundaries	57
3.1.3	Segments	58
3.1.4	Stem / Root	59
3.1.5	Morphological Analysis	63
3.1.6	Compounds	65
3.2	Aligning the Parameters	65
4	DCS-SH Alignment	68
4.1	STEP 1 - Extraction of data	69
4.1.1	Extracting DCS data and its representation	69
4.1.2	Extracting SH data and its representation	70
4.2	STEP 2 - Normalization	73
4.2.1	Aligning Chunks	73
4.2.2	Aligning Stems	74
4.2.3	Aligning Morphological Analysis	75
4.3	STEP 3 - Alignment	76
4.3.1	Alignment Results and Observations	77
4.4	STEP 4 - Alignment Issues and Modifications	78
4.5	Alignment observations	79
4.5.1	Causes of mismatches	82
5	Ranking Segmentations of Sanskrit Heritage Segmenter	86
5.1	SH Ranking	87
5.1.1	Ranking Criteria	87
5.1.2	Formulation of SH results	89
5.1.3	Preparing the data structure for the frequencies	92
5.1.4	Variations in ranking metrics	93
5.1.5	Ranking Algorithm	95
5.2	Evaluation	97
5.3	Observations	98
5.3.1	Sentence-level Evaluation	99
5.3.2	Word-level Evaluation	103
5.3.3	Performance on true unseen data	106
5.3.4	Evaluating the Ranking Algorithm	108
5.4	Representation of the Ranked Solutions	112
6	Conclusion	120
6.1	Key Contributions	120
6.1.1	New Alignment of DCS and SH - Normalized dataset	120
6.1.2	Improvement to the Heritage Segmenter	121
6.1.3	Integrating the updates with <i>Saṃsādhani</i> tools	123
6.2	Future Work	123
	Bibliography	126
A	DCS-SH Alignment	136

B	Resources	151
C	Word Segmentation and Morphological Analysis using SH	154

List of Figures

1.1	Possible Segmentations of the word <i>rāmālayaḥ</i> (word-forms and morphological analyses)	5
2.1	Sanskrit Heritage Segmenter’s Graphical Interface	41
2.2	The 10-phase lexical analyzer	43
2.3	SH Reader analysis of the sentence <i>rāmovanaṅgacchati</i> . Only the first two solutions are presented here for clarity.	46
3.1	Partial representation of DCS’ CoNLL-U annotation	56
4.1	Heritage Reader’s analysis of the compound <i>pātālabhāsuram</i>	72
5.1	Ranking results with the “First solution” mode (graphical interface) using the word metrics	113
5.2	Ranking results with the “Best <i>n</i> solutions” mode using the word metrics	114
5.3	Ranking results with the “Best <i>n</i> solutions” mode using the word metrics	115
5.4	Comparison of the ranking results with the “First” and the “Best <i>n</i> solutions” mode using the word metrics	115
5.5	Ranking results with the “First solution” mode using the morph metrics	116
5.6	Ranking results with the “Best <i>n</i> solutions” mode using the morph metrics	116
5.7	Ranking results with the “Best <i>n</i> solutions” mode using the morph metrics	117

List of Tables

1.1	Possible Segmentations the word <i>rāmālayaḥ</i> (word-forms)	4
2.1	DCS Segmentation Analysis	25
2.2	An example DCS Object data	50
3.1	SH Analysis for the word <i>siddham</i>	61
4.1	DCS data in JSON format for First Alignment	69
4.2	DCS data in JSON format for Second and Third Alignments	70
4.3	Reference for the fields in SH's analysis	73
4.4	Alignment Results Step 3	77
4.5	Comparison of Alignment 1 procedure over SIGHUM sentences and all DCS sentences	79
4.6	Alignment Comparison Steps 3 and 4	80
4.7	Overall Alignment Observations	80
4.8	Alignment 1 Results format in JSON	82
4.9	Alignment 2 and 3 Results format in JSON	83
5.1	An example of multiple morphological analyses	88
5.2	SH segmentation solutions for <i>rāmovanaṅgacchati</i>	88
5.3	Frequency lists obtained from the Alignment datasets	92
5.4	Performance Comparison of SH-Ranking with four datasets (SHMT, Alignment 1, SIGHUM sentences re-aligned with the new alignment algorithm, and SIGHUM sentences) on Bhagavad Gītā Chapter 3 verses Column 1 (Solution Rank) contains the rank of the correct solution. Column 2 contains the number of verses for which the correct solution was found in the rank (of column 1) when SHMT dataset was used to create the frequencies for the ranking algorithm. Similarly, column 3 contains the numbers when Alignment 1 dataset was used, column 4 contains the numbers when SIGHUM sentences re-aligned with the new alignment algorithm was used, and column 5 contains the numbers when the original SIGHUM dataset was used for the ranking. The overall accuracy (coverage) is also reported.	99
5.5	Sentence-level performance comparison of <i>Bhagavad Gītā</i> , <i>Meghadūta</i> , <i>Saṅkṣepa-Rāmāyaṇa</i> and rcNN-test-set on SH-Ranking and rcNN models	100
5.6	Sentence level performance comparison of SH-Ranking, rcNN and TransLIST on the test set of Hackathon Dataset (9,910 sentences)	103
5.7	Word-level evaluation of SH-Ranking and rcNN models on <i>Bhagavad Gītā</i> , <i>Meghadūta</i> , <i>Saṅkṣepa-Rāmāyaṇa</i> and rcNN-test-set	104
5.8	Word-level evaluation of SH-Ranking, rcNN and TransLIST models on the Hackathon dataset	105

5.9	Inter-Annotator Agreement between the three Annotators using Kappa Scores	107
5.10	Sentence-level evaluation of SH-Ranking and rcNN models on <i>Kathāsaritsāgara</i> (<i>lambaka 13</i>), where for the SH-Ranking metrics, only the sentences recognized by SH were considered, but for rcNN, additionally all the sentences were considered	107
5.11	Word-level evaluation of SH-Ranking and rcNN models on <i>Kathāsaritsāgara</i> (<i>lambaka 13</i>)	109
5.12	Compound evaluation of SH-Ranking metrics on <i>Bhagavad Gītā</i> , <i>Meghadūta</i> , <i>Śaṅkṣepa Rāmāyaṇa</i> and <i>Kathāsaritsāgara</i> (<i>lambaka 13</i>)	111
5.13	Ranking evaluation of SH-Ranking metrics on <i>Bhagavad Gītā</i> , <i>Meghadūta</i> , <i>Śaṅkṣepa-Rāmāyaṇa</i> and <i>Kathāsaritsāgara</i> (<i>lambaka 13</i>)	112
5.14	Comparison of chunk possibilities (for <i>Bhagavad Gītā</i> verse 1.1) across word, phase and morphological analysis	118
5.15	Comparison of overall possibilities (for <i>Bhagavad Gītā</i> verse 1.1) in the <i>first</i> and <i>best n</i> modes with each the metrics	119
A.1	DCS Word references without morphological analysis	137
A.2	DCS Morphological Parameters - Noun	137
A.3	DCS Morphological Parameters - Verbs (Part 1)	138
A.4	DCS Morphological Parameters - Verbs (Part 2)	138
A.5	DCS Morphological Parameters - Derivatives	138
A.6	SH Morphological Parameters - Derivatives	140
A.7	DCS - SH Primary Derivatives comparison	141
A.8	DCS-SH Mapping Example 3	142
A.9	Reference for the Tense-Mood combinations	144
A.10	DCS Tense-Mood combinations - comparison with SH	146
A.11	DCS Tense-Mood combinations - unrecognized by SH or missing in DCS	147
A.12	DCS-SH Morph Comparison - Nouns	149
A.13	DCS-SH Morph Comparison - Verbs	149
A.14	SCL-SLP Morph Comparison	150
C.1	Parameters for the Graphical Summary Mode (All)	155
C.2	Additional Parameters for the Ranking modes (First and Best)	156

Dissertation Related Articles

- Krishnan, S. and Kulkarni, A. (2019). Sanskrit Segmentation Revisited. In Proceedings of the 16th International Conference on Natural Language Processing, pages 105–114, International Institute of Information Technology, Hyderabad, India. NLP Association of India. <https://aclanthology.org/2019.icon-1.12>
- Krishnan, S., Kulkarni, A., and Huet, G. (2023). Validation and Normalization of DCS corpus and Development of Sanskrit Heritage Engine’s Segmenter. In Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference, pages 38–58, Canberra, Australia (Online mode). Association for Computational Linguistics. <https://aclanthology.org/2023.wsc-csdh.3>
- Krishnan S., Kulkarni A., and Huet, G. (2024) Normalized Dataset for Sanskrit Word Segmentation and Morphological Parsing. In Language Resources and Evaluation, Springer Nature, ISSN 1574-0218. <https://link.springer.com/article/10.1007/s10579-024-09724-0>

Chapter 1

Introduction

Segmentation is the first step towards analysing sentences in any language. Segmentation is done at various levels: word, compound word, sentence and topic. Word (and compound word) segmentation is a low-level task of Natural Language Processing where the unsegmented sentence is split into lexically and morphologically valid words.¹ Languages like English and German have explicit boundaries like space which indicate the split locations. Languages like Sanskrit, Chinese and Japanese have a tendency to join the words making the process of segmentation challenging. Words are joined either by concatenation or by euphonic transformation at the word boundaries. While in languages like Chinese, words are joined merely by concatenation, in languages like Sanskrit the consecutive words may additionally undergo euphonic changes.

In order to arrive at the individual words in any language, the primary requirement is the notion of a word. The definition of a word varies according to the language and hence language-specific word segmentation tasks are near-perfect for some languages (Shao et al., 2018). In Sanskrit, a word (referred as *padam*)² is either a noun form or a verb form.³ Segmenting a Sanskrit sentence into such individual words requires lexical and morphological validation of the words thus formed. Multiple meanings of the same word form, or multiple senses of the same stem or root (homonymy), or multiple morphological analyses of the same word form (syncretism) lead to non-determinism during segmentation. In addition to these, the phenomenon of *sandhi* (euphonic transformations at the word boundaries) is another major cause for non-determinism, as there is a tendency to write Sanskrit texts continuously without any breaks. *Pāṇini's Aṣṭādhyāyī*, the well celebrated Sanskrit grammar, records formation of all the pos-

¹ Word segmentation is also called as tokenisation, where the split units are termed as tokens.

² *suptiṇantaṃ padam* (*Aṣṭādhyāyī* 1.4.14)

³ The nominal stems (*prātipadikas*) are inflected with the nominal suffixes (*sup*) to produce the nominal forms (*subantas*) and the verbal roots are inflected with the verbal suffixes (*tiṇ*) to produce the verbal forms (*tiṇantas*).

sible *sandhis*. There are two types of *sandhis*: internal and external. Internal *sandhis* correspond to the transformation that occurs within a word, predominantly occurring during the generation of a word. External *sandhis* occur across words and at the word boundaries. An example of *sandhi* occurring in a sentence can be observed in the unsegmented-segmented pair of the following verse (*Bhagavad Gītā* 1.1):

Unsegmented

dharmakṣetre kurukṣetre samavetā yuyutsavaḥ
māmakāḥ pāṇḍavāścaiva kimakurvata sañjaya

Segmented

dharmakṣetre kuru-kṣetre samavetāḥ yuyutsavaḥ
māmakāḥ pāṇḍavāḥ ca eva kim akurvata sañjaya

Translation

Dhṛtarāṣṭra said: O Sañjaya, after gathering on the holy field of Kurukṣetra,
and desiring to fight, what did my sons and the sons of Pāṇḍu do?

We notice the following euphonic transformations that occur in the verse:

- *samavetā yuyutsavaḥ* → *samavetāḥ yuyutsavaḥ*⁴
- *pāṇḍavāścaiva* → *pāṇḍavāḥ ca eva*⁵
- *kimakurvata* → *kim akurvata*
- *dharmakṣetre* → *dharmakṣetre*, “-” indicating that it is a compound word. Similarly *kuru-kṣetre*.⁶

Compounds are special constructions of words where two words combine to form a new word with a different meaning, either modifying the first or the second word, or referring to a notion expressed externally to the compound word. In this case, *dharmā* (duty) and *kṣetre* (in the field / place), combine to form *dharmakṣetre* (in the field of duty) and is derived from *dharmasya* (of duty) *kṣetre* (in the field).

Segmenting a Sanskrit sentence thus involves three tasks:

⁴ *visarga-lopa-sandhiḥ*

⁵ *ścutva-sandhiḥ* and *vṛddhi-sandhiḥ* respectively

⁶ The meanings of these words do not contribute to the *sandhi* operation and hence not provided.

1. resolving *sandhis*,
2. identifying valid words, and
3. deciding whether the split words constitute a compound or are standalone words.

In each of these three tasks, non-determinism exists and segmentation involves resolving the non-determinism by arriving at the intended segmentation from a list of possible segmentations.

1.1 Motivation

The rules of *Sandhi* along with a morphological analyser to validate the segments do help in the segmentation process. In spite of such validation, there is always a possibility of non-determinism in segmentation, which can be observed at four levels: *sandhi*, word, compound and morphological analyses, which are described ahead.

- (a) **Sandhi:** Let us consider the word *rāmālayaḥ*. The first step during segmentation is the identification of the split location. Here it is the fourth letter *ā*, which can be obtained from the following four possible combinations:

- $a + a$ (*rāma* and *alayaḥ*),
- $a + \bar{a}$ (*rāma* and *ālayaḥ*),
- $\bar{a} + a$ (*rāmā* and *alayaḥ*),
- $\bar{a} + \bar{a}$ (*rāmā* and *ālayaḥ*),

Arriving at the correct combination requires more information regarding the lexical and morphological validity of the segments.

- (b) **Word and Compound:** Looking at the surface forms alone for the same example, there are 7 possible segmentation solutions (table 1.1). Of these, the first two analyses are compounds and the remaining are standalone words. Only the first two are grammatically and semantically meaningful. The remaining five are either grammatically or meaningwise incompatible.

(c) **Morphological Analysis:** The task of segmentation further becomes complicated, if one also selects the segments not on the basis of forms but on the basis of their morphological analyses. The two factors homonymy and syncretism thus bring in non-determinism, as illustrated below. Let us consider the words *rāmaḥ* and *vanam*.

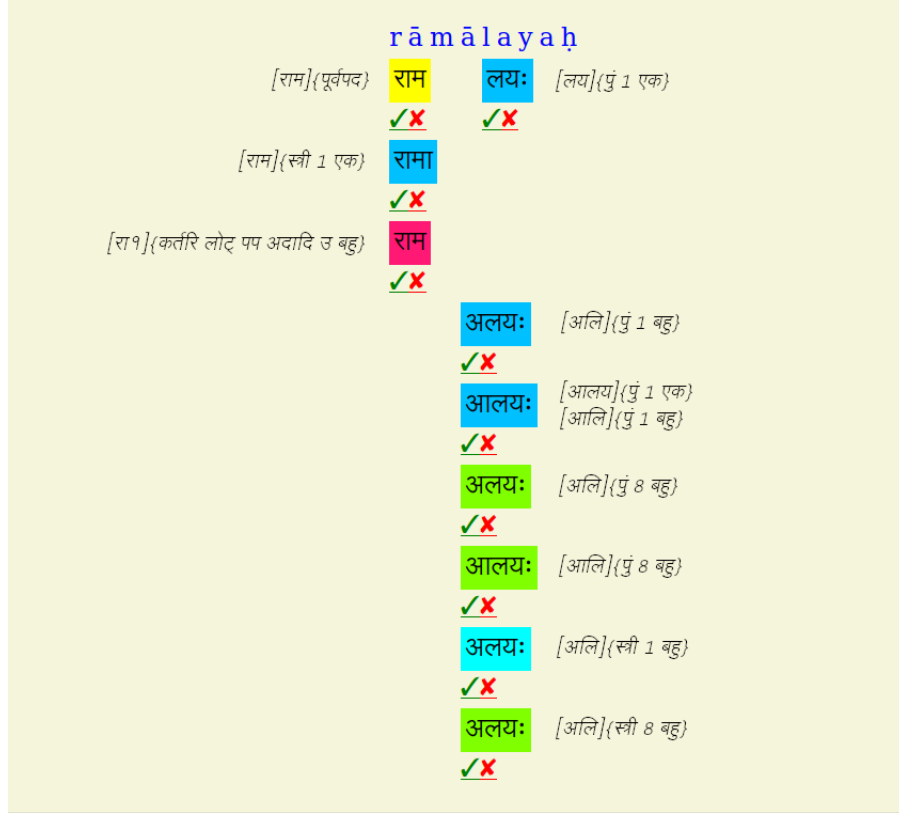
- **Homonymy** → depending on the context, *rāmaḥ* can either be a noun (masculine singular nominative form of the stem *rāma* (indicating a person named Rāma)) or a verb (present active first person plural form of the root *rā* (to give)).
- **Syncretism** → *vanam* can either be in the nominative or in the accusative case for the stem *vana* (a forest).

In the example *rāmālayaḥ*, if we consider the surface forms along with the morphological and lexical analysis, there are 17 possible solutions (figure 1.1). The colors correspond to different morphological analyses indicating different parts of speech. Among these again only 8 are grammatically and semantically compatible splits.

Table 1.1: Possible Segmentations the word *rāmālayaḥ* (word-forms)

Solutions of <i>rāmālayaḥ</i>	Meaning
1 <i>rāma-ālayaḥ</i>	<i>rāma</i> -(Lord Rama's) <i>ālayaḥ</i> (abode)
2 <i>rāma-alayaḥ</i>	<i>rāma</i> -(Lord Rama's) <i>alayaḥ</i> (scorpions / bees) <i>rāma</i> -(Lord Rama's) <i>alayaḥ</i> (non-destruction / non-dissolution)
3 <i>rāma ālayaḥ</i>	<i>rāma</i> ((we) give / bestow) <i>ālayaḥ</i> (abode)
4 <i>rāma alayaḥ</i>	<i>rāma</i> ((we) give / bestow) <i>alayaḥ</i> (scorpions / bees) <i>rāma</i> ((we) give / bestow), <i>alayaḥ</i> (non-destruction / non-dissolution)
5 <i>rāmā layaḥ</i>	<i>rāmā</i> (pleasant / charming) <i>layaḥ</i> (absorption in / clinging to)
6 <i>rāmā ālayaḥ</i>	<i>rāmā</i> (pleasant / charming) <i>ālayaḥ</i> (abode)
7 <i>rāmā alayaḥ</i>	<i>rāmā</i> (pleasant / charming) <i>alayaḥ</i> (scorpions / bees) <i>rāmā</i> (pleasant / charming) <i>alayaḥ</i> (non-destruction / non-dissolution)

Figure 1.1: Possible Segmentations of the word *rāmālayaḥ* (word-forms and morphological analyses)



With these challenges, the task of Sanskrit word segmentation has seen several approaches ranging from rule-based methodologies to finite state automata and towards machine learning methods as well. The rules from *Aṣṭādhyāyī* help in deciding the possible *sandhis*, but determining the unique segmentation requires much more information. Morphological validation reduces the possible segmentations considerably. But we need additionally the context and auxiliary information (like possible dependency relation) to get the desired segmentation. There have been efforts towards building a segmenter, like the Sanskrit Heritage Platform,⁷ that produces all possible analysis. Choosing the intended segmentation requires additional information like the context. Machine learning approaches do provide a way out but the unavailability of sufficient annotated dataset stands as an obstacle. There have been efforts towards building an annotated dataset, like the Digital Corpus of Sanskrit (DCS) (Hellwig, 2010) and the dataset for word segmentation (Krishna et al., 2017), which are commendable

⁷ <https://sanskrit.inria.fr/DICO/reader.fr.html>

but these datasets also come with limitations of their own. Some of the limitations include the unavailability of normalized datasets and differences in the design decisions for annotations in these datasets. Thus, there is a need for building a normalized dataset that encompasses and bridges the annotations of different datasets under a single setup. Along with the development of the normalized dataset, using it further to predict the correct segmentation solution of a given sandhied sentence forms the primary motivation of the present work.

1.2 Problem Statement

The Digital Corpus of Sanskrit (DCS) hosts more than 650,000 annotated sentences from around 250 texts. For each sentence, various features like lemma, morphological analysis, POS tag, links to dictionary entries, etc. are annotated. However, the corresponding segments in these sentences have not been annotated. On the other hand, Sanskrit Heritage Segmenter (SH) produces all possible segmentation solutions which includes in its analysis - the segment, lemma, all possible morphological analyses of the segment, POS tag and the *sandhi* information that occurs ahead of the segment.

Thus, there is a need for aligning the DCS analysis with one of the possible SH analyses, and in this process building a normalized dataset containing the annotations of both the systems. An alignment is previously attempted by Krishna et al. (2017) between the ground truth analysis proposed by DCS with the possible analyses proposed by SH. However, there is a lot of scope for further improvement of this alignment with additional linguistic insights and attributes.

The segmentation approaches developed recently come with a few limitations. Some models do not consider linguistic features and are purely data-driven. Some models perform compound word segmentation alone. Most of the models operate on a dataset which is not normalized. And while almost all the models resolve sandhi within a compound word, they don't mark the boundaries of its components. This introduces ambiguities during evaluation and also during the downstream NLP tasks.

The present work addresses these limitations by utilizing the normalized datasets thus produced and proposes a ranking algorithm on top of SH, leveraging the statistics generated from the normalized datasets.

1.3 Existing approaches

In the recent years, five main datasets have been created for the tasks of word segmentation and morphological analysis. These include Sanskrit-Hindi Machine Translation (SHMT) dataset, SandhiKosh (Bhardwaj et al., 2018), DCS (raw), DCS (segmentation) and “A dataset for Sanskrit Word Segmentation” (Krishna et al., 2017). SHMT dataset was an initial effort towards recording all the instances of *sandhi* and compound analysis from various Sanskrit texts. SandhiKosh used a subset of SHMT dataset along with manually generated datasets from four other sources. DCS (raw) is currently the biggest dataset resource available for Sanskrit with various features like lemma, morphological analysis, POS, etc. DCS (raw) was later re-analyzed to create DCS (segmentation) which can be used exclusively for the task of segmentation. “A dataset for Sanskrit Word Segmentation” is another reanalyzed version of DCS, where the alignment of DCS and SH analyses had been attempted for the first time resulting in a dataset of 107,000 sentences. This forms the basis of the present work.

There have been various rule based, probabilistic and machine learning approaches to address the problem of Sanskrit Word Segmentation. A finite-state-automata based approach can be observed in the Sanskrit Heritage Platform and (Mittal, 2010) while the latter also proposes a probabilistic method. Subsequent probabilistic efforts can be observed in (Kumar et al., 2010; Natarajan and Charniak, 2011). Machine-learning approaches are aplenty and include Hellwig (2015b); Hellwig and Nehrdich (2018), Krishna et al. (2016, 2018), Reddy et al. (2018); Aralikatte et al. (2018), Dave et al. (2021) and Sandhan et al. (2022).

1.4 Salient features of the current work

1.4.1 Differences between DCS and SH annotations

The annotations (segmentation and morphological analysis) from the DCS corpus are extracted and compared with the analyses proposed by the Sanskrit Heritage Platform. The linguistic differences between the two systems are recorded which helps towards standardization of annotations.

1.4.2 Normalized Dataset Generation from DCS and SH

A unified normalized dataset is created from DCS and SH, which can be used for both the tasks of segmentation and morphological analysis. The normalized dataset comprises of the unsegmented-segmented parallel corpus with the compound boundaries marked, morphological analysis of each of the segments (from both DCS and SH), POS tags, *sandhi* annotations, links to dictionary and occasionally the word senses.

1.4.3 Ranking SH segmentations

A ranking algorithm is proposed on top of the SH segmenter that prioritizes the segmentation solutions based on joint unigram probabilities of the segments. The ranking metrics uses the dataset generated previously from DCS and SH.

1.4.4 Joint task of Word Segmentation and Morphological Analysis using SH

SH produces three levels of analysis: *word*, *phase* (POS)⁸ and *morphological analysis*. Ranking is experimented at the level of word and at the phase and morphological analysis levels combined, thus making it either a word segmentation tool (producing only the words) or a word segmentation and morphological analysis tool (producing the words and their morphological analyses).

1.4.5 Compound Boundary Indication

The analysis produced by SH is rich in both lexical and morphological information, evident from its *phases* (POS) and the possible morphological analyses encoded in each of the phase. This is true of compound components also. Thus, the normalized dataset obtained from the alignment process has the compound component boundaries marked explicitly. And the ranking mechanism too, considers compound segmentation along with word segmentation, but treats both of them differently because the *sandhi* between compound components occurs predominantly across non-inflected forms.

⁸ *Phases* in SH are Sanskrit-specific POS tags.

1.5 Organization of the Thesis

Chapter 1 which is the introductory part of the thesis deals with the motivation, problems and aims of the current study. It explores the issues of segmentation and morphological analyses and gives a short account of datasets. It provides a basis for understanding the existing scenario on segmentation and morphological analysis.

Chapter 2, titled **A Review on *Sandhi*, Datasets and Segmentation Approaches**, explains in detail the traditional (grammatical) perspective of word, *sandhi*, segmentation and morphological analysis, followed by a note on the differences between the approaches for generation and analysis. It gives in greater detail the available datasets for segmentation and morphological analysis, along with the various implementations for the tasks. It finally discusses about Sanskrit Heritage Platform (SH), and how it can be used to generate an annotated dataset for segmentation and morphological analysis, and how it can further be used as a standalone segmentation tool. Finally, the need for aligning the DCS and SH analysis is discussed followed by the need for ranking the solutions in SH.

Chapter 3, titled **Overcoming Linguistic Issues in Alignment**, discusses the linguistic differences between DCS and SH which are to be considered prior to the alignment process. Further, it also records the details on how to overcome these differences during the alignment procedure.

Chapter 4, titled **DCS-SH Alignment**, discusses the alignment procedure in detail, along with the observations. Four alignments were carried out and their results are recorded.

Chapter 5, titled **Ranking Segmentations of Sanskrit Heritage Segmenter**, proposes the Segmentation Ranking algorithm on top of SH's segmentation engine. An illustration of how the alignment results were used along with comparisons with the recent state-of-the-art models is recorded.

Chapter 6 concludes the thesis and discusses the future perspectives of the study.

Chapter 2

A Review on *Sandhi*, Datasets and Segmentation Approaches

As is the case with any natural language, Sanskrit sentential processing, comprising the tasks of segmentation (or tokenization), morphological analysis, dependency parsing and sense disambiguation, has seen approaches in three ways: rule-based (or linguistically motivated), data-driven and most recently a hybrid of the two. Deciding the approach depends on the task along with various factors including available rules for the specific tasks, computational complexity of the tasks, sufficient availability of data for each of the tasks, etc. While there have been significant efforts towards segmentation and morphological analysis, using both rule-based as well as data-driven approaches, dependency parsing has seen a few efforts comprising of both the approaches, but word sense disambiguation has not seen much of a significant work (at the time of this writing). One major limitation for the application of data-driven approaches for these tasks corresponds to the low-resource nature of Sanskrit. Although Sanskrit being abundant in resources ranging from various disciplines,¹ the availability of digital, more importantly machine-readable, resources is alarmingly low when compared to languages like English, Chinese, etc. And raw machine-readable data alone would not be sufficient as the tasks' complexity increases with more linguistic features coming into play. For example, the notion of *sandhi* plays a crucial role for the task of segmentation, and the rich morphology of Sanskrit makes it even more harder to arrive at the intended segmentation. This urges the need for annotated datasets in each of the levels of processing.

For a rule-based approach, the methodology involves rigorous analysis producing all possible rules and exceptions obtained either from existing linguistic treatises or

¹ Disciplines including literature (*sāhitya*), philosophy (*darśana*), language and linguistics, grammar (*vyākaraṇa*), logic (*nyāya*), exegesis (*mīmāṃsā*), and various mathematical and scientific disciplines including astronomy and chemistry.

from examples for each of the cases. On the other hand, for data-driven approaches, consisting of statistical (or probabilistic) approaches and machine learning (deep learning) approaches, one would require a large dataset that encompasses excerpts from various domains, all annotated according to the task in hand. The datasets are to be large enough to have three sets for the three stages, namely training, development and testing. Annotations are generally task-dependent, and hence the annotation schema employed for a particular task may or may not be of use for another task. For example, for segmentation, the preliminary requirement is a parallel corpus of unsegmented-segmented sentences. Additional information like morphological analysis for each of the segments, might also come into play for a fine-grained analysis. For morphological parsing, the requirement is again a parallel corpus of sentences with segments of the sentence and their corresponding morphological analyses and the specific analysis according to the context. Furthermore, for dependency analysis, the segments, their possible morphological analyses, contextual morphological analysis and the relation between the words form the primary requirements of the dataset.

There has been a surge in the digitization of Sanskrit texts (most notably Gretil, DCS, Vedic Heritage Portal, etc.) where some of them (like DCS) even go a step further by producing annotated corpora for the various tasks. However, even with advancements in technology, we are forced to rely upon manual (or to some extent semi-automatic) annotation. Various tools and platforms have been built (and under development) for assisting the annotators (Sanskrit Heritage Platform, SanskritTagger (Hellwig, 2009), START (Kumar et al., 2024), etc.). Though what can be achieved with the amount of dataset we have is relatively low when compared to other languages, some efforts have been taken up where various models have been built exclusively for such low-resource setup (like Sandhan et al. (2023)). Nevertheless, efforts towards developing annotated datasets continues further with and without the use of such annotation tools. In the present chapter, we will look at various efforts for developing annotated datasets, specifically for the tasks of segmentation and morphological parsing. We will also look at the annotation schemas deployed in each of the datasets. Furthermore, we will discuss the various implementations of segmentation in Sanskrit.

Before diving into the details of each of the datasets, we will first look into the perspectives of traditional Indian grammarians on segmentation and morphological

analysis. This chapter starts with section 2.1 speaking about the grammarian’s view on words, morphology, the phenomenon of *sandhi* and the need for segmentation (*sandhi-viccheda*) and morphological analysis (*pada-viśleṣaṇam*). It also gives an account on how *Aṣṭādhyāyī* depicts the generation of a word and how it helps in extracting information for morphological analysis. Section 2.2 gives the details of all the datasets available for the task of segmentation and morphological analysis, along with their annotation schemas, highlighting the Digital Corpus of Sanskrit, which forms the base dataset for the present work. Section 2.3 provides a detailed overview of all implementations of segmentation in Sanskrit. It highlights the Sanskrit Heritage Platform which has been used extensively in the current work. Finally, section 2.5 proposes the need for an alignment of DCS and SH to create a unified dataset for segmentation and morphological parsing.

2.1 Traditional Grammarian’s perspective

For developing annotated datasets and also for NLP tasks like segmentation, the primary requirement is to understand how a language defines its basic constituents, starting from a word. Every language may have its own way of defining a word, each of them providing different ways of annotating a word. While a universal annotation schema is possible, that is based on a universal definition of a word, as observed in Shao et al. (2018), for languages with rich morphology like Sanskrit, a language-specific definition is additionally required because of the morphosyntactic nature of the structure of a word. It is thus worth exploring the definitions and observations of traditional grammarians (of Sanskrit), who mostly revolve around the definition proposed by *Pāṇini*.

2.1.1 Structure of a word

A word in Sanskrit can be either a noun (*subanta*) or a verb (*tiñanta*) according to the definition proposed by *Pāṇini*.² *Subantas* are words ending with *sup*-suffixes. These suffixes are twenty one in number and are affixed to *prātipadikas* (stems) to generate inflected forms (declensions) of nouns. *Tiñantas* are words ending with *tiñ*-suffixes.

² *suptiñantam padam - Aṣṭādhyāyī* (1.4.14)

These are eighteen in number and are affixed to *dhātus* (roots) to generate inflected forms (conjugations) of verbs. Thus the basic morphemes in Sanskrit include the *prātipadikas* and *dhātus*.

While there are about 2000 *dhātus*³ enlisted by *Pāṇini* in his auxiliary text *dhatū-pāṭha*, each referring to a particular action, these can be expanded further to form derived *dhātus* using the *san*-suffixes.⁴ On the other hand, *prātipadikas* can be underived or derived either from the *dhātus* or from other *prātipadikas*. Those which are derived from *dhātus*, using a certain set of suffixes (*kṛt*), to generate verbal forms are termed as *kṛdantas* (primary derivatives). Those which are derived from *prātipadikas* using certain other set of suffixes (*taddhita*) are termed as *taddhitāntas* (secondary derivatives). Both *kṛdantas* and *taddhitāntas* are again *prātipadikas* to whom *sup*-suffixes can be affixed to form noun forms. In addition to these, compound stems are also considered as *prātipadikas* where two or more words combine to form a single word, and the *prātipadika* thus formed is inflected with *sup*-suffixes.⁵ There are certain other words termed as *avyayas* (indeclinables) which include forms like the particles, conjunctions, interjections, etc. According to *Aṣṭādhyāyī* though, these are *subantas* whose *sup*-suffix is finally elided. *Pāṇini* enlisted an arbitrary set of such “indeclinable” words (*svarādi* - *svar*, *antar*, *hyas*, etc.)⁶ in his auxiliary text (*gaṇapāṭha*). *Avyayas* also include words with some specific *kṛt*-suffixes (like *tumun*) and *taddhita*-suffixes (like *tasil*), and also the *avyayībhāva* compound words.

2.1.2 *Sandhi* and Segmentation (*sandhi-viccheda*)

Sandhi is the process of euphonic transformation that occurs either at the morph or word boundaries. *Sandhi* that occurs at the morph boundaries is predominantly during the word formation, and is termed as internal *sandhi*. *Sandhi* when occurring across two words at their boundaries is termed as external *sandhi*. *Pāṇini*, in his *Aṣṭādhyāyī*, provides a huge list of rules for both the types. *Aṣṭādhyāyī*, in general terms, is a framework for generation purposes, which includes generation of words from the root

³ *bhūvādayo dhātavaḥ* - *Aṣṭādhyāyī* (3.1.1)

⁴ *sanādyantāḥ dhātavaḥ* - *Aṣṭādhyāyī* (3.1.32)

⁵ The *kṛdanta*, *taddhitānta* and *samasta-pada* are considered as *vṛttis*. More about these in the subsequent sections.

⁶ *svarādinipātamavyayam* - *Aṣṭādhyāyī* (1.1.37) - the list of words starting from *svar* in the *gaṇapāṭha* and also the words under the category *nipātas* are termed as *avyayas*.

and stems, derivation of *vr̥ttis*,⁷ the action of *sandhi* to join the words, generation of a sentence using the semantic roles (*kāraṇa*), etc. On the other hand, processing Sanskrit (or any language) requires the analysis which is the reverse process of generation, using the cues obtained from the rules of generation.

As most of the texts in Sanskrit are written continuously with the occurrence of *sandhi* (external) between the consecutive words,⁸ it becomes necessary to split them for further processing. The tradition does not propose any methodology that exclusively does this segmentation, as the onus is left on the reader to do the process of segmentation mentally by simultaneously detecting possible *sandhi* locations using *sandhi* rules and validating the morphological correctness of the discrete words thus formed. Thus possible application of various *sandhi* rules and morphological validation of the words form the crucial steps during segmentation. When faced with multiple possibilities, these are disambiguated using relevance in the given context (*yogyatā*). This is true while resolving all external *sandhis*. As internal *sandhis* are enforced during the formation of words and generation of some of the *vr̥ttis*, resolving them goes hand in hand with morphological analysis.

A brief account on the rules

The rules of *Sandhi* involve majorly two operations as proposed in *Aṣṭādhyāyī*: substitution (*ādeśa*) and deletion (*lopa*). That which is substituted is referred to as the *sthānin*, and the operations are carried on the *sthānin* depending on the context which could be either the left or the right, according to the rule. These are implemented at two situations: one, during the generation of a word from its morphemes (internal), and two, in *saṃhitā*, across word boundaries (external). Thus morphemes are sandhied to form a word and words are sandhied to form the *saṃhitā*. *Aṣṭādhyāyī* provides a uniform set of rules applicable at both the scenarios, with some exceptions. We shall consider the external *sandhi* under the purview of segmentation and internal *sandhi* under morphological analysis. So under external *sandhi*, the two operations

⁷ *vr̥ttis* exhibit a sense different from what was originally inherited in the word. There are five *vr̥ttis*, namely *kṛt* (primary derivation), *taddhita* (secondary derivation), *saṃāsa* (compound words), *ekaśeṣa* (special case of compounds where only one of its components remains and indicates the union of the components) and *sanādyanta-dhātu* (derivation of verbal forms).

⁸ This process of writing texts with the *sandhi* is termed as *saṃhitā*.

mentioned earlier apply themselves across two words and an abstract representation of a *sandhi* can be attempted as:

$$u + v \rightarrow w \quad (2.1)$$

where, u refers to the final part of the first word, and v refers to the initial part of the second word, and w , the resultant. u could have atmost two characters, v one character, and w zero to two characters (zero indicating a *lopa*). Thus when two consecutive words undergo *sandhi* (external) operation, then the following are the possible changes:

- only u undergoes change,
- only v undergoes change,
- both u and v undergo change,
- neither u nor v undergoes change.⁹

The intention behind the *sandhi* operation across words is to have a smooth transition from one word to another during the pronunciation of the words, hence these are termed euphonic transformations. This made sure that almost all the texts that have been written in Sanskrit, are in the *saṃhitā* form. Given a sandhied text, it creates a need for the reversal of the *sandhi* operation, which is segmentation (*sandhi-viccheda*).

As mentioned earlier, there isn't any specific rule addressing segmentation, but the rules of *sandhi* are taken into consideration here as well, for preparing the set of triplets u, v and w . All the *sandhi* rules have these three constituents, u and/or v as the modified, with w as the modifier in a given context. For example, consider the *sūtra* “*iko yaṇaci*”, which says that in the context of *ac* (viz. when followed by a vowel), the letters from *ik* (i, u, r, l) are substituted with the letters from *yaṇ* (y, v, r, l), respectively.

The resultant of a *sandhi* operation is typically unique, except for certain situations which have multiple optional resultants. On the other hand, given a resultant, obtaining the constituents prior to the *sandhi* operation is ambiguous. For example, the resultant $\bar{a}$ can be obtained from one of:

⁹ This is where no *sandhi* rule applies and the *varṇas* are just conjoined to form a *saṃhitā* as in *rāja-puruṣaḥ*.

1. $a + a$
2. $a + \bar{a}$
3. $\bar{a} + a$
4. $\bar{a} + \bar{a}$

And disambiguation of the *sandhi* requires the morphological validation of the words thus formed, along with the context. Thus we observe non-determinism at the very first level (*sandhi*) of processing during segmentation, then followed by non-determinism at the word and then at the morphological analysis level, and ultimately at the word-sense level. Non-determinism occurs even in a human mind, but due to the presence of world knowledge and usage statistics in a human mind, we find it easy and quick enough to resolve a sandhied resultant into its constituents. For a machine though, one can either feed sufficient data for its learning, and ask it to predict based on the existing data, or one can build the segmentation mechanism in stages where we first check the morphological validation of the words in each of the solutions thus formed, and then prioritize the validated solutions based on the context. This forms the basis of the present work.

2.1.3 Generation vs Analysis

We note here that the tasks of segmentation and morphological analysis are typically the reverse processes of *sandhi* joining and word generation, respectively. In both the cases, *sandhi* is involved either externally or internally. While we are searching for inflected forms in segmentation, we are looking at the individual morphemes and the corresponding features of a given word. We now observe some of the core aspects of the differences between the two approaches.

(Non-)Determinism

One major difference between the approaches of analysis and generation lies with (non-)determinism. *Sandhi* joining is predominantly deterministic in nature, with a very few exceptions. On the other hand, we observed how segmentation is non-deterministic in nature. Similarly, morphological generation is deterministic in na-

ture. Given a *prātipadika*, its gender, case and number, we can arrive at a single nominal form. We can arrive at a verbal form, given a *dhātu*, *lakāra*, *prayoga*, person and number. This is also applicable during the derivation of new roots and stems. But there is non-determinism during morphological analysis, both with the inflectional and derivational analysis.

Furthermore, we can observe non-determinism at various levels. For example, consider the word *rāmaḥ*. It could either be a *subanta* or a *tiñanta*. We can also observe non-determinism within a category, amongst its inflected forms. For example, *rāmābhyām* has the same form for three inflections of the stem *rāma* when in dual: in the third, fourth and fifth cases.¹⁰ It can also be observed during the derivation. For example, the stem *hita* can be derived from either *hi* or *dhā*. We can also observe non-determinism at the level of stems or roots, with respect to their meanings.¹¹ For example, there are three variations of the *dhātu* *kṛ* depending on the meaning, and their forms also differ. In order to handle them separately, these are encoded as *kṛñ* (*karāṇe*), *kṛñ* (*hiṃsāyām*) and *dukrñ* (*karāṇe*), respectively in the 1st, 5th and the 8th class. We can observe how the same root *kṛ* refers to an action of doing something in general (*karāṇe*) or specifically to an action of violence (*hiṃsāyām*). This is at a higher level where we have to consider sense-disambiguation techniques to arrive at the intended meaning. Non-determinism continues to be observed in the subsequent stages of language processing, and with the cascading effect, the chunk of possibilities tend to increase as we go through downstream NLP tasks.

It is worth noting here how the system of rules proposed by *Pāṇini* in the *Aṣṭādhyāyī* is *generative* in nature. “It explains how a Sanskrit locutor with a communicative intention is to proceed in order to construct a correct sentence with the intended meaning. It thus proceeds from semantics to syntax to morphology to phonology, applying rules until one obtains a terminal stream of phonemes representing the correct enunciation of a syntactically correct paraphrase of the intended meaning (Huet, 2007).” We also go further by incorporating the *sandhi* rules on the terminal stream of phonemes resulting into the *saṃhitā* form. At each step, the decisions of the previous step(s) are mandatory. Thus, the exact reversal of these processes would definitely

¹⁰ This phenomenon is termed as syncretism.

¹¹ This phenomenon is termed as homonymy.

yield multiple possibilities, especially due to constraints based on the decisions made at each step.

Fortunately, at the level of morphological analyses, there have been various efforts using finite state automata for detecting the morphological information using the generation principles. A paradigm based approach where the paradigms are fed to the FSA, capturing the common suffixes along with the modifications that occur at the boundaries of the morphemes help in mapping the final form with its individual morphemes. This helps in identifying all the possible morphological analyses of a given word. To arrive at the intended analysis, though, one has to rely on the context.

Linguistic Context

Context plays an important role during segmentation and morphological analysis. Given an unsegmented sentence, there could be an enormous number of possible segmentation solutions. As segmentation involves resolving *sandhi* at possible locations, context helps in choosing the right split, and in turn the right location and segments. Similarly for morphological analysis, given a word, we could have many possible analyses, and with the context, we can narrow down to the required analysis. But we have to define context prior to the process.

Context could be identified at various levels. For segmentation, the remaining segments of a sentence could be considered as the context. Sometimes one has to look into a few of the previous sentences in a given text, to identify the intended splits. Similarly for morphological analysis, we would require the analysis of other words in the sentence. In the sentence *rāmaḥ vanam gacchati* (Ram goes to the forest), *rāmaḥ* could either be a noun or a verb without considering context. So is *gacchati* where the noun is from the stem (*gacchat*) derived from the root *gam*. We can have one *tiṇanta* in a sentence according to the definition proposed by *Kātyāyana* in his *Vārtikas*.¹² With further help from the *śābdabodha* theories, specifically the triad of *ākāṅkṣā* (expectancy), *yogyatā* (compatibility) and *sannidhi* (proximity), we can deduce that the verb *rāmaḥ* expects a *karma* that is concrete and givable and the verb *gacchati* has an expectancy of both *kartā* and *karma* (who goes and where). On the other hand, the usage of the noun *gacchati* requires another word in its own case and number

¹² *eka-tiṇ vākyam* for *Aṣṭādhyāyī* 2.1.1

to fulfil the expectancy of the verb in *sati-saptami* (absolutive locative) usage. With respect to the word *vanam*, we have two noun forms from two inflections on the stem *vana*, namely nominative and accusative. The verb *gacchati*, being in active voice (*kartari-prayoga*), expects its *kartā* in the nominative case, in this case *rāmaḥ* and the *karma* in the accusative case, in this case *vanam*. All these information help us in extracting the relations of a word with other words in the sentence, which become crucial in deciding whether the chosen morphological analysis is applicable in the sentence or not, thus validating the entire sentence.

Thus, given a segmentation solution, and the corresponding morphological analysis of each of the segments in the segmented sentence, we can come up with the possible relations (called dependency relations) between the words. If we are unable to find relations for any of the words, or if the relations deduced are found to be incompatible with each other, the segmentation solution is syntactico-semantically invalid. So, first we validate using the *sandhi* rules and the morphological analysis of the splits obtained after the application of the *sandhi* rule at the word/morphology level. Then with the dependency analysis, we validate the sentence at the syntactico-semantic level. We can go further and do the disambiguation of various senses of a word. The sense of the stem *vana* (according to the Monier-Williams dictionary) are:

a forest, wood, grove, thicket , quantity of lotuses or other plants growing in a thick cluster (but in older language also applied to a single tree) Lit. RV. plenty , abundance Lit. R. Lit. Kathās. a foreign or distant land Lit. RV. vii , 1 , 19 (cf. [araṇya]) wood , timber Lit. RV. a cloud (as the vessel in the sky) Lit. ib. water Lit. Naigh. i , 12 a fountain , spring Lit. L. abode Lit. Nalôd. Cyperus Rotundus Lit. VarBṛS.

While the most probable meaning is forest, there could be specific (rather rare) occurrences of *vanam* with other senses, identification of such cases requiring more context. Here, we include the entire paragraph (discourse) into consideration or at least a few of the previous lines along with some of the next few lines. Considering the discourse, we can handle sentences like *śveto dhāvati*, whether it is *śvā itaḥ dhāvati* (The dog runs here) or *śvetaḥ dhāvati* (The white one runs) or more precisely *śvetaḥ dhāvati* (The (white) horse runs - where *śvetaḥ* refers to the white horse and it is obtained from the discourse).

Extra Linguistic Context

There could be three *vr̥ttis* of the meanings of a word, namely *abhidhā*, *lakṣaṇā* and *vyañjanā*. While *śābdabodha* deals with the power of verbal cognition, these *vr̥ttis* deal with the power of meaning. *Abhidhā* refers to the primary (literal) meaning of the words, *lakṣaṇā* to the secondary meaning, when the primary meaning is incompatible, and *vyañjanā* to a (suggestive) meaning that cannot be obtained either from the primary or the secondary meanings. Given a word, a literal meaning can be mapped using a dictionary, but we still have to check if the literal meaning is applicable in the given context or not. If not for any of these, then a suggestive meaning needs to be derived in order to fit the meaning proposed by the context.

All of this expands the possibilities by introducing various parameters like the situation, the speaker’s intentions, the listener’s capability of grasping the intended meanings, and also world knowledge. Each of these parameters become crucial in the *vr̥ttis* mentioned above. There could be various other parameters like influence of culture, neighbouring languages, abbreviations or shortening of words, etc. But Sanskrit, having crossed so many centuries and multiple transitions of different kinds of cultures, has not been much influenced by most of these parameters.

While some of the linguistic context (like *ākāṅkṣā* and *sannidhi*) have been considered during a rule-based implementation of segmentation and morphological analysis (and also in the dependency analysis), some (like *yogyatā*) have been implemented partially, and some (like *tātparya*) have not been yet explored, along with almost all of the extra linguistic context. Machine-learning based approaches can come to the rescue in such situations, which require a large amount of annotated data for training and development. We will now look at various datasets available for the tasks of segmentation and morphological analysis.

2.2 Datasets and Annotations for Segmentation and Morphological Analysis

The previous decade saw various datasets released for the tasks of Segmentation and Morphological Analysis in Sanskrit. The widely used datasets are elaborated here.

2.2.1 SHMT

Sanskrit Hindi Machine Translation (SHMT) consortium¹³ developed a corpus for segmentation.¹⁴ It has the following data:

1. word-frequencies (27,704 unique words from 284,930 word references),
2. compound-component frequencies (16,130 unique entries from 118,302 compound component references),
3. frequencies of sandhi between words (724 unique sandhi rules from 280,622 sandhi instances across words),
4. frequencies of sandhi between compound components (381 unique entries from 78,907 sandhi instances within a compound),
5. unsegmented-segmented parallel corpus extracted from annotated texts (81,431 entries),
6. parallel dataset of compounds annotated with the constituency structure and compound type information (59,326 entries),
7. Part-of-Speech tagged corpus,¹⁵ and
8. proof-read corpus for around 90 texts.¹⁶

All the tagged corpora have the original text in Devanagari notation, and the frequencies are stored in ‘.tsv’ format with the entries in WX notation.

2.2.2 Sandhi-Kosh

Bhardwaj et al. (2018) created a corpus, named SandhiKosh which comprises of five sub-corpora that provide for a complete coverage of all the *sandhi* rules of *Aṣṭādhyāyī*.

These five sub-corpora include the following:

¹³ A consortium under the funding from DeiTY (Department of Electronics and Information Technology), Govt. of India (2008-12)

¹⁴ https://sanskrit.uohyd.ac.in/scl/GOLD_DATA/tagged_data.html

¹⁵ This corpus is available for various texts from different genres like drama, short stories, *kāvya*, *purāṇa*, articles etc.

¹⁶ These texts are also from different genres like drama, short stories, *kāvya*, *purāṇa*, articles etc. For some texts, only the raw proof-read version is available. For some, the sandhi-split corpus is also available. For some, even the compound types are tagged.

1. **Rule-based corpus:** Unique examples for each of the *sandhi* rules from *Aṣṭādhyāyī* were collected and classified them as examples of either internal or external *sandhi*. 150 examples for internal and 132 examples for external *sandhi* were collected.
2. **Literature corpus:** Contains 150 examples from 11 different literary texts.
3. **Bhagavad-Gītā Corpus:** 1,430 examples for external *sandhi* were created from the first nine chapters.
4. **UoH Corpus:** 9,368 examples were extracted from the SHMT corpus for external *sandhi*.
5. **Aṣṭādhyāyī corpus:** 2,700 examples from the *sūtras* are considered here.

While the size of this dataset is relatively low for use in statistical or machine learning approaches, this can be used for testing and evaluation purposes very well, similar to how the authors had done on various segmentation tools available at that time.

2.2.3 DCS - raw

The Digital Corpus of Sanskrit (DCS) (Hellwig, 2010)¹⁷ is an annotated dataset, rich in morphological and lexical analysis, collected from around 250 Sanskrit texts. There are about 650,000 sentences with more than 4,500,000 word references and 175,000 unique words.

There are two major versions of DCS. The first is the old DCS dump in the SQL format.¹⁸ The second version is the more recent dataset released in the CoNLL-U format.¹⁹

For every word reference, the following list of attributes is present in the old DCS dump:

1. lemma,
2. morphological class,

¹⁷ <http://www.sanskrit-linguistics.org/dcs/>

¹⁸ <https://drive.google.com/open?id=1zKHtrnRTqW6Tro0oepFgTGBsPT9D6i6k>

¹⁹ <https://github.com/OliverHellwig/sanskrit/tree/master/dcs/data/conllu>

3. CNG value,²⁰
4. preverbs (optional),
5. finite verbal form (optional),
6. infinite verbal form (optional),
7. position in sentence,
8. position inside chunk, and
9. meaning of the lemma.

DCS shifted all its data into a universal format, following the tagging guidelines and principles of universal dependencies. According to universal dependencies (UD) scheme, the morphological specification of a (syntactic) word consists of three levels of representation: lemma, POS tag and a set of features representing lexical and grammatical properties that are associated with the particular word form. The DCS CoNLL-U version approaches the morphological tagging in a similar way, but also includes a few more columns for each of the word references. The overall list of columns that DCS tags for each word reference of a sentence is as follows:

1. Position of word in the sentence,
2. Lemma,
3. Universal POS tag,
4. Language Specific POS tag,
5. Morphological Analysis (using key-value pairs),
6. Dependency Relation (both the tag and the position of the related word)
7. Miscellaneous section (which has lemma_id, semantic_id, etc.)

In 2022, DCS updated the CoNLL-U version by incorporating more fields like the unsandhied word, punctuation, is_mantra, etc., along with a few updates to the morphological analysis keys and pairs. However, these unsandhied words are either generated from a neural network model (Hellwig and Nehrdich, 2018) or has been generated using the lemma and the morphological analysis of the word.

The present work discusses in detail regarding the issues with the existing DCS representation of the values like the unsandhied term, morphological analysis, word-meanings, etc. in Chapter 3. Thus, we see that the usage of all the versions of DCS

²⁰ A value denoting the case, number, and gender of the given word, for nouns. Or the tense, aspect, person and number for verbs. Each of these is represented as an integer and given a set of features, the overall CNG value is calculated using the formula proposed in cng-calculation.

lies in various tasks in word segmentation, morphological parsing, word sense disambiguation, etc.²¹

2.2.4 DCS - segmentation

The sentences from DCS were re-analysed using the SanskritTagger software (Hellwig, 2009) in Hellwig and Nehrdich (2018). Re-analysis is necessary because the older version of DCS stored the morpho-lexical analysis of strings, but did not record the segments and *sandhi* rules applied. Thus, this segmentation data contains the surface forms of DCS sentences, the split points and the *sandhi* rules proposed by the tagger.

For 561,596 sentences, this dataset contains 2,978,509 strings and 4,171,682 tokens.²² The statistics of the dataset showed the high frequency of *sandhi* occurrences and the predominance of compounding, especially in texts from classical Sanskrit, when compared to the earlier texts (Vedic, ritualistic and Dharma texts).

DCS is curated single-handedly and thus we can assume consistency in tagging. But, according to Hellwig and Nehrdich (2018), the quality of this data tested on a small sample (50 sentences with 250 words and 2,354 characters including space) revealed that around 5.5% of the compound splits are doubtful and around 2% errors are due to annotation of the segmentations. The error analysis further shows how the compositionality of compounds influenced most of the errors.

Table 2.1 presents the analysis of the sentence:²³

pañca ratnāni mukhyāni coparatnacatuṣṭayam

pañca - five; *ratnāni* - precious gems / jewels; *mukhyāni* - important; *ca* - and; *uparatna-catuṣṭayam* - four types of uparatna (semi-precious gems)

Sandhi occurs at two locations:

- (a) between *ca* and *uparatna* to form *coparatna* (*sandhi* across two words),
- (b) between *uparatna* and *catuṣṭayam* to form *uparatnacatuṣṭayam* (*sandhi* between the components of a compound word).²⁴

²¹ For an overview of DCS' representation of its corpus, refer Appendix A.

²² A token is an unsandhied word barring compound words, and a string refers to a sequence of characters that is delimited by a space.

²³ The table does not resemble the original annotation in the dataset, but rather the information extracted from the original version.

²⁴ In this case, we don't visibly see any euphonic transformation during the *sandhi*, but only the concatenation of the two components.

Table 2.1: DCS Segmentation Analysis

Segmented form	Stem	Grammar	Sandhi Marking
<i>pañca</i>	<i>pañcan</i>	NUM	No
<i>ratnāni</i>	<i>ratna</i>	NC	No
<i>mukhyāni</i>	<i>mukhya</i>	JJ	No
<i>ca</i>	<i>ca</i>	CCD	Yes
<i>uparatna</i>	<i>uparatna</i>	NC	Yes
<i>catuṣṭayam</i>	<i>catuṣṭaya</i>	NC	-

DCS marks both these *sandhis* but does not distinguish them. Looking at the Grammar tags for each of the segments, *ratnāni*, *uparatna* and *catuṣṭayam* all have been tagged as “NC” (indicating noun category). While *ratnāni*, a standalone word not involved in any *sandhi*, should be marked as “NC”, *uparatna* should be marked as “iic.” or “CPD” indicating an initial compound component. Although the base rules governing *sandhi*, either between words or between the compound components, are the same, there is a necessity to distinguish these two, as the *sandhi* between the compound components predominantly occurs across the non-inflected forms while the *sandhi* across words occur over the inflected forms. Capturing the differences in the behaviour of inflected forms versus the non-inflected forms gives additional insights for segmentation, and also becomes crucial in the subsequent tasks such as parsing. However, this dataset is exclusively for word segmentation and hence compound splits are looked at in similar terms with word splits. Thus, differentiating the *sandhi* between words and the *sandhi* between compound components at the stage of word segmentation is debatable. This gives us a single dataset for word segmentation, unlike the SHMT dataset which has a dedicated frequency list for compound components.

2.2.5 SIGHUM - A Dataset for Word Segmentation

Due to the unavailability of the final segmented forms, and the *sandhi* rules applied, in the DCS dataset, Krishna et al. (2017) aligned the analyses of DCS’ sentences with the possible analyses of Sanskrit Heritage Segmenter (SH)²⁵ to create a unified dataset.

For each of the DCS sentences, SH analyses were scraped and aligned with the DCS analysis using the lemma and morphological analysis. A detailed explanation is

²⁵ <https://sanskrit.inria.fr/>

provided in section 2.5.1. Around 115,000 sentences from the original DCS corpus were released in the SIGHUM dataset, which had the input sequence, ground truth segmentation, and morphological and lexical information about all the phonetically possible segments.

2.3 Recent Implementations for Segmentation

Text segmentation is the broad domain which encompasses segmentation at various levels like word, compound word, sentence and topic. The present work focuses on Word Segmentation, and to some extent also discusses it in the context of compound words. The term *segmentation*, from now on, refers to word segmentation by default, unless specified otherwise. Some languages have delimiters marking explicitly the word boundaries, such as space and other punctuation markers in English or German, while the initial, medial, final and isolated forms in Arabic help in detecting the word boundaries. Some languages do not have such explicit hints and undergo either concatenation or euphonic transformation at the word boundaries. We will now look at the recent approaches in such languages and propose a comparison with segmentation in Sanskrit.

Concatenation based languages

Looking at word segmentation in languages with concatenation (Thai, Chinese and Japanese), there are two basic issues: word-boundary disambiguation and unknown word identification (Gao et al., 2005) where disambiguation happens between compound forms and individual word forms. In Thai, possible word boundaries are predicted using the syllable information and word segmentation is preceded by syllable segmentation first (Haruechaiyasak et al., 2008). In Chinese, the characters (*hanzi*) are easily identifiable and segmentation can be done by tagging (Xue, 2003), or by using deep learning techniques to determine word-internal positions (Ma et al., 2018). In languages like Vietnamese, although the space delimiter is present, it can act either as a syllable separator or as a word separator (Nguyen et al., 2020).

Differences with Sanskrit Segmentation

Treatment of word segmentation in these languages differs from that of Sanskrit primarily due to the *Sandhi* in Sanskrit. These languages (Thai, Chinese, Japanese) have concatenation of words while Sanskrit has both concatenation as well as euphonic combination at the word boundaries. Segmentation in Japanese and Chinese mainly involve predicting the word boundaries based on semantic context, grammatical cues (like particles and verb endings in Japanese) and relative frequency of character combinations. Thai differs from Chinese and Japanese in what each symbol constitutes. While a symbol is a character in Thai, it is a syllable in Chinese and Japanese. The approaches for detecting word boundaries differ significantly between these languages. In Sanskrit, the preliminary focus is on resolving *sandhi* and then picking the correct segmentation based on context. Similar *sandhi* phenomena can also be observed in Indian languages (in Tamil and Telugu), but the rules of *sandhi* vary for each language.

On the other hand, unknown word identification is a universal problem and has been approached using collocation of characters, character-based tagging and chunking, and in recent years, with contextual embeddings. Except for some efforts towards predicting the stems of noun forms (Goyal and Huet, 2016),²⁶ there is no module that can analyse or identify Out Of Vocabulary words, in Sanskrit.

Annotation Schemes

The annotation schemes followed by these languages are language-specific. The annotation in Chinese, for example, is done similar to the annotation of Multi Word Expressions (MWEs). They use B, B2, B3, M, and E tags to mark whether the character is the beginning character, second character, third character, middle character or the end character, and S is used to denote a single character word (Zhao et al., 2006). Another annotation uses {B}, {M}, {E}, and {S} where {B, M, E} denote *Begin*, *Middle*, *End* of a multi-character segmentation, respectively and S represents a *Single* character segmentation. Japanese also has a similar annotation scheme annotating the *Begin*, *Inside*, *End* of multi-character segmentation along with *Single* for single character segmentation (Kitagawa and Komachi, 2018). For languages with spaces as delimiters, an extra tag {X} is used to mark the spaces that do not belong to any word (Shao et al.,

²⁶ It requires additionally a manual validation.

2018). For Thai word segmentation, in addition to the existing scheme, few more tags are added to handle the syllable level segmentation. Chormai et al. (2020) proposes length-sensitive tagset: BI-short, BI-mid, and BI-long for words that have 1-2 syllables, 3-4 syllables and 5+ syllables, respectively. In another scheme, a BI- k tag is introduced where k represents the length of the word in terms of number of syllables.

On the other hand, the annotation schemes followed in Sanskrit segmentation are predominantly defined on the basis of the requirements of the models. For boundary detection or identification of the split location, a boundary window size is marked, where the size ranges from 2 to 5. For identification of the segments, annotation is not trivial due to the *sandhi* that happens across word boundaries, and the schemes are chosen based on the model. In an unsegmented-segmented parallel corpus, the segmented sentence has words separated by space and compound components are separated by a hyphen “-”. For models considering Sanskrit segmentations as a character level segmentation task, the characters participating in the *sandhi* phenomenon are marked explicitly.

2.3.1 Segmentation in Sanskrit

Sanskrit Word Segmentation has been addressed from various aspects, from rule-based approaches to statistical and machine learning approaches. There has been an evolution of datasets from SHMT and DCS to a refined DCS dataset in the SIGHUM dataset. Here is a summary of the segmenters developed for Sanskrit, along with their datasets, most of which were described in detail in section 2.2. Finite State Machine based approaches are discussed first, followed by statistical approaches, then the machine learning based approaches are described and finally, a short note on the limitations of these approaches is presented.

Sanskrit Heritage Segmenter

Huet (2003) developed the SH, a tool that uses Finite State Automata to handle the non-determinism of resolving *sandhi* resulting into morphologically and lexically valid segments. This was further enhanced by introducing a graphical interface which presents all the possible lexically and morphologically valid segments (Goyal and Huet, 2016). More details about these are described in section 2.4.

Automatic Sanskrit Segmentizer Using Finite State Transducers

Mittal (2010) used OpenFST (Riley et al., 2009), an open-source library for building and using finite state automata, in particular, weighted finite state transducers, for the task of segmentation. Two approaches were proposed. In the first approach, the FST was modified by linking the final states to appropriate intermediate states incorporating the *sandhi* rules. Traversing through the input string from left to right, applying the FST generates all possible splits that are morphologically valid. In the second approach, which is based on *Optimality Theory* (Prince and Smolensky, 1993), where all possible splits are generated and then each split is validated using a morphological analyser.

The baseline is considered assuming that a word can be segmented into only two segments, where the word is traversed from left to right, and is segmented by the first applicable rule from the list of 2,650 rules, provided the segments are valid morphs. With a test data of 2,510 parallel word corpus, for 52.7% of the entries, the first option was correct.

In the first approach, an FST is built with the existing set of words (taken from the corpus) along with their morphological features. This FST is further augmented using transitions from the state with the penultimate character of a word, to the state corresponding to the second character of the next word. This transition corresponds to the *sandhi* rule applied across the words. The modified FST is then traversed to find all possible splits and the weights of each of the splits is calculated.

A parallel corpus of sandhied-segmented sentences (25,000) is extracted from the SHMT dataset, along with 2,650 *sandhi* rules including the cases of mere concatenation, and the frequency distribution of these rules in the parallel corpus are generated. The estimated probability of the occurrence of a *sandhi* rule P_{R_i} is calculated. Given candidate S_j with k constituents as $\langle c_1, c_2, \dots, c_k \rangle$ by applying $k - 1$ *sandhi* rules $\langle R_1, R_2, \dots, R_{k-1} \rangle$ in between the constituents, the weight for a specific split s_j is calculated as:

$$W_{s_j} = \frac{\prod_{x=1}^{k-1} (P_{c_i} + P_{c_{i+1}}) \times P_{R_x}}{k} \quad (2.2)$$

where P_{c_x} is the probability of the occurrence of the word c_x in the corpus. P_{R_x} is the probability of the occurrence of the rule R_i in the corpus. And k is the number of individual components in the split s_j . Here the sum of the probabilities of two consecutive words are taken into account and multiplied by the probability of the *sandhi* that occurs between them.

The results showed that with 500 rules inserted into the FST, in the given test set, for more than 71% of the input words the correct solution was found to be in the first rank. But the size of the FST increased as more rules are added slowing down the entire process.

The second approach incorporated the principles of the Optimality Theory (GENerate, CONstraint, EVALuate), where all the possible splits are generated by applying the rules wherever applicable first, followed by two constraints: the constituents must be valid morphs, and split with maximum weight has to be selected. The morphological validation is evaluated using the morphological analyser (Bharati et al., 2006) provided by the Apertium group. With this approach, for 92.87% of the input texts, the correct solution was identified in the first position.

Sanskrit Compound Segmenter

Kumar et al. (2010) developed a segmenter exclusively for Sanskrit compounds using probabilistic methods and optimality theory. It re-used the second approach of Mittal (2010), using the GEN-CON-EVAL methodology, the only difference being the weight calculated for prioritizing the solutions. Here the unigram probabilities of the words were considered instead of the sum of the probabilities of two consecutive words, redefining the weight as:

$$W_{s_j} = \frac{(\prod_{x=1}^k P_{c_x}) \times (\prod_{x=1}^{k-1} P_{R_x})}{k} \quad (2.3)$$

The results indicated that for 92.46% of the input texts, the correct solution was found in the first position. Although, change in the weight calculation did not bring about a significant difference (0.41%) when compared to the previous weight.

Statistical Saṃdhi Splitting

Natarajan and Charniak (2011) proposed a statistical *sandhi* splitter where a novel posterior probability function was introduced along with Bayesian Word-Segmentation methods to handle *sandhi* formations.

A new posterior probability function is proposed, $\hat{P}(s)$,²⁷ the probability of generating the split $s = \langle c_1 \dots c_m \rangle$, with m splits, and rules $r = \langle r_1, \dots, r_{m-1} \rangle$ applied on the input, where

$$\hat{P}(s) = \hat{P}(c_1) \times \hat{P}(c_2|c_1) \times \hat{P}(c_3|c_2, c_1) \times \dots \quad (2.4)$$

$$\hat{P}(s) = \prod_{j=1}^m \hat{P}(c_j) \quad (2.5)$$

$\hat{P}(c_1)$ is the probability of occurrence of the word c_1 . $\hat{P}(c_2|c_1)$ is the probability of occurrence of the word c_2 given the occurrence of the word c_1 , and so on. This describes a generative model and does not have any rule probability. SHMT dataset is used from where both the train and test data of the previous two approaches are merged here, and random samples of train ($\frac{3}{4}$) and test ($\frac{1}{4}$) are generated, where the test sets could have instances with internal *sandhi*, external *sandhi* and no *sandhi* at all. The optimality theory based approach is taken up with this posterior probability function.

The *saṃdhi*-splitter algorithm is based on the two-stage modelling framework proposed by Goldwater et al. (2006, 2005), where one stage (*generator*) corresponds to the generation of morphs likely to be found in a lexicon, from some probability distribution, and the other stage (*adaptor*) which determines how often each of these morphs occur.²⁸ In this case, the *generator* generates a super-set sequence of morphs: $M = M_1, M_2, \dots, M_n$ from a probability distribution P_s . The *adaptor* generates a sequence of integers: $Z = z_1, z_2, \dots, z_m$, each of which are identifiers of one particular item from M . The Chinese Restaurant Process (CRP) (Aldous, 1985) is used as the adaptor, and combined with a morph generator, the new model is described as $TwoStage(CRP(\alpha), P_s)$, where the generator P_s uses a unigram phoneme distri-

²⁷ The notation $\hat{P}$ is used instead of P because the true value can only be estimated from the training set, but can not be known with certainty.

²⁸ Since it handles both external *sandhi* and internal *sandhi*, both word boundaries and morph boundaries are considered here.

bution. Gibbs Sampling is used to sample from the posterior distribution of *Samdhi* analysis. Both an unsupervised and a supervised approaches are proposed.

The results show that with the proposed posterior probability function, the current approach with optimality theory outperforms the previous two optimality theories by a margin of 3.5% of F-Score. Four variations of the two-stage framework are proposed. In the first, the basic state of the framework is used where the algorithm possesses no linguistic knowledge. In the second, the sampling is done only when the generated segments do not violate the Sonority hierarchy.²⁹ In the third, a morphological analyser is used and sampling is done only when the segments obtained are morphologically valid. The fourth uses both the morphological analyser and the training data, and the algorithm follows a supervised approach. This supervised approach outperforms the remaining by a margin of 10 in terms of F-Score.

RNN for Joint Compound Splitting and Sandhi Resolution

Hellwig (2015b) interprets *sandhi* and compound resolution as a sequence labeling task, making use of a recurrent neural network with LSTM cells. The training data is extracted from the corpus of SanskritTagger (Hellwig, 2009). Since *sandhi* information is not available, each sentence is re-analyzed and the corresponding *sandhi* information is extracted from the analysis that matches the gold analysis of the sentence stored in the database. It splits a string into phonemes p (observed sequence) and each phoneme is associated with the desired type of transformation rule (target sequence). It defines two classes of *sandhi* rules based on whether the result of applying a *sandhi* rule is a single vowel (vocalic) or not (non-vocalic). It also proposes 5 rule types for inter-word *sandhi*, the distinctions based on (1) whether the phoneme changes from the observed to the target sequence, and (2) whether a word or compound split is inserted into the target sequence. Using these rules, the observed and target sequences are extracted to form the training dataset.

The aim of the learning algorithm is to split compounds³⁰ at correct positions, resolve *sandhi* and produce *sandhi* rules. The input layer received the phoneme at position t in a string in 1-of-n encoding. The output layer contains as many units as there

²⁹ Sonority hierarchy is as follows: Vowels > Semivowels > Nasals > Spirants > Voiced Stops > Unvoiced stops.

³⁰ Compounds here primarily refer to the sandhied chunk and also to the compound words (*samasta-pada*).

are target classes. The forward and backward hidden layers capture the left and right context of t , respectively, and LSTM cells are used in the hidden layers. The output layer receives outputs from the hidden layers and performs a softmax regression for the desired target values.

The results show that out of the five rule types, the rule indicating to keep the original phoneme, and the rule that refers to replacement with a single phoneme are identified correctly most of the times, resulting in a high F score. The remaining rules, which describe, either undoing a vocalic *sandhi* (like *caiva* $\rightarrow$ *ca-eva*) and adding a compound split (-), or undoing a non-vocalic *sandhi* (like *aśvaśca* $\rightarrow$ *aśvaḥ-ca*) and then adding the compound split, or simply inserting a compound split (like *mahāgiriḥ* $\rightarrow$ *mahā-giriḥ*), observe very less scores.

Segmentation using Path-Constrained Random Walks

Krishna et al. (2016) identified 350,000 sentences, from the DCS corpus, for segmentation and used the segments produced by SH and approached the selection of segments as a query expansion problem using Path Constrained Random Walks (PCRW) (Lao and Cohen, 2010) framework with linguistically motivated Inductive Logic Programming (ILP) formulations for finding the correct segmentation. The approach effectively combines the morphological features in addition to the word co-occurrence features as their context with minimal usage of morphological information.

An input sentence is passed to the SH platform, which produces the segments of all possible segmentation solutions, along with the morphological analysis of the segments. These segments are the candidates, and a graph is conceptualized with the candidates as the nodes. The most promising candidate is chosen as the initial query node and then PCRW is performed using a set of pre-defined path types, to produce a winner node among the candidates. All the candidates conflicting with the winner node are eliminated using positional information and *sandhi* rules. This process is continued until no more candidates are left to be evaluated.

A weighted multi-digraph $G(V, E, W)$ is formed from the possible segments, with the vertex set V representing a candidate. Every node has three attributes: word-form, lemma and POS tag (morphological information) of the given segment. For every non-conflicting pair of nodes, edges with varying edge weights from the attributes of a

source node to each of the attributes of a target node are formed, resulting in 9 edges between a pair of nodes. The edge weights are the co-occurrence probability of the attribute value at the target node, given the attribute value at source node.

The corpus is represented as another graph structure $G_2(V_2, E_2, W_2)$, where V_2 is the vertex set containing the union of vocabulary of the three attributes. Directed edges are formed between every pair of nodes that co-occur in a sentence in the corpus. The weight is then calculated as the ratio of total number of sentences in which the source and target nodes co-occur to the count of documents in which the source node occurs. DCS was used to build this graph as the training data.

A set of 9 constraints (termed as path types) which are linguistically motivated are considered as Inductive Logic Programming formulations. All possible paths are formed in the graph G which satisfy the constraints of the path types. The edge weights are populated from the corresponding paths from the graph G_2 . In case of multiple paths to be traversed from the source node to the target node, the product of the weights of the intermediate paths is taken as the weight of the path. One example of the path type is shown in the equation:

$$POS_i^{Noun} \xrightarrow{P(i|j)} Lemma_j^{Verb} \quad (2.6)$$

This indicates a path from a POS attribute of a source node which is a noun, to the lemma attribute of a target node which is a verb, and the weight denotes the probability of co-occurrence of the two parameters in the corpus. For some of the path types, a random walk over the graph G is done and all the nodes which are non-conflicting are marked with the paths and weights. For some of the path types, path scores are calculated based on random walk traversal over the graph G_2 . A weighted sum of paths is calculated and the winner node is selected.

The results over the test set of 2,148 strings used by Natarajan and Charniak (2011) show an overall improvement of 28.81% (from the previous approach) in terms of F Score. 83.05% of strings were segmented with an F-score of one. It further proposes three baselines: (1) longest word selection, which iteratively selects the longest segment eliminating the conflicting segments in each iteration, (2) greedy candidate selection, where the morphological analyser's output is considered as a tree, and a greedy selection is performed that maximises the overall likelihood of the selection, and (3)

unsupervised random walks with restart, where only some of the path types are considered which have the length of the path as one. 1,00,000 sentences from DCS were collected with 90,000 for the training and 10,000 as held out dataset. The supervised PCRW performs the best with a margin of 13.79% and 10.60% in precision and recall.

Sequence to Sequence Labeling

A purely engineering based approach is proposed by Reddy et al. (2018) where a deep sequence to sequence (seq2seq) model is considered. An encoder-decoder framework is used where the sandhied sentences are treated as the input to the encoder and the unsandhied sentences as the output of the decoder. The model is trained so as to predict the unsandhied sequence given its corresponding sandhied sequence. It only uses the parallel segmented and unsegmented sentences for training without considering the linguistic features like lexical and morphological analysis.

The input sequence is reversed first. The ‘*sentencepiece*’ model (Kudo and Richardson, 2018) is used to obtain a new vocabulary for the corpus, where new words are identified using a greedy approach that maximises likelihood of the language model. Three layers of LSTMs are used at both the encoder and the decoder. The input sentences (105,000) and the ground truth inflected forms were extracted from the SIGHUM dataset. Given the training set, the objective was to maximise the log probability of the segmented sequences T where the unsegmented sequences S are given. For a new sentence, the output should be a sequence T' with maximum likelihood for the given input. The model is experimented with and without attention for comparison.

The supervised PCRW described earlier and a structured prediction approach using graph based Conditional Random Fields were considered as the baseline. A test dataset of 4,200 sentences was considered for evaluation with string-wise macro-averaged precision, recall and F-score as the metrics. The seq2seq with attention outperforms the remaining models with a margin of 16.29% in F score. It also shows how the length of the sentence alters the results. For sentences with more than 10 words, the PCRW model performs better, and for sentences with less than 6 words, seq2seq with attention performs better.

Sequence to Sequence ($seq2seq^2$)

Aralikatte et al. (2018) proposed another sequence to sequence (Double Decoder RNN) model with two tasks: finding the split locations and then the individual splits. A deep bi-directional character RNN encoder and two decoders with attention ($seq2(seq)^2$) form the core architecture of the model. The two decoders added to a bi-directional encoder-decoder model were: (1) *location decoder* which learns to predict the split locations, and (2) *character decoder* which generates the split words. A compound word is fed into the encoder character by character. Each character's embedding is passed to the encoder's LSTM units which encode the word in both the forward and the backward directions. The encoded context vector is passed to the global attention layer. In the first phase of training, the location decoder is trained and the model learns to identify the split locations. In the second phase, the character decoder is trained where the decoder learns the underlying rules of *sandhi*. It uses as the context, the attention layer which is already trained to identify split locations.

The SHMT corpus and the SandhiKosh dataset were combined and heuristically pruned to get 71,747 words and their splits. This is used as the benchmark dataset. 80% of it was randomly sampled for training and the remaining for testing. A comparison is done with other publicly available tools like the UoH, SH and JNU segmenters, where the ground truth is checked in the top 10 predictions from these segmenters and compared with the standard RNN architectures along with the proposed Bi-directional DD RNN. This shows a significant difference between the performances, where the B-DD-RNN outperforms all the segmenters. For a comparison with other architectures of RNN, a unidirectional encoder decoder without attention, a bi-directional encoder-decoder with and without attention were considered. The accuracy on the benchmark dataset for split location prediction is 95% and for split words prediction is 79.5% where the accuracy of split prediction is 14.7% more than the next best. The $seq2seq$ mentioned earlier is outperformed by the current model by approximately 6.47%.

rcNN

Hellwig and Nehrdich (2018) released a revised DCS dataset exclusively for the segmentation task and performed the segmentation using a character-based recurrent and convolutional neural network model, that works well with just the parallel corpus of

unsegmented-segmented sentences. This is referred further as rcNN. It uses the DCS-segmentation dataset described in section 2.2.4. The input characters are initialized with uniform random values from $[-1, +1]$ and updated during training. During the training, when a split rule is encountered, its left and right context (character n-grams with lengths $n \in [2, 7]$) are extracted and a vector (split probability) for the left context with length n is calculated as the ratio of the count of the left context specifically from the observed character, to the total number of the context from all characters. Similarly, the vector is calculated for the right context.

The baselines chosen are the Bi-directional RNN described earlier applying it over the entire sentence instead of chunks of isolated strings, the seq2seq model retrained with the DCS-segmentation dataset and a Transformer (Vaswani et al., 2017) architecture for the input preprocessed with *sentencepiece*.

The idea behind incorporating the convolutional element is that a combination of the convolution and the recurrent elements are effective, where complex local features are extracted by the convolutional element and then considered in larger context by the recurrent element. Three models are proposed: (1) crNN, where a convolutional element is applied to the character embeddings. Its outputs are fed to a bidirectional recurrent layer, (2) rcNN, where the order of convolutional and recurrent layers is switched, and (3) rcNN_{short}, where shortcut connections are introduced that concatenate character embeddings and the RNN output with the feature maps.

A comparison with the seq2seq models showed that rcNN performed on par with them taking significantly less time for training and inference. A comparison with external models like Bidirectional RNN, seq2seq and Transformer is presented where the rcNN performs the best, edging the Transformer model with less than 0.3% in all the measures. A comparison is done between crNN, rcNN and rcNN_{short} with and without split probabilities, where the rcNN_{short} with split probabilities outperforms the remaining.

Another important observation regarding the compositionality of compounds is noted. Upon testing on a domain specific Buddhist treatise *Triṃśikāviñaptibhāṣya* and the philosophical text *Nyāyamañjarī*, the performance of both rcNN and Transformer observes a significant drop, predominantly due to disagreement about the compositional or non-compositional analysis of technical compounds.

Energy-based model

Krishna et al. (2018) worked on a structured prediction framework and addressed word segmentation along with morphological tagging as a joint task using an energy-based model. The previously released SIGHUM dataset was used here. The segments produced by SH form the nodes of the graph, $G(V, E)$, and directed edges in both the directions are constructed between nodes that can co-occur in at least one segmentation solution (called as *exhaustive segmentation* in the paper). The task is to search for the minimum cost maximal clique on G . If a single node contains multiple morphological analyses (due to homonymy and syncretism), these individual analyses are considered as separate nodes to have a fine grained approach. Learning consists of finding an energy function that associates lower energies to cliques with increasing similarity to the correct clique. For maximum clique selection, a greedy heuristic approach inspired by Prim’s algorithm (Prim, 1957) is used. The algorithm starts with a single node. A vertex v is added 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, and discarding all the nodes that are conflicting with v . The maximal clique is obtained when there exists no more vertices to loop through. The same is performed starting from every node in G , and the maximal clique with the least score is selected from all the cliques produced.

From a morphologically tagged corpus, morphological constraints are prepared to condition the distributional information between the candidate nodes of an edge. Features are generated to capture this distributional information, using the cooccurrence of the nodes in the corpus. The morphological constraints are a set of grammatical category combinations, which can be either *complete* (for example, ‘genitive-masculine-singular’) or *partial* (for example, ‘genitive-masculine’). The feature generation and selection are approached using the Path Ranking Algorithm (Lao and Cohen, 2010).

Supervised PCRW, EdgeGraphCRF, seq2seq and three variations of EBM, namely Lattice-EBM, Tree-EBM and Clique-EBM are considered for experiments. The evaluation is done on two tasks: Word Prediction Task which is a word segmentation task, evaluated based on the correctness of the inflected word forms predicted; Word++ Prediction Task where the joint task of segmentation and morphological prediction is evaluated.

The 350,000 sentences of the PCRW model is used as the corpus for generating the edge vectors and the test set of SIGHUM is used as the test set for the second task. For the first task, the test set of seq2seq is taken as the test set. The evaluation is done on Precision, Recall, F Score and Perfect Matching. Clique-EBM outperformed all the models for the first task, across the four metrics with an improvement of 7.06% in F Score over the seq2seq model and by 6.05% over the next best (Tree-EBM) model. For the second task, it outperformed the next best (Tree-EBM) model by 8.57% in F Score.

Neural Compound word Sandhi Splitting

Dave et al. (2021) proposed an RNN based two-stage deep learning method for segmentation of isolated compound words without using any lexical or sentence information. It used the SHMT dataset and formulated the segmentation problem in two stages: predicting the *Sandhi* window in stage 1, using an RNN model and splitting the *Sandhi* window in stage 2 using a seq2seq model. In the first stage, the input sequence is a compound word and the target output is an array of integers, whose length is the same as that of the input sequence. All elements in this output are 0 except for those in the window.³¹ An RNN was used with bidirectional LSTM as RNN cells. It produces a sequence of real numbers between 0 and 1 with the size of the sequence equal to the length of the input sequence. In the stage 2, the input sequence is the *sandhi* window of the compound word, output sequence is set as truncated words T_{W_1} and T_{W_2} . The best results were obtained with LSTM as basic RNN cell for decoder and a bidirectional LSTM as basic RNN cell for encoder. The compound word C_W and the location of the *sandhi* window S_W pairs form the data for stage 1. S_W , T_{W_1} and T_{W_2} form the data for stage 2. For the location prediction accuracy when compared to the seq2seq model, the latter performs better by a margin of 2.7% in terms of accuracy, but for the split prediction task, the former performs better by a margin of 7.3%.

TransLIST

Sandhan et al. (2022) proposed a Transformer-based Linguistically Informed Sanskrit Tokenizer (TransLIST) for the task of segmentation and is the current state-of-the-art system. For this setup, TransLIST uses the SIGHUM dataset and the Hackathon dataset

³¹ The window size is fixed as 5.

for a fair and an exhaustive comparison with all the previously built word segmenters for Sanskrit (including rcNN). Hackathon dataset refers to a partially released dataset of the present work which was released for the purpose of Word Segmentation and Morphological Parsing Hackathon.

It consists of three modules: (1) a module that encodes the character input along with latent-word information, taking into account the *sandhi* phenomenon, (2) a novel soft-masked attention to prioritize potential candidate words and (3) a novel path ranking algorithm to rectify the corrupted predictions. Thus, it formulates Sanskrit word segmentation as a character-level sequence labelling, integrated with latent word information from SH. This soft masked attention helps in predicting the words but sometimes the words might not be part of the candidate solution space. So, with the help of SH’s possible segments, the predicted words are appropriately substituted to the suitable candidates.

The LIST module first receives candidate space solutions from SH if available, or it resorts to using all possible n -grams which helps to add inductive bias about neighbouring candidates in the window size of 4. The candidate segments or the n -grams are fed to the Transformer encoder and the classification head learns to predict the gold standard output. The novel soft-masked attention allows interactions between the candidate/characters, and also prioritizes the candidate words containing the input character for which the prediction is being made. The third module PCRP addresses the mistakes where the predicted segments are not part of the candidate segments, by appropriately substituting suitable candidates.

It was observed that the rcNN performed the best amongst the baselines, TransLIST model outperforms all of the baselines taken into consideration, by a margin of approximately 2%. It also showed the importance of LIST module (which comprises of *sandhi* rules, Sanskrit vocabulary, n -grams and SH’s segments) without which the overall performance, in terms of Precision, Recall and F-score, is affected by a reduction of upto 5%.

Some limitations of the existing models

First, while some of the models are purely data-driven and do not consider any linguistic feature, some provide segmentation for compounds alone, some detect the split

locations, some of them consider all of these but operate on either a limited dataset or a dataset which is not normalized. Second, compound components are addressed similar to words in most of the models, which introduces ambiguities during evaluation. While the *sandhi* rules are much the same for both *sandhi* across words as well as *sandhi* within a compound, detecting a compound word in this stage of segmentation becomes crucial in downstream NLP tasks. While almost all models resolve sandhi within a compound word, they don't mark the boundaries of its components. Third, handling Out Of Vocabulary words is still yet to be explored at a relatively larger scale when compared with other languages, except for the preliminary effort towards guessing the *prātipadika* (stem / root) of noun forms.

2.4 Sanskrit Heritage Segmenter (SH)

The Sanskrit Heritage Engine is a platform that hosts a lexicon (The Sanskrit Heritage Dictionary) and various tools like segmenter, lemmatizer, declension and conjugation engines. The segmenter (referred to as SH in the paper) analyses any given text and segments it into all possible splits and displays them in a graphical interface where the user has the option to choose the required split. An example is shown in figure 2.1.

Figure 2.1: Sanskrit Heritage Segmenter's Graphical Interface

dharmakṣetre	kurukṣetre	samavetā	yuyutsavaḥ	māmakāḥ	pāṇḍavāścaiva	kimakurvata	sañjaya					
dharma	kṣetre	kuru	kṣetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	kim	akurvata	sañjaya	
✓X	✓X	✓X	✓X	✓X	✓X	✓X	✓	✓	✓	✓	✓X	
dharma	kṣetre	kuru	kṣetre	samavetā	yuyutsavaḥ	māmakāḥ		aiva			sañjaya	
✓X	✓X	✓X	✓X	✓X	✓X	✓X		✓X			✓X	
	kṣetre		kṣetre	sama	vetāḥ	māma	kāḥ	eva			san	jaya
	✓X		✓X	✓X	✓X	✓X	✓X	✓X			✓X	✓X
	kṣetre		kṣetre		vetā	mā	kāḥ					jaya
	✓X		✓X	✓X		mā	kāḥ					✓X
						✓X	✓X					
						āma						
						✓X						
						ama						
						✓X						
						āma						
						✓X						

It closely follows *Pāṇini*'s system i.e., all the rules governing the concept of *sandhi* that occur in *Aṣṭādhyāyī* are taken into consideration. This segmentation is lexicon directed, using forms systematically generated from its own lexicon. It combines a fast segmentation algorithm using finite-state transducers and dynamic programming

with a first-pass of chunking that limits the inherently exponential complexity to small-length chunks, making the whole segmentation analysis fast enough in practice to be usable interactively.

2.4.1 SH methodology

SH operations involve three stages, namely Chunking (which includes normalization), Segmentation and representation of the results. These are explained in detail ahead.

1: Chunking: As a part of preprocessing, the sentence is initially divided into chunks based on certain rules,³² some of the default spaces provided in the input sentence or the *avagraha*. This reduces the segmentation task into chunk-level segmentation and increases exponentially the speed of the segmentation process. Additionally, two kinds of normalization is done. One, the *anusvāra* is converted to the homophonic nasal consonant corresponding to the following character. For example, *saṁdhi* to *sandhi*. Two, when the duplication in some of the conjunct consonants (as in *sattva*) is presented without the duplication (*satva*), these are handled by accepting both the forms with an entry for either of the lemmas in the lexicon.

2: Segmentation: Each chunk is analysed further where segmentation and morphology recognition are done parallelly, using finite state transducers and dynamic programming. Morphologically valid segments are collected for final representation of the segmented results.

“There exists a finite number of solutions to the inverse of junction transduction, a special case of transducers definable by finite-state automata under which falls external sandhi” (Huet, 2009) forms the basis for the segmentation mechanism in SH.

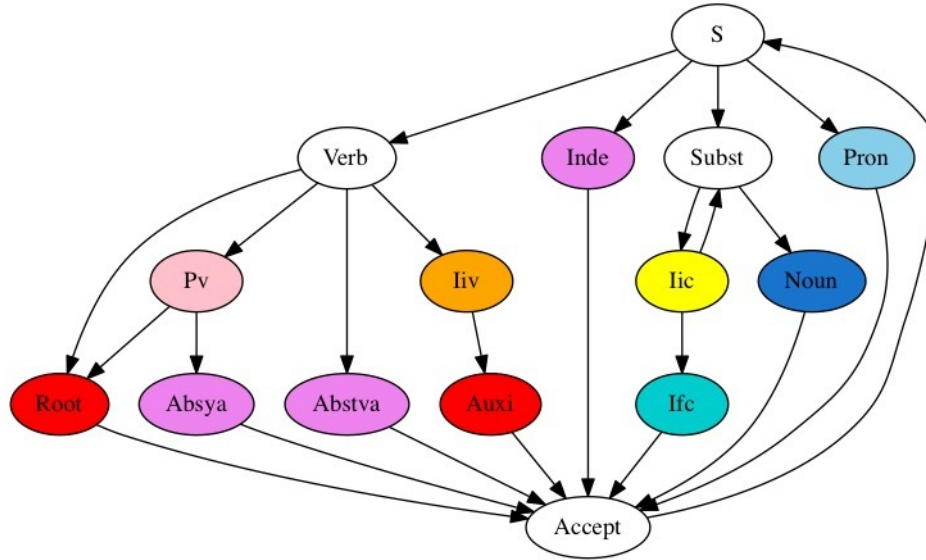
SH has two dictionaries: (a) Monier-Williams (Sanskrit-English) and (b) Heritage (Sanskrit-French). While the former has been obtained from the XML version available online, the latter is being currently developed, taking entries mostly from MW, but also having new entries depending upon the requirement of the segmentation engine.³³ Additionally, it also has multiple contextual information for some of the entries. Either of these dictionaries can be used for the segmentation task.

³² One such example is *mo'nusvārah* (Aṣṭ. 8.3.23), which states that the *m* at the end of a *pada* when followed by a consonant, is replaced by *anusvāra* (*m̐*) in the context of *saṁhitā*. Thus if an *anusvāra* is observed at the end of a word (viz. followed by a space), we can assume the chunk boundary here.

³³ Some of the compounds which require a non-compositional meaning are lexicalized.

Entries or *lemmas* (stems and roots) from the dictionaries are stored in a decorated trie structure to form the base lexicon (Huet, 2005a). While the trie structure encodes the lemmas, the grammatical information (like the POS role, gender/number for substantives, valency and other sub-categorization information for verbs, etc.) are stored as decorations on the lemmas. The grammatical information represented by the inflections (or the suffixes) of a lemma are invariably *morphisms* on the lemma (or the subtree corresponding to the position of the lemma). Finally, a structured lexicon (called *lexmap*) is formed by inflecting the lemmas with all possible suffixes using the internal *sandhi* rules. Each of the categories like Noun, Verb, Iic, etc. have dedicated tries, from which such lexmaps are obtained using the same process. These inflected forms tries are used as deterministic skeletons of a non-deterministic finite-state transducer, by decorating the transducer with rewrite rules. Rewrite rules are typically the *sandhi* rules of the form represented in equation 2.1. Thus a regular transducer compiler is used to produce a finite state transducer from this structure, and this FST (decorated with rewrite rules) along with a morphological tagger form the core elements of the segmentation mechanism. For each of the categories (or phases), a lexmap and a transducer are built and used for the segmentation. Figure 2.2 depicts the top-down view of the control graph of a simplified version of SH's phases. These phases form the states of the automaton.

Figure 2.2: The 10-phase lexical analyzer



A sentence S is understood as a sequence of words W , which can be either Substantives, Verbs or Invariable forms. A Substantive is either an inflected Noun (*rāmaḥ*) or a sequence of Iics with a terminal (inflected) Ifc (*tapah-svādhyāya-niratam*). A Verb can be an inflected form of a Root (*gacchati*) or a possible combination of a sequence of preverbs and a Root (*sam-ā-gacchati*), or a composite form of an auxiliary root finite form (*śuklī-karoti*). The Invariables can be regular indeclinable forms (mostly available as a list defined in *Aṣṭādhyāyī* - *ca*, *api*, etc.), or generated from the forms from dictionary by adding the absolutive and infinitive suffixes (*gam + ktvā = gatvā*, and *gam + tumun = gantum*, respectively). This is a simplified version considering only the inflectional morphological analysis. A detailed structure incorporating finer morphological categories (40) including the derivational morphological analysis can be observed in Huet (2024).

Given chunks of normalized text from the input string, the algorithm runs through the different FSTs, for each chunk, keeping a stack of the non-deterministic choices and an additional stack for storing the partial (in essence the cumulative) output. It simultaneously extracts the morphological information decorated in the trie from the output. The choices thus encompass the segment, its corresponding rewrite rule, the morphological information associated with it, and the phase information. These are sent to display routines for presenting these results as described ahead.

3. Representation: SH has two representation modes - Reader, where the segmentation results are produced in a list, and Summary, where all the segments are populated in an interactive graph where users can choose or reject a segment.

Reader mode: The Reader mode computes all possible segmentation solutions for a given text, and ranks them by assigning penalties based on shallow parsing based on *kāraka* analysis. There is also a filtering mechanism which applies this ranking mechanism and prunes out non-sensical solutions (Huet, 2007). This shallow parsing identifies the verb and searches for thematic roles for the verb to be meaningful for denoting a situation or an action. For example, a transitive verb has the expectancy of an Agent (*kartā*) and a Patient (*karma*). Depending on the voice of the verb, i.e., if active, these are realized by a substantive form (in nominative and accusative cases respectively) and if passive, an Instrumental is needed for the Agent. Similar basic *kāraka* rules are encoded. When applying all the rules based on the verb, if certain

roles are missing or if there are extra nominatives which do not agree in number with any other nominative in the sentence (for adjectives), penalties are assigned. Since we find multiple morphological tags for a segment,³⁴ an integer penalty is computed with each tag and the minimum of all such penalties across all the segments forms the compound penalty of the segmentation. Finally the segmentations are ranked based on the low penalty and less number of segments.

For each segmentation solution, it produces the segments, their possible part-of-speech and lexical analyses, and the subsequent *sandhi* (called transition) that happens at the word/component boundaries. Figure 2.3 shows an example of the reader mode for the unsegmented sentence *rāmovanaṅgacchati*. The expected segments are *rāmaḥ*, *vanam*, *gacchati* and the preferable solution would mean: Rām (*rāmaḥ*) goes to (*gacchati*) forest (*vanam*).

This semantic analysis based ranking is extremely naive, assuming that the order of words in a Sanskrit sentence is irrelevant to its meaning. Thus, this is applicable to very simple sentences, without subordinate verbal phrases.

Summary mode: The segmentation combined with the ranking in the Reader mode increases the time and space complexity. This includes the fact that listing out all the segmentations for long and complex sentences, especially with ellipses and other anaphoric or discourse operators where dependencies are context-sensitive, with the mechanism and its representation as a web-service resulted into very long webpages, in some cases eventually leading to choking of the server. It was also not trivial for deploying any sharing mechanism on the list of segmentations. Goyal and Huet (2016) came up with a lean interface to solve this problem where the key idea was to represent an abstraction of the union of all segmentations, re-aligned on the input utterance, where the interface is amenable to sharing and represented compactly on the screen.

The segmentation results are fed to a graphical interface (called Summary mode), where all the possible segments (extracted from all possible segmentation solutions) are populated on this interface along with options to choose or reject the segments (refer figure 2.1). Upon selection or rejection of a segment, the interface updates itself dynamically by respectively choosing or pruning out the possibilities associated with the segment, effectively reducing the search space.

³⁴ due to syncretism and homonymy

Figure 2.3: SH Reader analysis of the sentence *rāmovanaṅgacchati*. Only the first two solutions are presented here for clarity.



If s is the input sequence, the segmentation mechanism computes the set of all possible segmentation solutions $S = Segs(s)$, and displays $D(S)$ which is the *tabulated display*, defined as a set of aligned segments obtained as the union of all its corresponding aligned segment collections. Every aligned segment (k, z) is represented as the segment z displayed with an offset k spaces from the left margin. Overlapping segments are displayed in different (successive) rows. Longer segments are listed above shorter ones. Each segment is represented with the following features:

- word
- stem

- inflectional morphological analysis
- base
- derivational morphological analysis
- *phase*: indicated by colours of the segments, each colour corresponding to a part of speech
- link to the lexicon entry for each stem and base

Thus, they can be considered as *aligned lemmatised segment* $(k, (l, z))$, where l (lemmatization) contains all the above mentioned features. Additional features are available like choices to user for selection or rejection of segments, an undo option to revert back the choice, redirection to the Reader mode to enlist the solutions, and to a filtering mechanism based on the semantic constraints mentioned earlier, and also a pipeline to the *Samsādhani*³⁵ platform's dependency parser (Huet and Kulkarni, 2014).

This fine-grained analysis helps the user to disambiguate and correctly pick the intended split and prune the solutions that are not required. Such information also helps in the further stages of sentential analysis like parsing, disambiguation, and discourse analysis.

2.4.2 Observations on SH

A major advantage of SH is its ability to address both segmentation and morphological analysis simultaneously. The usage of finite state automata along with the preprocessing of chunking makes it faster as well. We can observe various features of the segments produced (unsandhied word-form, lemma, morphological analysis, etc.), which help in identifying the correct segments. These along with the interactive summary mode is useful for any reader of Sanskrit. SH also proposes a new set of Part-of-speech tags (called *phases*) and their relevance from the perspectives of *Pāṇini* is a topic of further research. For example, the way it handles derivation in *kṛdantas* and splitting the compound components and assigning each of the components a separate category, etc. We can also observe categories which encompass multiple inflections of the same

³⁵ <https://sanskrit.uohyd.ac.in/scl/>

stem, and at the same time, some inflections are assigned an altogether a different category. For example, the nominative and accusative cases of the stem *vana* have the word form *vanam* and are considered under a single phase, while the vocatives are allotted a separate category. Some of these phases help the reader further to identify whether a particular segment is a part of an *avyayī-bhāva* compound, and also if it is a part of *bahuvrīhi* compound.

On the other hand, there are a few limitations of SH. One, the system does not consider *yogyatā* (meaning compatibility) in deciding the correct segmentation. Two, it handles Out Of Vocabulary words semi-automatically. For the unrecognized words, the interface allows the user to suggest the lemma (provided it is available in the Monier Williams Dictionary (Monier-Williams, 1899)), which is then stored in the local dictionary for future use. The Reader mode does handle the over-generation problem to a certain extent by ranking the solutions based on a shallow parsing algorithm (Huet, 2007), but this is a primitive ranking mechanism and can be applied only on short sentences and sentences devoid of large compound words. Three, Sanskrit's nature to build words by successive combinations or derivations increases the time complexity of the engine. Sentences with large number of compounds each having complex *sandhi* possibilities,³⁶ are always hard to process. Four, SH is completely deployed as a web application, built predominantly using Ocaml (Objective Caml). We do not have an option to extract the results of SH, except for scraping through the webpage results. These results, if available in a standard format, can then be piped with various kinds of approaches to solve the segmentation problem. These can also act as auxiliary information to other tasks like dependency analysis. Scraping the webpage has many disadvantages, mainly the inapplicability of the scraping mechanism with changes to the webpage.

SH works well as a reader companion, where a user figures out the intended segmentation and morphological analyses of the segments by choosing or rejecting from available set of segments. For example, let us consider the analysis in Figure 2.1. There are 41 segments in total, and to produce the correct solution, one has to select 9 segments out of the 13 words in the sentence. With a longer sentence, the number of selections increase. And for sentences with huge compounds and multiple *sandhi* oc-

³⁶ Here, the complexity is measured in terms of the number of *sandhi* possibilities, given a split location.

currences within a chunk, the interface becomes huge making it difficult for readers to go through. One can think of automating the process of selection and rejection without a human intervention. To automate the process of such selection and rejection, the solutions have to be prioritized. The Reader mode of SH is the first attempt to enlist all the solutions but it is done only if the total number of solutions is less than 100. It uses a dovetailing mechanism after obtaining all the solutions along with shallow dependency analysis. A ranking mechanism is thus required to prioritize the solutions obtained which will be applicable for all kinds of sentences, and also faster than the Reader. Automating the selection and rejection process has two advantages: One, it preserves all the features that come along with a segment. Two, in this process, a standalone implementation of segmentation is achieved using the results of SH.

Krishna et al. (2016) worked on something similar, where they relied upon the results of SH and proposed a path traversal algorithm to obtain the segmentation solution. Similarly, we can come up with good prioritization techniques to rank the solutions arriving at the intended solution in less time.

2.5 Necessity for aligning DCS and SH analysis

The ranking techniques mostly require a sufficient amount of training data compatible with the analysis of SH. While DCS has a huge annotated dataset, it does not contain the segments and does not explicitly mark the compound boundaries. SH's analyses which contain the segmented forms and also the compound boundaries marked, in addition to the lemma and morphological analysis, can then be aligned with the analysis from DCS for all the DCS sentences. This results into a unified dataset useful not just for the ranking mechanism, but also for various other approaches to handle the segmentation problem.

DCS has the ground truth analysis, with lemma, Universal Part-of-speech tag, morphological analysis, and links to its lexicon. SH produces the segment, detailed morphological information (both inflectional and derivational), Part-of-speech tag from a novel set of POS tags and a link to its lexicon. SH produces all possible segmentation solutions along with all these information for each of the segments of the solutions. To prepare a dataset with sufficient annotated information, we can leverage the various

features produced by SH by comparing the lemma and morphological analysis of each of the segments with the ground truth lemma and morphological analysis from DCS. This ensures that the correct analysis is chosen along with more linguistic features annotated in the resultant dataset.

The design decisions of DCS and SH differ for various parameters, and can be observed in their representation and nomenclature. While a one-to-one mapping of some of the nomenclatures is trivial, we can find differences in the representation structures of the two systems which lead to a one-to-many mapping.³⁷ This necessitates a comparison between the representations of the two systems prior to the process of alignment, to ensure these are converted to a format inclusive of analysis from both the systems. Krishna et al. (2017) provided details of the differences between the two systems along with the methodology to align the analyses of the two systems, creating a dataset of 115,000 sentences, which is described ahead.

2.5.1 SIGHUM Alignment

DCS’ analyses for each of the sentences were represented as objects containing the sentence details like chunks, lemmas, and CNG values. A glimpse of what an object looks like is depicted in table 2.2.

Table 2.2: An example DCS Object data	
Sentence Id	→ 1
Sentence	→ <i>pañcaratnāni mukhyāni coparatnacatuṣṭayam</i>
Chunks	→ [<i>pañcan</i> , <i>ratna</i> , <i>mukhya</i> , <i>ca</i> , <i>uparatna</i> , <i>catuṣṭaya</i>]
Lemmas	→ [[<i>pañcan</i>], [<i>ratna</i>], [<i>mukhya</i>], [<i>ca</i>], [<i>uparatna</i>], [<i>catuṣṭaya</i>]]
Morphological Class (CNG)	→ [[<i>“41”</i>], [<i>“41”</i>], [<i>“41”</i>], [<i>“2”</i> , <i>“3”</i> , <i>“31”</i>]]

To match the two systems, data from SH’s analysis was scraped and certain parameters such as word, lemma, position, morphological information, chunk number, word length, and pre-verbs were extracted. Corresponding to the morphological analysis the CNG value was generated for the ease of alignment. With all these parameters

³⁷ This is mostly from DCS to SH, as SH provides additional information like derivation of conjugation, class and *pada* information.

as attributes of the nodes, graphs were built for all the sentences. Standard graph processing libraries (Leskovec and Sosič, 2016; Hagberg et al., 2008) were considered for extracting data from these graphs.

The XML based GraphML format was used to represent the candidate space segments. The GraphML files consist of graph structures, $G(V, E)$ as the representation for the analyses of a sentence. The nodes, V , are the possible splits, and the values in the edges, E , denote whether the participating nodes can co-exist in a solution or not i.e., whether or not they have an overlap in the position relative to the sentence, and that the overlapped portion does not follow any *sandhi* rule. The attributes of a node include the word (final inflected form), lemma (stem or root), morphological information, CNG value of the morphological information (this is to cross-check with the DCS' CNG values), chunk number, word position (relative to the chunk), word length and pre-verbs.

The two main parameters considered for aligning DCS and SH were *lemma* and *CNG value*. For every segment of DCS, a comparison was done with each of the nodes from SH, until a perfect match is obtained. Since there are differences in design decisions between DCS and SH, the candidate segments provided by SH had to be adapted and a few additional segments were added so as to match the entries in DCS. There were multiple issues while aligning the segments, discussed by Krishna et al. (2017) along with their respective solutions, a brief account of which is presented ahead.

Phonetic Variations: Inconsistencies while dealing with *anusvāras* (nasal consonant) which should have actually been *anunāsikas* (homonasal consonants), were normalized. While SH's normalisation handles this, DCS implicitly stores the nasal consonant and not the homonasal variant, hence normalized. For example, *śaṃkara* to *śaṅkara*. SH's stems are disjoint with the preverbs while DCS' stems are sandhied with the preverbs. Additional nodes were added with preverbed stems to handle this mismatch. For example, consider the word *praṇamāmi* (I bow), SH annotates the stem as *pra-nam* which is changed to *praṇam* by doing the *sandhi* between *pra* and *nam* which also introduces a phonetic variation on the character *n* (to *ṇ*).

Compounds and Named Entities: DCS' analyses is context-dependent and hence it prefers non-compositional analysis of compounds wherever necessary. SH produces the non-compositional analysis only when it is available in its lexicon. In other cases,

additional nodes were generated by sandhi-ing the components to match with the non-compositional analysis of DCS.

Secondary Derivative Affixes: Some secondary derivative affixes like *vat*, *tva*, etc. are treated differently by the two systems. SH keeps these morphemes as part of the root word, and DCS treats them as separate words. For example, the word *śighratvāt* is analysed by DCS as two components with the lemmas *śighra* and *tva* with the inflectional analysis pertaining to *tva*. SH sticks to the *Pāṇinian* methodology and addressess this with the lemma as *śighratva* and inflectional analysis on the whole lemma. This is taken care during the alignment process.

Lemma markers: DCS marks the transformed base form as the lemma for some of the non-final components of compounds (like *mahā* in *mahādeva*), while SH maintains the convention and presents the lemma as the original base (in this case, *mahat*). These are also resolved during the alignment process.

2.5.2 Why to align again?

The previous effort handled the major inconsistencies of the two systems during the alignment. Both the systems evolved in the recent years: DCS added more data and represented this annotated dataset in a new format (CoNLL-U) with additional features like dependency relations and word senses, and SH saw a significant development from the perspective of both segmentation and lexicon. Its lexicon is being populated with named entities and entries that were previously unrecognized. Another modification was regarding the pre-verbs. Earlier only the pre-verbs with derivational lemmas were joined, but in the current version even the inflectional lemmas also have the pre-verbs attached alongwith. Additionally, the necessary linguistic insights were not effectively utilised by the previous alignment approach.³⁸

A re-alignment is thus attempted in the present work to incorporate all these changes, to prepare a detailed documentation on the differences between the two systems, which further helps us in the alignment process, resulting into a unified dataset with as many features as possible.

³⁸ The codebase for the SIGHUM alignment was also unavailable which necessitated to work on a new alignment from scratch, although the insights were helpful during the new alignment.

The dataset's applicability is also to be considered, for which SH has been feeded with an additional layer of ranking, where the ranking mechanism incorporates the statistics from the unified dataset and narrows down to the most probable solutions. The present work thus focuses on two aspects: alignment and generation of a new unified dataset rich in lexical and morphological information from both DCS and SH, and also on developing the SH to produce the best possible solution(s) from the list of available solutions.

Chapter 3

Overcoming Linguistic Issues in Alignment

DCS acts as an annotated gold corpus for more than 250 texts. While the recent DCS annotations adhere to the CoNLL-U format, including the word, lemma, POS tag and morphological analysis of the word, it also has redirections to its lexicon and word-sense dictionaries. The morphological analyses are represented using various keywords (categories) according to morphological structure of words in Sanskrit. For example, nouns are addressed using the categories “Case”, “Number” and “Gender”. The SQL version of DCS generates an integer value (CNG) that encodes this analysis. There is also a clear distinction between the analysis of inflected words and that of derived words in both the versions. A negative integer denotes *tiṇanta* and *kṛdanta*, and a positive integer denotes *subanta*. But a major limitation corresponds to the missing surface forms of the words in the annotations. In the recent developments some of them have been reconstructed from their corresponding stem and morphological analysis. However, an exact one-one mapping of the stem-morphological analysis pair to the surface word-form has not been achieved due to various reasons. In this chapter, we will look into these reasons, specifically while trying to align the DCS annotations with the annotations proposed by SH.

While DCS’ main intention is to annotate a gold corpus useful for segmentation, morphological parsing, POS tagging and word sense disambiguation, SH primarily focuses on producing the possible segmentations along with the morphological tagging of each of the segments. The nature of both DCS’ and SH’s annotations is static¹ and SH’s results can be used for annotations of new corpora. SH is designed to be used as a reading companion to know all the possible segmentations and morphological anal-

¹ Given a word in a context, we get exactly the same result every time, unless there is a change in the core layer of DCS’ database, or SH’s lexicon or its segmentation mechanism.

yses of a given input text. DCS annotations are according to the context for all the input texts.

Both DCS and SH have approached their representations from the perspective of Western (or general) Linguistic terminologies. In some cases, both have kept in mind Sanskrit's nature of *Vṛttis* (derivations like *kṛdanta*). But from the perspective of Traditional grammarians' representation of the morphological analysis, a mapping has to be done between the structure and nomenclatures of tradition with those in the linguistics. SH's representation structure is considered as the base in the present work since it is quite similar to that of the tradition. A super-imposition of DCS' representation over SH's representation is attempted to understand the differences. Mapping the nouns and pronouns was trivial. But for verbs, SH represents using the *lakāra* system of the tradition which has to be mapped to the tense-mood pair of DCS, and similarly for finite and non-finite verbal forms. The *taddhitānta* (secondary derivatives) are seldomly analysed by both. And there are more differences than the similarities in the way the two systems analyse compounds. We will look into more such differences between such design decisions of DCS and SH in detail in this chapter. The DCS' CoNLL-U version is considered predominantly for the comparison as DCS has shifted to this representation. However, occasionally the SQL representation will also be referred to, especially the CNG value, since the comparison was started with the SQL representation.

3.1 DCS-SH differences

This section focuses on the issues in the representations at various levels such as sentence, chunk, word, stem, morphological analysis and compound, which hinder the process of alignment.

3.1.1 Marking Sentence Boundaries

SH does not restrict itself to any sentence-level boundaries, since it provides all possible segmentations and morphological analyses for any given text, which could be a sentence or a verse or a sandhied chunk or even a word. On the other hand, DCS has sentence wise annotations. But marking the sentence boundaries has not been

uniform in DCS’ approach. While sentences from most of the prose texts have been annotated as DCS’ sentences, for most texts of poetry, the hemistichs of the same verse are documented as separate sentences even when they correspond to the same sentence. For example, figure 3.1 shows the annotation of the verse 1.2 from *Bhagavad Gītā*. In some cases, a sentence might run across multiple verses where DCS marks each of their hemistichs as separate entries.

Figure 3.1: Partial representation of DCS’ CoNLL-U annotation

```
# text = samjaya uvāca
# sent_id = 230305
# sent_counter = 2
# sent_subcounter = 1
1      samjaya      samjaya      NOUN      Case=Nom|Gender=Masc|Number=Sing      -
2      uvāca      vac      VERB      Tense=Past|Mood=Ind|Person=3|Number=Sing      -
8613

# text = dr̥ṣṭvā tu pāṇḍavānikam vyūḍham duryodhanastadā
# sent_id = 230306
# sent_counter = 2
# sent_subcounter = 2
1      dr̥ṣṭvā      dr̥ṣ      VERB      VerbForm=Conv      -      -      -      -      -      -      -      -      -      -
2      tu      tu      PART      -      -      -      -      -      -      -      -      -      -
3-4    pāṇḍavānikam      -      -      -      -      -      -      -      -      -      -      -      -
3      pāṇḍava      pāṇḍava      NOUN      Case=Cpd      -      -      -      -      -      -      -      -      -
4      anikam      anika      NOUN      Case=Acc|Gender=Neut|Number=Sing      -      -      -      -      -      -      -      -
5      vyūḍham      vyūh      VERB      Case=Acc|Gender=Neut|Number=Sing|VerbForm=Part      -
27
6-7    duryodhanastadā      -      -      -      -      -      -      -      -      -      -      -      -
6      duryodhanah      duryodhana      NOUN      Case=Nom|Gender=Masc|Number=Sing      -      -      -      -      -      -      -      -
7      tadā      tadā      ADV      -      -      -      -      -      -      -      -      -
      LemmaId=96104|OccId=3607294|

# text = ācāryamupasaṃgama rājā vacanamabravit
# sent_id = 230307
# sent_counter = 2
# sent_subcounter = 3
1-2    ācāryamupasaṃgama      -      -      -      -      -      -      -      -      -      -      -      -
1      ācāryam      ācārya      NOUN      Case=Acc|Gender=Masc|Number=Sing      -      -      -      -      -      -      -      -
2      upasaṃgama      upasaṃgam      VERB      VerbForm=Conv      -      -      -      -      -      -      -      -
3      rājā      rājan      NOUN      Case=Nom|Gender=Masc|Number=Sing      -      -      -      -      -      -      -      -
4-5    vacanamabravit      -      -      -      -      -      -      -      -      -      -      -      -
4      vacanam      vacana      NOUN      Case=Acc|Gender=Neut|Number=Sing      -      -      -      -      -      -      -      -
5      abravīt      brū      VERB      Tense=Impf|Mood=Ind|Person=3|Number=Sing      -
|WordSem=88742
```

We can observe that the metadata of a sentence contains the sentence string, its ID,² and two additional parameters *sent_counter* and *sent_subcounter*, which provide clues regarding the levels of the sentence. But there are two issues: (1) in the given example, the verse is split into three sub-parts. But the second part cannot be considered as a complete sentence since it does not contain a finite verb.³ It would have been more appropriate to call it a phrase rather than *sent_subcounter*. (2) This representation is not uniformly available for all the texts.

For the tasks of segmentation and morphological parsing, hemistich-level analyses might be sufficient because no *sandhi* occurs across hemistichs, and the morphologi-

² This is unique across the entire DCS database

³ Considering *Kātyāyana’s vārtika (eka-tiṅ vākyam)* for *Aṣṭādhyāyī* 2.1.1.

cal analyses can be obtained only after deciding the segmented words. Choosing the contextual morphological analysis might require the entire sentence. DCS also provides details of part of speech tags and dependency relations (both based on Universal Dependencies) and semantic ids of word senses, all of which would be used in the subsequent tasks of sentential analysis (dependency parsing, word sense disambiguation).⁴ And these make sense only when we consider the entire sentence as context.

However, in the current alignment of DCS and SH, all the entries which are marked as sentences in DCS are considered as sentences in SH as well, even if they are only partial sentences or hemistichs. This is mainly to normalize the existing dataset and maintain a uniform representation of DCS and SH. To build a dataset with the actual sentence-level analyses, DCS has to be reorganized wherever necessary, which is a tedious process.

3.1.2 Chunk boundaries

When two consecutive words undergo *sandhi* operation, then the following are the possible changes:

1. only the last character (and occasionally the last two characters⁵) of the first word undergoes change,
2. only the first character of the second word undergoes change, or
3. both the last character(s) of the first word and the first character of the second word undergo changes.

In all the three cases, SH considers the two words to be in a single chunk. Chunking happens in SH with the help of certain rules like the terminal *anusvāra* in a word. It is mainly done for exponentially reducing the computational complexity. On the other hand, DCS treats them as a single chunk only in the third case, and in the first two cases, the two words constitute two different chunks. The definitions of chunks in the two systems differ which results into the mismatch between the number of chunks. For example,

⁴ All such information are not uniformly available for all the texts.

⁵ In the case of *visarga-sandhi*, the last two characters undergo transformation or elision depending on the first character of the next word, particularly if we are using Non-Devanagari encoding scheme.

Unsegmented: *yan nābhisthitam*

Segmented: *yat nābhi-sthitam* (which (*yat*) is situated at the navel (*nābhi-sthitam*))

According to DCS, there are two chunks viz. “*yat*” and “*nābhisthitam*” (where *nābhi* and *sthitam* are components of the second chunk), whereas SH treats them as a single chunk since *t* in *yat* is changed to *n* in the presence of the following *n*. This mismatch between the number of chunks affects chunk-by-chunk mapping of the two systems.

3.1.3 Segments

Another drawback of DCS’ representation is that the segmented words are not present separately and one has to construct the words from the available parameters vis-à-vis lemma and inflectional morphology. In the latest updates of DCS, the segmented forms have been introduced. While one set of word forms are generated using the neural network Word Segmentation model proposed by (Hellwig and Nehrdich, 2018), another set of word forms are reconstructed using the lemma and morphological analysis. In the former case, the generated unsandhied forms do not resolve the terminal *sandhis*. For example, *punaḥ* is represented as *punar*. And there could also be possible errors due to segmentation. In the case of the reconstructed word forms, though, these terminal *sandhis* are resolved. But, there could be multiple possible word-forms generated from the same lemma and morphological analysis.

For example, the root *gup* (to guard) is derived with gerundive suffixes (*yat*, *anīyar*, *tavya*) to form respectively, *gopya*, *gopanīya* and *gopitavya* (all the three indicating the sense of “to be guarded / preserved”, the difference being the usage based on context). DCS marks all the three forms as “Ger” (gerundives). With only the root *gup* and the morphological tag “Ger”, it is not trivial to single out one of the three word-forms. Such information missing at the levels of stem and morphological analysis become crucial for word generation. SH, on the other hand, produces the word forms (segments) with the terminal *sandhis* and a one-to-one mapping from words to stem-morphological analysis pairs. This calls for normalization of both DCS and SH analyses into a unique representation which avoids word-level ambiguities in the subsequent stages of processing.

3.1.4 Stem / Root

Before diving into the differences at the stem and morphological analysis levels, it is important to understand how homonymy plays a crucial role. Homonymy is the phenomenon where an entity (a word, stem or root) can possess more than one sense. In English, the word *bank* is considered homonymous as it possesses at least two meanings as a noun, one a *river-bank* and two a place where we store money. It also has an extended meaning of the sense of storing something, as we can observe in the case of *blood-bank*. Similarly, in Sanskrit, we find a rich usage of words and stems with multiple senses. The famous example quoted is the stem *hari* which possesses fourteen different senses, as observed in *Amarakoṣa*.⁶ In addition to homonymy, the derivational process of building words, and compound formations also impart ambiguities in the analysis of stems. The following are a few points to be noted from the perspective of how these phenomena impose challenges during the alignment of the two systems.

Handling derived nominal stems

In the case of derived words,⁷ DCS is not uniform in marking the analyses. Sometimes, it marks both the inflectional as well as the derivational analyses and sometimes only one of them. SH, on the other hand, marks both the analyses uniformly. This may result into one to many mapping if the words are ambiguous at inflectional / derivational level. For example, let us consider the word *hitam* (sent, impelled, urged on, etc.). DCS stores only the inflectional analysis (iic.) with the stem *hita*, while SH provides the base also: *hi* (to send forth, set in motion, etc.) or *dhā* (to put, place, set, etc.). Interestingly, there are four dictionary entries in DCS for the form *hita*, but none of them have any information regarding the base form. The SH's representation is as follows:

```
[hita_1 {pp.}[hi_2]]{n. sg. acc. | n. sg. nom. | m. sg. acc.}
[hita_2 {pp.}[dhā_1]]{n. sg. acc. | n. sg. nom. | m. sg. acc.}
```

The SH dictionary entries for these two stems show the meaning difference.

⁶ *Amarakoṣa* is a thesaurus of Sanskrit, with a collection of around 10,000 words found in regular usage, along with latent information regarding possible ontological relations between these words.

⁷ derived from *dhātu* with *kṛt*-suffixes or *prātipadika* with *taddhita* suffixes.

hita_1	[pp. hi_2] a. m. n. f. hita
(French)	envoyé, lancé, émis.
(English)	sent, launched, issued
hita_2	[pp. dhā_1] a. m. n. f. hita
(French)	placé, mis, disposé convenable, avantageux ; utile, propre à, bon pour <dat. g. loc.> ; salutaire amical, bienveillant ; qui fait le bien avantage, profit, intérêt ; bien, chose utile ; bien-être.
(English)	placed, put, disposed suitable, advantageous ; useful, suitable for, good for <dat. g. loc.> ; beneficial friendly, caring ; who does good advantage, profit, interest ; well, useful thing ; well-being.

The DCS dictionary entries are as follows:

adj	sent; impelled; urged on; set in motion; going; running; speeding
adj	put; laid upon; situated in; established; fixed; arranged; made ready; reckoned among; ...
m	benefactor; a friend
n	anything useful or salutary or suitable or proper; ...

Thus, we can observe that automatic alignment is not possible with just the available DCS parameters. The design decision to mark either the derived lemma or the base lemma depending on the context results into a non-uniform annotation and it also suppresses derivational information which could play an important role in the tasks of compound analysis or sentential analysis. Such cases are recorded separately for discussion.

Handling homonymous stems

One important aspect of SH is that the dictionary has different entries for homonymous stems, and the morphological analyser provides the homonymy index of the stem. For example, the word *siddham* is analysed by SH as shown in table 3.1.

Table 3.1: SH Analysis for the word *siddham*

Stem	Base	Derivational Analysis	Inflectional Analyses
<i>siddha_1</i>	<i>sidh_1</i>	pp.	n. sg. acc. n. sg. nom. m. sg. acc
<i>siddha_2</i>	<i>sidh_2</i>	pp.	n. sg. acc. n. sg. nom. m. sg. acc

If we look at the meanings of these two senses, we find that they are almost opposing each other.

```

siddha_1  [pp. sidh_1] a. m. n. f. siddhā
(French)  accompli, réalisé; gagné, obtenu; parfait
          qui a atteint son but, réalisé son objectif
(English) accomplished, realized; won, obtained; perfect
          who achieved his goal, achieved his goal
siddha_2  [pp. sidh_2] a. m. n. f. siddhā
(French)  empêché, écarté, repoussé.
(English) prevented, pushed aside, pushed back

```

On the other hand, DCS has four entries with varied meanings.

```

adj  driven off; scared away
adj  accomplished; fulfilled; effected; gained; acquired;
     one who has attained his object; successful; ...
m    any inspired sage or prophet or seer (e.g. Vyāsa); ...
n    magic; supernatural power; sea-salt;
     Name eines āsanas; [alchemy]

```

We can manually infer that the first meaning of SH is similar to the second meaning in DCS, and the second in SH is similar to the first in DCS. But an automatic mapping is not trivial in most cases as it requires mapping the meaning spaces of each of the

senses in DCS and SH. Hence, during the alignment process, such multiple senses were clubbed together. Although it is not used now, such distinctions would definitely be of greater use for sense disambiguation of such homonymous words.

Handling the iics of compounds

In the case of the initial component of a compound, SH marks the stem whereas DCS marks the surface form itself as the stem. For example, *mahā* in the compound *mahāyoga*, is analysed by SH with stem *mahat* while DCS marks the iic form *mahā* as the stem. Additionally, SH analyses privative compounds like *anivṛttam* as *a-nivṛtta*, but is also updated regularly to lexicalize such compounds to have non-compositional meaning. Since DCS' analysis depends on the compositionality according to the context, the analysis might not map with SH's analysis.

Handling the derived verbal stems

In the case of derived verbal stems⁸ such as causative forms, DCS provides the analysis with the causative verbal form as the stem while SH provides the base verb stem with “causative” as a feature in the analysis. For example, for the word *gamayati* (sends), DCS marks the derived verbal form *gamay* (to send) as the stem while SH marks the underived verbal form *gam* (to go) as the stem and “causative” is marked as a feature along with other features viz. tense (present active), person (third), number (singular). Here one needs to construct the causative form *gamay* from *gam* + *causative* in order to align the morphological analysis with that of DCS, which cannot be trivially done as it involves the rules from grammar, but other measures have to be taken to map such stem forms. In this case, having both *gam* and *gamay* in the annotation would definitely help a reader understand the effect of the causative suffix in the stem, and also to associate the exact meaning from both the stems.

⁸ derived from *dhātu* with *san*-suffixes

3.1.5 Morphological Analysis

Noun

As far as the inflectional analysis of nouns is concerned, barring the nomenclatures, the analysis of DCS and SH match perfectly. However, for numerals and pronominal forms, SH marks a “*” for gender and DCS leaves the value empty. The cases and numbers are matched without any discrepancy.

Verb

Regarding the verbal inflectional analysis, DCS marks the “Tense”, “Mood”, “Voice”, “Number” and “Person”. The tense-mood pair of DCS has to be mapped with SH’s tense parameter which follows the *lakāra* system with the linguistics nomenclature. For example, the present indicative, imperative and optative forms are addressed with the Tense parameter as “Pres” and the Mood parameter as “Ind”, “Imp”, “Opt”, respectively by DCS. On the other hand, SH represents them as “pr.”, “imp.” and “opt.”, respectively referring to the *laṭ*, *loṭ*, *vidhiliṇ lakāras*. In some cases, there is a one-to-many mapping from DCS to SH, in the most recent updates of DCS where the Aorist and the Perfect Tenses have been clubbed as Past, while SH continues to provide “aor.” and “pft.”, respectively. We can also find many-to-one mapping from DCS to SH, in the subjunctive and injunctive moods. DCS marks subjunctives with both Present and Aorist (Past) tenses, but SH sticks with one “subj.”. SH does not handle subjunctives and injunctives extensively, and also the pluperfect forms. Hence mapping the DCS entries with such analysis is not trivial at the moment.

A mapping of DCS’ “Number” and “Person” with SH’s analysis is possible. The “Voice” parameter is tagged only when the verb is in the passive voice. Thus mapping verbs in passive voice is trivial. But SH also provides the information of the set of suffixes (*parasmaipada* (active voice) vs *ātmanepada* (reflective / middle voice)) which DCS doesn’t distinguish. The *prayoga* and the *pada* information are clubbed in SH as: “ac.” for *kartari parasmaipada*, “md.” for *kartari ātmanepada* and “ps.” for *karmaṇi / bhāve*. Additionally, SH also marks the class (*gaṇa*) to which the verb belongs to. This information is crucial from the disambiguation point of view since the same verbal root with different classes would have different meanings. SH also provides the conjugation

information (causative, desiderative and intensive) of the verbs while DCS doesn't mark these. In some cases, both non-causative and causative verbs can have the same surface forms and this distinction helps in such disambiguation.

Let us try to quantitatively deduce the difficulty in such mappings, taking one example. SH provides the *gaṇa* (10), *prayoga-pada* pair (3), and conjugations (1 primary and 3 secondary). For the present indicative form in active or middle voice, ("Tense=Pres|Mood=Ind" in DCS), we obtain a mapping in SH with a maximum of $10 \times 2 \times 4 = 80$ corresponding entries. For the passive voice, we find a maximum of 40 corresponding entries, putting together 120 resulting in a 2-to-120 mapping for one *lakāra* - *laṭ*. Such is the gravity of disambiguation required while aligning the DCS and SH representations.

Primary Derivative

For the primary derivatives (*kr̥danta*), there are two morphological analyses: derivational and inflectional. Both SH and DCS indicate the primary derivative suffixes but SH provides additional information about the nature of the verb (class, voice and conjugation) similar to the observation recorded in verbs. DCS on the other hand does not mark this information for such forms which results into one-many mapping from DCS to SH.

Secondary Derivative

A secondary derivative suffix (called *taddhita*) is used on a nominal stem to form another stem with a different sense. SH does not automatically detect the suffixes but prominent stems derived using secondary suffixes have been lexicalized in the dictionaries. Hence, some of the secondary derivative nouns are recognized by the engine but with only the inflectional morphological analysis. DCS does not mark any secondary affix separately and annotates all such forms with the inflectional morphological analysis and lexical entries from its dictionary as the stems. Thus, some of these secondary derivative nouns would be aligned across the two systems, and those which aren't lexicalized in at least one of the two systems would go unaligned.

Invariables

SH considers different kinds of invariables like particles, conjunctions, adverbs, absolutes, etc. and put together all of them under the tag indeclinables or *avyayas*. This set also contains *avyayas* which are obtained by affixing *dhātus* with certain *kṛt*-suffixes (like *ktvā*, *tumun*), or *prātipadikas* with certain *taddhita*-suffixes (like *tasil*). DCS marks some of them as “ind” but mostly leaves such fields empty. With the help of the lexicon ID, one can check if it is an indeclinable or not from the grammar field of the lookup table provided for the lexicon. Some of the words with empty morphological analysis can be handled in this way. But one has to resort to other techniques in case the lexicon fails to give any information about the morphological category.

3.1.6 Compounds

In addition to the issue with iics of compounds (section 3.1.4), DCS and SH also differ in respect to the compositional analysis of a compound. If the context requires a compositional analysis, the initial component (called iic - *in initio compositi*) is marked with “Cpd” and the final component (called ifc - *in finito compositi*) possesses the inflectional morphological analysis of the entire compound word. In case of a non-compositional requirement, the entire compound is considered as a single element of the lexicon, and we would find only the morphological analysis of the entire compound. SH, without the knowledge of context, produces compositional analysis always but may also produce the non-compositional analysis if the entry is lexicalized. Thus, compounds like named entities which are lexicalized by SH with non-compositional analysis would map with that of DCS. But the other non-compositional compounds of DCS would require other measures to align with the SH’s compositional analysis of the compounds.

3.2 Aligning the Parameters

The similarities and the differences between DCS and SH at various levels, described in the previous section, give us clarity on where the alignment can be done trivially, and where measures have to be taken to consider the linguistic differences during the

alignment process. In this section, we shall summarise the differences that can be handled, and those which cannot be.⁹

The alignment is done sentence-wise, where the sentences from DCS are directly run on SH to get the results. Hence, sentence-level issues in DCS are not considered during the alignment.

The alignment has been attempted three times, where for the first two attempts, the analyses were compared based on the chunks. For the third alignment, the in-position chunk-level comparison was removed, and an overall comparison was attempted where only the presence of DCS analysis was detected amongst the SH analyses, and not based on positions of the segments.¹⁰ The segmented forms were not proposed by DCS at the time of the alignments, and one of the main motivations for the alignments was to generate these forms with the help of the analyses from SH.

Considering the stem-level differences, since both the systems use an adaptation of Monier-Williams Sanskrit-English dictionary, most of the primary stems or base roots will have similar forms, thus mapping them is easy. The derived stems of DCS are compared with both the base and derived stems of SH. For some stems (pronominal and iic), a lookup table was generated to map the stems from the two systems. The conjugated roots also required a separate lookup table where the DCS conjugated root is mapped to the SH base root wherever necessary. Homonymy has not been addressed in this alignment process as it is not trivial to align the homonymy indices of the two systems mechanically.

Considering the differences at the morphological analyses level, for the first alignment, integer values representing each of the morphological analyses were generated (addressed as “CNG” values). A lookup table was constructed to map the SH analyses with these “CNG” values. For the remaining alignment approaches, this intermediate “CNG” values are removed and a separate lookup table is generated where both the inflectional and the derivational analysis of the two systems are populated in the ta-

⁹ The design decisions of DCS and SH along with their differences are documented in greater detail in Appendix A.

¹⁰ Since the chunk segments were not available during Alignment 1, the purpose of the alignment was to generate the segments by aligning the stem / root and morphological analyses. For the Alignments 2 and 3, although chunk segments were available in DCS, due to the segment-level issues discussed in section 3.1.3 (terminal sandhi, issues in reconstructed forms), it is not trivial to consider measures like edit distance to join SH chunks when aligning with a DCS chunk. Hence the chunk-level mapping was lifted off during Alignment 3.

ble. All the possible morphological analyses which are available in DCS and which are generated by SH are only considered in this table. Since SH proposes a lot more information than DCS, the representation of SH is considered as the normal form, and the DCS morphological analyses are converted to SH representation accordingly using the lookup table.

This alignment of the manually tagged analyses of DCS with one of the analyses produced by SH would provide us with:

1. Identifying wrong annotations from DCS,
2. Consistent uniform (and normalized) analysis,
3. Probable compounds with non-compositional meaning,
4. Mapping between the morphological analyses of DCS and those of SH, and
5. Parallel corpus of segmented-unsegmented texts.

Chapter 4

DCS-SH Alignment

Three alignment approaches were carried out. The first alignment process is in essence the same as that described in SIGHUM (Chapter 2.5.1). The main difference is that an improved version of SH is considered and the differences in the representation of linguistic information was given more importance. The procedure for Alignment 1 was carried out taking insights from SIGHUM.¹ All the sentences from DCS were considered for the alignment process. This resulted into the Alignment 1 dataset. This dataset was released partially for the Word Segmentation and Morphological Parsing for Sanskrit - Hackathon.² This will be referred to as the Hackathon dataset. Additionally, the alignment process and the insights from the SIGHUM alignment helped in preparing the pre-requisites which made the next two alignments simpler and faster.

In 2022, DCS released the improved version of the data in CoNLL-U format.³ SH was also improved further taking into consideration the observations from Alignment 1. Hence the alignment process was carried out **with and without** considering the **differences due to chunk-unit** in both SH and DCS, respectively resulting into the **second and third** alignments.

For all the three approaches, alignment was done in four steps as follows:

1. For every sentence of DCS, extract DCS data and SH analyses,
2. Convert the extracted data into a normalized form,
3. Compare DCS entry with all possible analyses of SH to get the aligned analysis,
4. From the entries where the alignment was not possible, handle the mismatches due to compounds ignoring compositionality, handle causative-root mapping, preverbs, etc. and rerun the alignment process.

¹ Since the codebase for SIGHUM alignment was not available, a revised version of the codebase was created using the details from SIGHUM.

² <https://sanskritpanini.github.io/>

³ <https://universaldependencies.org/docs/format.html>

4.1 STEP 1 - Extraction of data

Both DCS and SH are under continuous development since their introduction. While new texts are being annotated in DCS, SH has seen developments in its lexicon as well as its engine that performs parallelly the segmentation and morphological recognition. Although a similar alignment had been done previously (in SIGHUM), for the current alignment, the latest updates of the data from DCS and of the analyses from SH are required. Hence, instead of re-using the existing alignment data, new data is extracted from both the systems.

4.1.1 Extracting DCS data and its representation

SIGHUM represented DCS SQL analyses as Pickle objects (table 2.2). These objects were converted to a JSON format for the first alignment (table 4.1). For the second and third alignments, the CoNLL-U representation of DCS was converted to a JSON representation (table 4.2) for ease of analysis.

Table 4.1: DCS data in JSON format for First Alignment

Sentence Id	→ 1
Sentence	→ <i>ādīśvarāya praṇamāmi tasmai yenopadiṣṭā haṭhayogavidyā</i>
Lemmas	→ [[“ādi”, “īśvara”], [“praṇam”], [“tad”], [“yad”, “upadiṣṭā”], [“haṭhayoga”, “vidyā”]]
Morph Code (CNG)	→ [[“3”, “109”], [“-11”], [“109”], [“89”, “30”], [“3”, “30”]]
Morphs	→ [[“m”, “m”], [“1. Ā.”], [“pron”], [“pron”, “6. Ā.”], [“m”, “f”]]
Der Lemmas	→ [[“”, “”], [“”, [“”], [“”, “upadiś”], [“”, “”]]
Der Morph Code (Der CNG)	→ [[“”, “”], [“”, [“”], [“”, “-190”], [“”, “”]]
Prefixes	→ [[“”, “”], [“pra”], [“”, [“”, “upa”], [“”, “”]]

Table 4.2: DCS data in JSON format for Second and Third Alignments

text	→	<i>gheraṇḍasaṃhitā</i>
text ID	→	1
chapter	→	ghers, 1
chapter ID	→	270
sent ID	→	1
joint sentence	→	<i>ādīśvarāya praṇamāmi tasmai yenopadiṣṣā haṭhayogavidyā</i>
unsegmented form	→	[<i>“ādīśvarāya”, “praṇamāmi”, “tasmai”, “yenopadiṣṣā”, “haṭhayogavidyā”</i>]
stem	→	[<i>“ādi”, “īśvara”</i>], [<i>“praṇam”</i>], [<i>“tad”</i>], [<i>“yad”, “upadiṣṭa”</i>], [<i>“haṭhayoga”, “vidyā”</i>]]
stem ID	→	[<i>“57973”, “64099”</i>], [<i>“162415”</i>], [<i>“37875”</i>], [<i>“37877”, “160400”</i>], [<i>“121886”, “121245”</i>]]
morph	→	[<i>“Case=Cpd”, “Case=Dat Number=1 Gender=Masc”</i>], [<i>“Tense=Pres Mood=Ind Person=1 Number=Sing”</i>], [<i>“Case=Dat Number=1 Gender=Masc”</i>], [<i>“Case=Ins Number=1 Gender=Masc”</i>], [<i>“Case=Nom Number=1 Gender=Fem VerbForm=PPP”</i>], [<i>“Case=Cpd”, “Case=Nom Number=1 Gender=Fem”</i>]]
grammar	→	[<i>“m”, “m”</i>], [<i>“1. Ā.,1.P.”</i>], [<i>“pron”</i>], [<i>“pron”, “6.P.,6. Ā.”</i>], [<i>“m”, “f”</i>]]
xpos	→	[<i>“NC”, “NC”</i>], [<i>“V”</i>], [<i>“PRD”</i>], [<i>“PRL”, “PPP”</i>], [<i>“NC”, “NC”</i>]]
preverbs	→	[<i>“”, “”</i>], [<i>“pra”</i>], [<i>“”</i>], [<i>“”, “upa”</i>], [<i>“”, “”</i>]]

4.1.2 Extracting SH data and its representation

Alignment 1

Figure 2.1 showed the web-based representation of the SH analyses in the form of a graphical interface. SIGHUM scraped through the web page results to extract the analyses. The first alignment followed a similar approach and extracted the parameters such as word, lemma, position, morphological information (both inflectional and

derivational), chunk number, word length, and pre-verbs. This was represented as a graph in the GraphML format with the segments as nodes, segment parameters as the node attributes, and the edge values indicating the co-occurrence of the participating nodes in a solution. The morphological information was represented using the “CNG” parameter and the conversion of the inflectional and derivational morphological analyses to their corresponding “CNG” values was done in this stage.

Every node during the first alignment was represented in the following format:

```
( id, { color_class, position, chunk_no, word, lemma, sense,
      cng, pre_verb, morph, length_word, der_pre_verb, der_lemma,
      der_sense, der_morph, der_cng, char_pos } )
```

All these values are extracted from the scrapped data. lemma, sense, pre-verb, morph and cng denote respectively the word’s *prātipadika/dhātu* (stem/root), sense (based on different meanings), *upasarga* (pre-fix for verbs), morphological details, CNG - (case, number and gender value for nouns and tense, mood, person, number and voice for verbs) corresponding to the morphological information. der_lemma, der_sense, der_pre_verb, der_morph, der_cng correspond to the information pertaining to derivational morphology. color_class is an attribute for the interface and it denotes a particular color depending on the *phase*. For example, iics have yellow, substantive/adjective forms have blue, indeclinable forms such as adverbs, conjunctions, prepositions have pink colors, etc. Except for the derivational details, the sense information and the position based on character, all the others were created by SIGHUM.

As in SIGHUM, the graph edge values are used to identify the co-existence of two nodes (segments) in a single solution. The edge value ‘1’ indicates that the two nodes can be a part of a solution. The value ‘2’ indicates that the two corresponding nodes cannot be in a single solution. For example, in the compound *pātālabhāsuram* (Figure 4.1), *pātāla* and *bhāsuram* are non-conflicting nodes and hence their edge is labeled ‘1’. But *bhā* and *bhāsuram* are separate nodes which are conflicting, and hence their edge is labeled ‘2’. Later, the value ‘3’ will be stored for the edges in the correct segmentation solution. The edge values provide a way to connect the nodes and form the solutions. Thus this graph represents a reusable structure synonymous with the graphical interface of SH.

Table 4.3: Reference for the fields in SH's analysis

field_id	value	field_id	value
1	segment_id	8	derived_stem_sense
2	word_form	9	inflectional_morph_analysis
3	chunk_id	10	base
4	pos_in_chunk	11	base_sense
5	phase	12	derivational_morph_analysis
6	phase_color	13	pre_verb
7	derived_stem		

Hence, *phase*, *phase_color*, *derived_stem_sense* and *base_sense* are not used in the alignment process. *chunk_id*, *pos_in_chunk*, *base*, *derivational_morph_analysis*, *derived_stem* and *inflectional_morph_analysis*, are the primary parameters for comparison in the alignment process, where *chunk_id* and *pos_in_chunk* are considered for alignments 1 and 2 only.

4.2 STEP 2 - Normalization

The comparison on the design decisions of the two systems (Chapter 3.1) showed that there were many differences and each has to be considered prior to the process of alignment to make sure that the representations of both the systems are normalized. The normalization is done at three levels: chunk, stem and morphological analyses, as explained ahead.

4.2.1 Aligning Chunks

We observed how DCS and SH differ in marking the chunk boundaries owing to how they address the segmentation within a chunk. Initially, for the first two alignments, this chunk-level difference was not considered at all and an in-position comparison of the segments of DCS and SH was carried out during the alignment. For the third alignment, though, considering these differences, the strict comparison was lifted off

and only the analysis reported by DCS was compared with all the analyses produced by SH without matching the chunks.

4.2.2 Aligning Stems

- **Base vs Derived stem:** SH provides both the base and the derived stems while DCS sticks to one stem that is more apt in the given context. For the word *gopanīyam* (to be concealed or hidden), DCS marks the stem as *gup* (to guard; to defend; to protect; to preserve; to hide; to conceal), while SH produces additionally the derived stem *gopanīya*. In such cases, the DCS stem is aligned with either the derived stem or the base stem of SH, whichever is available in DCS, but both the derived and the base stems of SH are considered for the final dataset.
- **Mapping pronouns:** SH uses the *Pāṇinian* system for pronouns' stems while DCS uses the conventional forms for the stems. For example, the first person pronoun is represented as *asmad* by SH and *mad* by DCS. Similarly the second person pronoun is represented as *yuṣmad* by SH and *tvad* by DCS. While the stem-forms *mad* and *tvad* are conventionally used in the recent times, the *Pāṇinian* tradition follows the stem-forms *asmad* and *yuṣmad*. These are normalized to the SH format.
- **Mapping derived verbal stems:** Causative, desiderative and intensive forms are stored with modified stems in DCS without providing its derivation while SH stores the base stem with a morphological feature such as causative (ca.), desiderative (des.) or intensive (int.). For example, *gopayet* (hide something) is analysed by DCS with the modified stem *gopay*, while SH uses the root *gup* (to guard / hide) along with the morphological feature “ca.” (causative). Since the information from SH is more detailed, SH's format is considered for the alignment.
- **Handling Nasal Consonant (*Anusvāra*):** In both DCS and SH, the forms with *anusvāra* (nasal consonant) are normalized to their corresponding *anunāsika* (homonasal consonant). Similarly in some verbal non-finite forms, reduplication

of a consonant is observed optionally. In all such cases the forms are normalized to the ones without reduplications.

- **Handling iics:** SH and DCS produce different stems for the *iic* component of compounds. These are also normalized to the forms produced by SH for alignment purpose.
- **Handling Preverbs:** DCS produces stems sandhied with their preverbs. For the first alignment, the stems of SH were disjoint with the preverbs and hence preverbs were sandhied with the stems.⁵ For example, *pra-nam* → *praṇam* for the word *praṇamāmi*. New nodes were created with the sandhied stems. For the second and third alignments, the .tsv format produces the stems sandhied with their corresponding preverbs.

4.2.3 Aligning Morphological Analysis

- **Morphological Analysis Mapper:** Since the representations of the morphological analysis is different in the two systems, a one-to-one mapping is required to compare the analyses of both the systems. For the first alignment, a look-up table was constructed with the keys being unique CNG entries and the values being SH's morphological analysis. For the second and third alignments, a morphological analysis table was built to map the morphological analysis of SH with that of DCS.⁶
- **Inflectional vs Derivational Analysis:** It is observed that DCS provides the derivational analysis only when the context requires it. Hence, the first preference for comparison is given to the default inflectional analysis alone. If the alignment fails with the inflectional analysis, then the derivational analysis is taken into consideration along with the inflectional analysis, provided the stem marked by DCS is mapped with SH's base stem. For example, the word *hitam* has the stem *hita* and morphological analysis "Case=Nom|Number=1|Gender=Neut" but SH produces the derivational analysis as well ("pp."). In this case, the inflectional analysis alone is considered for the alignment. Once an alignment is ob-

⁵ For this process, the *sandhi* joiner of the *Saṃsādhani* platform was utilised.

⁶ The look-up table for DCS' CNG and SH analyses and the morphological analysis mapper are available at: dcs-sh-alignment

served, both the inflectional and derivational analyses are considered for the resultant dataset. For the word *gopanīyam*, DCS marks both the inflectional as well as the derivational form: “Case=Nom|Number=1|Gender=Nuet|VerbForm=Ger”. DCS’ stem (*gup*) matches with the SH’s base stem, and DCS’ “VerbForm=Ger” matches with the “pfp. [2]” of SH, where “[2]” indicates the specific kind of gerundive which DCS does not mark. Hence both the inflectional and the derivational morphological analyses are considered together for the alignment.

- **Handling Missing morphological Analyses:** There are about 4.5 million word references in DCS and about 17% of these do not have the morphological analysis marked. We extracted a list of indeclinables from DCS’ dictionary and *Saṃsādhani* and compared it with the list of unanalysed words. About 90% of these words were found to be indeclinables. Thus during the alignment, these words were annotated as indeclinables.

4.3 STEP 3 - Alignment

For every sentence, DCS’ ground truth analysis (stem and morphological analysis) of a segment is compared with the analysis of every segment produced by SH. DCS’ stem is compared with the base and derived stems of SH. DCS’ morphological analysis is compared with the inflectional and derivational morphological analysis of SH, using the CNG look-up table (in Alignment 1) and the Morphological Analysis Mapper (in Alignments 2 and 3).

For the first and the second alignments, *chunk_id* and *position_in_chunk* help in an in-position comparison where the stem and morphological analysis from DCS are aligned with the SH analyses at exactly the same position in the sentence as proposed by DCS. It was observed in Chapter 3.1.2 that DCS and SH differ in their definitions of a chunk and how it affects the chunk-based in-position alignment process. To circumvent this drawback, this restriction of in-position comparison was lifted off in the third alignment.

4.3.1 Alignment Results and Observations

Table 4.4 shows the results after Step 3 of the alignment process in the three approaches:

Table 4.4: Alignment Results Step 3			
	A1 ¹	A2 ²	A3 ³
Number of sentences	621,445	621,327	621,327
Sentences Unrecognized by SH	* ⁴	145,313	145,313
Sentences Recognized by SH	*	476,014	476,014
Aligned sentences	73,429	112,738	160,453
Multiple Alignments	18,100	7,313	10,996
Missed Alignments	503,761	355,963	304,565

¹ Alignment 1, used the DCS SQL Dump

² Alignment 2, used the DCS CoNLL-U data

³ Alignment 3, used the DCS CoNLL-U data (ignoring chunk boundaries)

⁴ * - data unavailable

The following observations are recorded after Step 3:

- The alignment is much stricter when the chunk-level comparison is present and hence the number of sentences with complete alignment is less in Alignments 1 (73,429) and 2 (112,738) when compared to 3 (160,453).
- With the recent updates to SH and the modification of the alignment algorithm, the number of aligned sentences increased from Alignment 1 to Alignments 2 and 3.
- Sentences falling under Multiple Alignments have one to many correct mappings of DCS to SH analyses and hence it is not trivial to disambiguate them.
- The sentences under Missed Alignments are then considered for further processing in Step 4.

4.4 STEP 4 - Alignment Issues and Modifications

During the first alignment, upon manually analysing the sentences that fall under the missed and multiple alignments category, it was observed that the preverbs (*upasargas*) were not sandhied with their corresponding stems. For example *praśaṃsanti* has the stem form as *pra-śaṃs* in SH while DCS had *praśaṃs* as its stem. An internal *sandhi* was carried out in such cases between the preverb and the base stem (or root) and a new node with the sandhied stem was inserted amongst all the analyses. During the second and third alignments, SH was internally modified in such a way that it produced the stem already sandhied with its preverb.

The causative forms of the verbs (in the tenth *gaṇa*) were annotated differently by the two systems. While SH marked the base root of the verb and additionally provided the information of causative as one of the features, DCS annotated the intermediate representation of the causative form as its stem. For example, SH analyses the word *pūjayati* with *pūj* as the stem and the morphological analysis as “ca. pr. [10] ac. sg. 3”, where “ca.” is the additional causative information of the verb. On the other hand, DCS analyses the stem as “*pūjay*” and the morphological analyses contains only the information regarding the tense-mood combination, person and number (“Tense=Pres|Mood=Ind|Person=3|Number=Sing”). In such cases, a lookup table was prepared for all possible roots occurring in DCS which stores the base root and the intermediate causative representation of the stem as pairs. This lookup table is further used in the next stage of the alignment and updated as and when a missed out alignment is found to be due to the presence of the causative feature.

Another issue was regarding the compositionality of compounds. As discussed in Chapter 3.1.6, DCS annotates a compound as non-compositional when the context requires the non-compositional meaning and otherwise a compositional meaning. On the other hand, SH produces the non-compositional analysis only when the form is lexicalized but produces the compositional analysis always. When a compound with a non-compositional analysis in DCS is encountered which is not lexicalized in SH, a mismatch occurs. For example, the compound *śaṅkhaśuktyudbhavam* is analysed in SH separately as *śaṅkha-śukti-udbhavam*, but *śaṅkha-śuktyudbhavam* is the expected solution according to DCS. So, for this chunk, all possible combinations of compounds

are constructed and then each of it is compared to the DCS analysis. If the correct one is matched, a new node with the modified lemma, word and other information is created. The values for the attributes lemma and word, are kept as hyphen-separated components instead of the sandhied components for future usage in compound analysis. The chunk_ids are taken from the first segment (*śaṅkha*) and the morphological analyses taken from that of the third segment (*udbhavam*).

We should also make a note here that the total number of combinations is a Catalan number. So, generating all possible combinations for a given compound results in exponentially slow algorithm. For the second and third alignments, the same approach was carried out and a new segment with the updated compound information was added to the list of existing segments.

Finally, the process of alignment is carried out once again on the mis-aligned entries with the modified analyses.

4.5 Alignment observations

Table 4.5 records the overall observations of the Alignment process considering the Alignment 1 procedure, and compares it with the same over sentences from SIGHUM.

Type	SIGHUM	Alignment 1
Overall Sentences	119,004	621,445
Aligned	65,699	130,439
Aligned (with multiple analyses)	36,755	84,469
Not aligned (with missed analyses)	4148	103,808
Not aligned (with both multiple analyses and missed lemma)	4265	110,760
Not aligned (modifications could not be done)	6925	165,456
Not aligned (due to other reasons)	1212	26,513

Table 4.5: Comparison of Alignment 1 procedure over SIGHUM sentences and all DCS sentences

Table 4.6 shows the comparison of observations in Step 3 versus Step 4. Table 4.7 shows the summary of the alignment process in all the three approaches.

The motivation of the alignment was to produce a dataset which is normalized, standardized and which is rich in features from both DCS and SH. While it wasn't trivial to generate the combined features for all the sentences of DCS owing to various

Table 4.6: Alignment Comparison Steps 3 and 4

	A1 ¹		A2 ²		A ³	
	Step 3	Step 4	Step 3	Step 4	Step 3	Step 4
All Sentences	621,445	338,305	621,327	163,221	621,327	151,137
Unrecognized	* ⁴	*	145,313	0	145,313	0
Recognized	*	*	476,014	0	476,014	0
Aligned	73,429	56,921	112,738	34,801	160,453	30,718
Multiple	18,100	66,369	7,313	26,865	10,996	18,312
Missed	529,916	215,015	355,963	101,555	304,565	102,107
Modified	338,305	*	163,221	*	151,137	*

¹ A1 - Alignment 1² A2 - Alignment 2³ A3 - Alignment 3⁴ * - data unavailable

Table 4.7: Overall Alignment Observations

	A1 ¹	A2 ²	A3 ³	S ⁴
All Sentences	621,445	621,327	621,327	119,004
Aligned	130,350	147,539	191,171	65,699
Multiple	84,469	34,178	29,308	36,755
Missed⁵	406,626	439,610	400,848	16,550

¹ A1 - Alignment 1² A2 - Alignment 2³ A3 - Alignment 3⁴ S - SIGHUM (The dataset of 119,004 sentences from the SIGHUM dataset were considered to compare its results using the current alignment implementation with the observations of the current alignment over the entire DCS dataset.)⁵ Includes those sentences which were not recognized by SH, those sentences which were not aligned in step 4 and neither modified in step 4

reasons quoted earlier, it was possible to at least generate it for almost a third of the DCS sentences. Following are the achievements of the alignment processes:

1. Parallel Corpus of unsegmented-segmented sentences,
2. List of all aligned tuples (word, lemma, cng, morph) and their frequencies from Alignment 1 and list of all aligned tuples (word, stem, base, derived_morph, base_morph) and their frequencies from Alignments 2 and 3,
3. List of all *sandhi* rules and their frequencies,
4. In the first alignment, the edge values in the GraphML files had been updated further to have value 3 between the segments which are a part of the ground truth segmentation. In addition to this, unified datasets with attributes from both the DCS and SH analyses (in JSON and GraphML format from Alignment 1 and JSON format from Alignments 2 and 3) were built. A sample of the **unified dataset** from Alignment 1 is shown in table 4.8. The JSON format of the same sentence after alignments 2 and 3 are shown in table 4.9. In addition to the keys shown here, additional keys like upos, xpos, semantic_ids, dependency relations, which are useful for subsequent tasks are present along with those which correspond to the metadata of the sentence like text, text_id, chapter_id, sent_counter.
5. We can observe a significant increase in the number of aligned sentences from Alignments 1 to 3. But the number of sentences with multiple analyses has reduced. This shows that the addition of linguistic parameters and the removal of chunk boundaries have helped towards the alignment of sentences with multiple analyses.
6. In spite of following the same approach as in SIGHUM, the current implementation could align only 65,699 of the 119,004 sentences from SIGHUM (table 4.5). However, the total number of aligned sentences is more than that of SIGHUM. This difference could be due to the updates to the two resources (DCS and SH), and also due to the implementation style of the two alignments.⁷
7. For alignment 1, there were three inconsistencies observed during Step 3: preverbs, causative forms and non-compositional compounds. These were resolved during Step 4 resulting in alignment of 56,921 sentences. For alignments 2 and 3, the preverbs were internally handled by SH, and the causative roots mapping obtained earlier was used here in Step 2. Thus, compositionality was the only

⁷ All the resources used for the alignment are recorded in Appendix B.

inconsistency resolved (partially) during Step 4 resulting in the alignment of 34,801 and 30,718 sentences respectively during alignments 2 and 3.

Table 4.8: Alignment 1 Results format in JSON

key	value
sent_id	14,924
sentence	<i>vāyuh prakupito yasya rūkṣāhārasya dehinaḥ</i>
word	[[“vāyuh”], [“prakupitaḥ”], [“yasya”], [“rūkṣa”, “ahārasya”], [“dehinaḥ”]]
stem	[[“vāyu”], [“prakupita”], [“yad”], [“rūkṣa”, “ahāra”], [“dehin”]]
sense	[[“1”], [“1”], [“1”], [“1”, “1”], [“1”]]
morph	[[“m. sg. nom.”], [“m. sg. nom.”], [“m. sg. g.”], [“ic.”, “m. sg. g.”], [“m. sg. g.”]]
cng	[[“29”], [“29”], [“149”], [“3”, “149”], [“149”]]
base_stem	[[“”], [“prakup”], [“”], [“”, “”], [“”]]
base_sense	[[“0”], [“1”], [“0”], [“0”, “0”], [“0”]]
base_morph	[[“”], [“pp.”], [“”], [“”, “”], [“”]]
base_cng	[[“”], [“-190”], [“”], [“”, “”], [“”]]
pre_verb	[[“”], [“pra”], [“”], [“”, “”], [“”]]
graphml_node_ids	[[“47”], [“54”], [“34”], [“52”, “36”], [“39”]]

4.5.1 Causes of mismatches

We can observe from table 4.6 that in all the three alignment approaches, there was a significant number of sentences that went unaligned in both step 3 and 4. The common causes of mis-alignment are recorded here.

Non-recognition by SH: The first case deals with the Out Of Vocabulary words (unrecognized words) and it amounts to about 25% of the DCS sentences. Secondary

Table 4.9: Alignment 2 and 3 Results format in JSON

key	value
sent_id	"14924"
joint_sentence	" <i>vāyuh prakupito yasya rūkṣāhārasya dehinaḥ</i> "
position	[["1", "2", "3", "4", "5", "6"]]
unsegmented_form	[" <i>vāyuh</i> ", " <i>prakupito</i> ", " <i>yasya</i> ", " <i>rūkṣāhārasya</i> ", " <i>dehinaḥ</i> "]
sh_word	[[" <i>vāyuh</i> ", " <i>prakupitaḥ</i> ", " <i>yasya</i> ", " <i>rūkṣa</i> ", " <i>āhārasya</i> ", " <i>dehinaḥ</i> "]]
word	[[" <i>vāyuh</i> ", " <i>prakupitaḥ</i> ", " <i>yasya</i> ", " <i>rūkṣa</i> ", " <i>āhārasya</i> ", " <i>dehinaḥ</i> "]]
phase	[["Nouc", "Kric", "Pron", "licc", "Nouv", "Nouc"]]
phase_color	[["Deep_sky", "Deep_sky", "Light_blue", "Yellow", "Deep_sky", "Deep_sky"]]
dcs_stem	[[" <i>vāyu</i> ", " <i>prakup</i> ", " <i>yad</i> ", " <i>rūkṣa</i> ", " <i>ahāra</i> ", " <i>dehin</i> "]]
sh_stem	[[" <i>vāyu</i> ", " <i>prakupita</i> ", " <i>yad</i> ", " <i>rūkṣa</i> ", " <i>ahāra</i> ", " <i>dehin</i> "]]
sh_stem_sense	[["1", "1", "1", "1", "1", "1"]]
stem	[[" <i>vāyu</i> ", " <i>prakupita</i> ", " <i>yad</i> ", " <i>rūkṣa</i> ", " <i>ahāra</i> ", " <i>dehin</i> "]]
sh_base	[["", " <i>prakup</i> ", "", "", "", ""]]
sh_base_sense	[["", "1", "", "", "", ""]]
base	[["", " <i>prakup</i> ", "", "", "", ""]]
dcs_morph	[["Case=Nom Number=1 Gender=Masc", "Case=Nom Number=1 Gender=Masc VerbForm=PPP", "Case=Gen Number=1 Gender=Masc", "Case=Cpd", "Case=Gen Number=1 Gender=Masc", "Case=Gen Number=1 Gender=Masc"]]
sh_morph	[["m. sg. nom.", "m. sg. nom.", "m. sg. g.", "iic.", "m. sg. g.", "m. sg. g."]]
sh_base_morph	[["", "pp.", "", "", "", ""]]
pre_verb	[["", " <i>pra</i> ", "", "", "", ""]]
xpos	[["NC", "PPP", "PRL", "JJ", "NC", "NC"]]

derivatives (*taddhita* forms) are recognized by SH only when their stem is lexicalised. There could be a huge list of domain-specific named entities among these unrecognized words. The required solution would be to update SH's lexicon with the missing entries.

Incorrect segmentation by SH: Some forms are not analysed by SH resulting into incorrect segmentations. This could be due to issues in resolving *sandhi*, mostly for sentences from Vedic literature where preverbs are disjoint with their corresponding verbal forms. Incorrect segmentations could also be caused from non-recognition and missed out morphological analyses of certain forms. For example, the sentence *tarjanīmadhyamānāmā aṅgulitrayayogataḥ* has the analysis (stems) from DCS as *tarjanī-madhyamā-anāman aṅguli-traya-yoga*⁸ but SH could analyse only the stems *tarjanī* and *aṅguli*. *Madhyamā* is not analysed as an *iic* and *aṅguli-traya-yogataḥ* is not recognized as a compound. Compositionality (discussed ahead) also plays a role in deciding the correct segmentation in this example.

Issues with CNG: During the first alignment, mapping the CNG values is not one-to-one. For a given CNG value, there could be multiple morphological analyses. For example, the CNG value of -190 has morphological analyses as 'ca. pp.', 'des. pp.', 'int. pp.' and 'pp.' thus multiple analyses are mapped for the same lemma. For the word *vibhūṣitam*, SH produces derivational analysis of both the causative and primary of the root *vi-bhūṣ* while DCS annotates both of their CNGs similarly (-190). Thus, CNG-value based comparison was removed in the second and third alignments where a direct comparison of the morphological analyses was done using the morphological mapper generated earlier.

One-to-many mapping: The mapper between the morphological analyses of DCS and that of SH also has a one to many mapping. In addition to the secondary conjugation suffixes, the information of class (*gaṇa*) is missing in DCS which is important to decide the meaning of the verb. Similarly, the information of active and passive voice is needed for deciding the dependency structure of a sentence, but not annotated in DCS. These are some of the limitations of the morphological analysis mapper.

Compositionality: This is a more challenging case where DCS has an explicit entry for the non-compositional analysis of certain compounds, whereas SH sticks to the

⁸ *tarjanī* (fore-finger), *madhyamā* (middle finger), *anāman* (ring finger) *aṅguli-traya-yoga* (the triad of these fingers together)

compositional analyses.⁹ Compositionality depends on the individual meanings of the combining entities and also the context. The word *pītāmbaram* when referred to as the deity *Viṣṇu*, takes the non-compositional meaning, and when referred to as the yellow cloth, it has a compositional meaning (from its constituents *pīta* (yellow) and *ambaram* (cloth)). It is tedious to update SH's lexicon with all such non-compositional forms as it could also result into over-generation of the lexicon entries.

This was handled partially by generating all possible combinations of the grouping of compound components, which are then compared with the DCS entries. Some of the compounds with a huge number of components were ignored as this part of the algorithm was time consuming since the possible groupings of compound components is a Catalan number, C_n ,

$$C_n = \frac{(2n)!}{(n+1)! n!} \quad (4.1)$$

if there are n components in a compound (Huet, 2009).

⁹ with a few exceptions where certain named entities have been lexicalised with non-compositional analysis

Chapter 5

Ranking Segmentations of Sanskrit Heritage Segmenter

Chapter 2.4 gave a detailed account of the three main stages of SH, namely chunking, segmentation and representation. Generally, representation is considered as a part of the User Interface, and not as a stage in itself. Thus chunking and segmentation form the core of SH. However, the graphical interface is one distinguishing feature of SH, that bridges the gap between the core segmentation algorithm and the user. It has made the results of the segmentation algorithm accessible for a reader and also for a programmer who wishes to explore solving the problem of segmentation and morphological parsing using SH.

However, there is a tight coupling between the segmentation and graphical interface, and with the implementation in Ocaml, most of the efforts for solving the segmentation problem resorted to scraping the webpage, extracting the analysis from this graphical interface and creating their own data structures to incorporate the segments and segmentations for their algorithms. The present work, on the other hand, is intended to come full circle, starting with extracting the analysis from SH for creating the training corpus (chapter 4), addressing the segmentation problem in two stages: segmentation and ranking, where the ranking is incorporated on top of the segmenter, finally providing the revised (most probable) set of solutions to the graphical interface. In essence, a new layer is inserted to the SH between the stages of segmentation and representation, that takes all the segmentation solutions, ranks them, extracts the segments from the top n solutions, and provides them to the graphical interface for representation.

SH produces all possible analyses where the possibilities increase exponentially at each level of analysis, starting from the surface word form all upto the senses of the stems or roots. A single word can be obtained from multiple lexical categories. Simi-

larly, we find syncretism where a single word form can have multiple morphological analysis. We also find homonymy where multiple senses can be referred to by the same stem or root form. On top of all this, we find non-determinism in the *sandhi* rules which further increases the number of solutions. While resolving all of these requires the entire context, here is an experiment to consider an n-gram based ranking mechanism that prioritizes the list of all possible segmentations based on various metrics.

This chapter proposes the ranking criteria, formulates the results of SH, discusses the variations in metrics and the implementation of the ranking algorithm on top of its segmentation engine. Further, three evaluation strategies are described followed by observations of the ranking on various texts evaluated using these strategies. A comparison with the recently developed segmentation models is proposed along with an analysis on the differences in the way segmentation is done in these systems.

5.1 SH Ranking

5.1.1 Ranking Criteria

SH in addition to providing possible ways to segment a string, also provides the *sandhi* information between the segments and also the morphological analyses of the components involved. These features are useful from the disambiguation point of view. For example,

(a) The string *śvetodhāvati* is ambiguous between *śvā itaḥ dhāvati* (A dog runs from here) and *śvetaḥ dhāvati* (The white one runs). Though both of these segmentations are meaningful, the second one is more frequent in Sanskrit than the first. Our segmenter should rank these solutions accordingly.

(b) The string *rāmovanaṅgacchati* can be split in only one way viz. *rāmaḥ vanam gacchati*. However each of these three words have more than one morphological analyses (table 5.1). The SH segmenter instead of providing only one segmentation, provides four different solutions with combinations presented in table 5.2.

Each of these is further ambiguous since the word *vanam* and also *gacchati* as a present participle have multiple morphological analyses, leading to 12 possible unambiguous solutions. But SH segmenter clubs the morphological analyses in a single

Table 5.1: An example of multiple morphological analyses

Segment	Analysis
<i>rāmaḥ</i>	noun → masculine singular nominative of stem <i>rāma</i> verb → present active first person plural of root <i>rā</i> (of class 2)
<i>vanam</i>	noun → neuter singular accusative of stem <i>vana</i> noun → neuter singular nominative of stem <i>vana</i>
<i>gacchati</i>	verb → present active third person singular of root <i>gam</i> (of class 1) present participle → masculine singular locative of root <i>gam</i> (of class 1) present participle → neuter singular locative of root <i>gam</i> (of class 1)

Table 5.2: SH segmentation solutions for *rāmovanaṅgacchati*

No.	Segment	Analysis
1	<i>rāmaḥ</i>	noun → masculine singular nominative of stem <i>rāma</i>
	<i>vanam</i>	noun → neuter singular accusative / nominative of stem <i>vana</i>
	<i>gacchati</i>	verb → present active third person singular of root <i>gam</i> (of class 1)
2	<i>rāmaḥ</i>	noun → masculine singular nominative of stem <i>rāma</i>
	<i>vanam</i>	noun → neuter singular accusative / nominative of stem <i>vana</i>
	<i>gacchati</i>	present participle → masculine singular locative of root <i>gam</i> (of class 1)
3	<i>rāmaḥ</i>	verb → present active first person plural of root <i>rā</i> (of class 2)
	<i>vanam</i>	noun → neuter singular accusative / nominative of stem <i>vana</i>
	<i>gacchati</i>	verb → present active third person singular of root <i>gam</i> (of class 1)
4	<i>rāmaḥ</i>	verb → present active first person plural of root <i>rā</i> (of class 2)
	<i>vanam</i>	noun → neuter singular accusative / nominative of stem <i>vana</i>
	<i>gacchati</i>	present participle → masculine singular locative of root <i>gam</i> (of class 1)

phase where it cannot be disambiguated structurally, and hence only four solutions. Phases are similar to part of speech tags and represent grammatical classes in Sanskrit. These phases are dictated by the grammatical constraints the language imposes during word formation. The structure of a sentence having words with different phases showing their sequences was shown in figure 2.2 (Chapter 2.4.1).

Among these four solutions (table 5.2), the first one is more probable and the remaining three are almost rare or may be possible when embedded in a larger context. With these two scenarios of disambiguation and the available data, one can think of three ranking criteria:

1. use the word frequency: this will be useful to identify the correct segments as in the example *śvetodhāvati* shown earlier in (a), and

2. use the stem-morphological analysis frequency: since SH clubs some of the morphological analyses into a single phase as seen in the example *rāmovanaṅgac-chati* shown earlier in (b), deciding the frequency of an analysis is not trivial when the phase corresponds to multiple morphological analyses. In such cases, the highest frequency of the morph analyses in that phase is chosen for computational purpose.
3. considering the transition information, we can additionally use the transition frequencies. Transition in this context refers to the *sandhi* that occurs ahead of a segment.¹

Equation 2.1 proposed in chapter 2.1.2 comes into picture and is restated below as a *sandhi* (rewrite) rule (Hyman, 2008) or a transition:

$$u|v \rightarrow w \quad (5.1)$$

where u refers to the final one/two characters of the current segment and v refers to the first character of the next segment and w is the resultant character after performing the *sandhi*.

Although these rules are the primary criteria for resolving *sandhi*, the bigram probabilities of the transitions are very sparse and cannot be used alone for prediction. For example, the character $\bar{a}$ in the word *rāmālayaḥ* has the following possibilities while resolving *sandhi*: $a+a$, $a+\bar{a}$, $\bar{a}+a$ and $\bar{a}+\bar{a}$. *rāma* and *rāmā* are the possibilities of the first segment, while *alayaḥ* and *ālayaḥ* for the second segment. Coincidentally all the four are available in the lexicon. Thus it becomes necessary to consider the words and choose the most probable amongst these.

We now have three parameters: word frequency, word and transition frequency, and third, the *phase* frequency, which calculates the highest frequency of the morphological analyses of a given word form in a given *phase*.

5.1.2 Formulation of SH results

Let us first formulate the segmentation results of SH. Given an input sentence S , SH produces a set of all possible segmentation solutions S' . Since SH first divides the

¹ The terms transition and *sandhi* are used interchangeably.

input sentence into chunks, and the segmentation is run on each of the chunks, S is transformed into $C = \langle C_1, C_2, \dots, C_x \rangle$, where C_i refers to the i^{th} chunk. Each chunk is further segmented where a set of chunk-level segmentations are produced. Let us represent the chunk-level segmentations of the chunk C_i as $C'_i = \langle C'_{i,1}, C'_{i,2}, \dots, C'_{i,y} \rangle$. Let a single chunk segmentation solution $C'_{i,j}$ be represented as $\langle c'_{i,j,1}, c'_{i,j,2}, \dots, c'_{i,j,z} \rangle$. Thus the set of all segments can be defined as

$$S' = \{c'_{i,j,k}, i \leq x, j \leq y, k \leq z\} \quad (5.2)$$

where i represents the chunk ID; j represents the chunk-segmentation ID corresponding to i and y represents the total number of chunk segmentations of the chunk i ; k represents the segment ID corresponding to i and j , and z represents the total number segments in that particular chunk segmentation.

Once a chunk level segmentation solution is obtained, its segments are populated on the graphical interface (the tabulated display, $D(S)$, explained in 2.4.1), saving only the union of the set of all segments obtained from all the segmentation solutions. The tabulated display considers three parameters of a segment $(k, (l, z))$, namely the segment (z) , its offset (k) according to the input sequence and the lemmatization (l) . In our case, we do not need the offset parameter, but will require the segment and lemmatization (at a later stage). Hence, our aim is to extend further by translating the chunk-level segmentations of all the chunks into a single list of segmentation solutions representing each of the solutions at the segment level, resulting into $S' = \langle S'_1, S'_2, \dots, S'_p \rangle$. We define a single segmentation solution as $S'_a = \langle s'_{a,1}, s'_{a,2}, \dots, s'_{a,q} \rangle$, where $a \leq p$. And the set of all segments can be defined as

$$S' = \{s'_{a,b}, a \leq p, b \leq q\} \quad (5.3)$$

where a represents the segmentation ID, p is the total number of segmentations formed; b represents the segment ID of a^{th} segmentation and q is the total number of segments in the same segmentation.

The translation from eq. 5.2 to 5.3 requires one to construct the list of all segmentation solutions, by iterating in two levels: chunk and chunk segmentation. The chunk segmentations of a chunk are independent to the segmentations of other chunks, hence

the total number of segmentations would be equal to the product of the lengths of chunk segmentations of each of the chunks, thus p of eq. 5.3 is $\prod_{i=1}^x \text{len}(C'_i)$. The ranking mechanism has to prioritize this list of segmentation solutions. The *Reader* mode has a dovetailing mechanism that produces the list of all segmentation solutions, but the ranking is within the dovetailing algorithm which is executed after the chunk segmentations are produced. This slows down the entire segmentation, and when deployed as a web service, produces a very long page, taking up more space and time, ultimately resulting into choking of the server.

Let us consider a segment $(c'_{i,j,k})$ from 5.2, with x being the number of chunks, y being the number of chunk segmentations in the i^{th} chunk, and z being the number of segments in the j^{th} chunk-segmentation. Each segment has three attributes, namely *word* $w_{i,j,k}$ (segmented form), *phase* $p_{i,j,k}$ (POS-tag), and *transition* $t_{i,j,k}$ (*sandhi* rule). The word attribute contains the segmented (inflected) form (*pada*), where the terminal *sandhi* is handled. Collecting only the words of a solution results into the surface forms, which can also be referred to as *pada-pāṭha*.²

The transition reflects the rule in 5.1 where u denotes the final part of the segment $c'_{i,j,k}$ and v denotes the initial part of the segment $c'_{i,j,k+1}$, where $k < z$. If $k = z$, v denotes the first part of the segment $s_{i+1,*,1}$ which belongs to one of the chunk segmentations of the next chunk. If $i = x$ and $k = z$, essentially the last segment of the last chunk, then an empty transition is assigned. More importantly, a transition is assigned only when w is non-empty and does not denote any elision. Thus for all mere concatenations, the transition is obtained as empty.

Additionally, the morphological analysis $m_{i,j,k,l}$ of the segment is obtained using the phase and word. For the given phase, its corresponding lexical map is traversed using the given word and the decorations stored for the word form give the morphological tags of the word. For the word *rāmaḥ*, we get “Masculine, Singular, Nominative” as the tags. The introduction of the index l is to address the multiple morphological

² This is different from the Vedic *pada-pāṭha* which is one of the *prakṛti-pāṭhas* (*saṃhitā*, *pada*, *krama*) of the Vedas. The Vedic *pada-pāṭha* is a device that has, in addition to the segmentation of a *mantra*, various other features like compound word indicators (using *avagraha*), indicators disambiguating the usage of some *taddhita*-suffix with certain *sup*-suffixes, end of mantra indicators, etc. In our case, we will call the collection of the segmented words alone as the *padapāṭha* of the unsegmented text without considering such extra information.

analyses (commonly called as multi-tags in SH, due to the phenomena of homonymy and syncretism) encapsulated under a single phase (like *vanam*).

We can thus use $w_{i,j,k}$, $t_{i,j,k}$ and $m_{i,j,k,l}$, respectively for extracting the frequencies for the corresponding word, transition and morphological analysis.

5.1.3 Preparing the data structure for the frequencies

The alignment process provided us with the parallel corpus of unsegmented and segmented sentences, and for each entry in this corpus, the word, base, inflectional analysis, derived stem and derivational analysis of every segment were also obtained. Six sets of frequency lists were extracted as presented in table 5.3.

Table 5.3: Frequency lists obtained from the Alignment datasets

type	unique entries	total word references
pada ¹	27,704	284,930
comp ²	16,130	118,303
pada_trans ³	784	280,622
comp_trans ⁴	381	78,907
pada_morph ⁵	33,000	291,751
comp_morph ⁶	18,770	118,958

¹ word frequencies

² frequencies of the compound components; obtained mainly to distinguish between the *sandhi* across words and the *sandhi* within a compound word.

³ *sandhi* across words; contains the tuple $\langle u, v, w \rangle$

⁴ *sandhi* across compound components

⁵ morphological analysis of words; these contain the tuple of $\langle \text{base, inflectional analysis, derived stem and derivational analysis} \rangle$

⁶ morphological analysis of compounds; similar to 5

In order to maintain consistency in the data structures used in SH, the word and morph data were encoded in decorated trie structures. For the word and compound component frequencies, the frequencies act as the decorations over the word forms. For the morph frequencies, the tuple of $\langle \text{inflectional analysis, base, derivational analysis and frequency} \rangle$ were the decorations over the stem. This trie structure is used mainly to share the common suffixes in words. Although sharing makes less sense in stems, the morph and their frequencies were still stored in this compact structure, mainly for consistency. On the other hand, since the transition types are less in num-

ber and size, a simple hash table was enough to store the three parts and their corresponding frequencies. SH represents each of the characters of Sanskrit using integers for internal processing. Hence these transition parameters were converted to their corresponding integers. The results of the three alignments, along with the Hackathon dataset, were considered for generating these frequencies and encoding them in the data structures as described earlier.

In addition to the alignment dataset, three more datasets were considered for a comparison.

1. SHMT dataset
2. SIGHUM's overall dataset (107,000 aligned sentences)
3. SIGHUM's sentences run over the new alignment algorithm (70,000 aligned sentences)

5.1.4 Variations in ranking metrics

Now that we have the chunk-segmentations for all the chunks and the frequencies from the corpus, we now turn to the ranking metrics. Given the set of all words $w_{i,j,k}$, let us define the first metrics as the unigram probabilities of the words, where we calculate the product of the unigram probabilities of all the segments in the chunk. Thus the joint unigram probability of the j^{th} segmentation of the i^{th} chunk, can be represented by the following equation:

$$P_{i,j} = \prod_{k=1}^z P_{w_{i,j,k}} \quad (5.4)$$

where $P_{w_{i,j,k}}$ is the unigram word probability of the k^{th} segment. In this metrics, all the information about phase (part of speech) or morphological analysis or transition are discarded and only the words are considered, with an assumption that, at the level of segmentation, the morphological analysis or transition do not impact the *sandhi* joining. This assumption notes that although the *sandhi* rules are used for performing the *sandhi* operation, during the reverse process of segmentation, these do not play much of a significant role beyond the resolution of the *sandhi*.

The second metrics considers the word probability along with the bigram probabilities of the transitions and a cumulative product across all the segments of the chunk was calculated as:

$$P_{i,j} = \prod_{k=1}^z P_{w_{i,j,k}} \times P_{t_{i,j,k}} \quad (5.5)$$

where $P_{t_{i,j,k}}$ is the *sandhi* probability of the k^{th} segment. The intuition behind considering both the word probabilities and the transition probabilities was to check if the transition probability introduces any significant improvement when compared with the first metrics. This lifts off the restriction imposed earlier by introducing the transition information as well.

The third metrics involves the joint probability based on the stem-morphological analysis frequencies:

$$P_{i,j} = \prod_{k=1}^z \max_{1 \leq l \leq t} P_{m_{i,j,k,l}} \quad (5.6)$$

where, $P_{m_{i,j,k,l}}$ is the unigram probability of the tuple $\langle \text{stem, inflectional morphological analysis, base-stem, derivational morphological analysis} \rangle$ corresponding to the l^{th} morphological analysis in the k^{th} segment. t is the total number of multi-tags for the segment. There are two ways to calculate the final probability of the segment: either to take all the multi-tags together representing the entire phase's probability, or to consider the most probable amongst the multi-tags of a phase (represented by the one with the highest frequency). We will choose the second approach. Here, the word parameter is implicit in the morphological analysis tuple, as we always find a one-one correspondence of the morph tuple with the word.

It is important to also note here the levels of analyses presented by SH. Some of these were already discussed during the differences between DCS and SH representations in Chapter 3. We can observe this hierarchy of the levels of analysis: word < phase < morphological analysis. As we move up the order, we get a more fine grained analysis. For example, the word *gacchati* represents two phases: Verb and Krid (primary derivative). The Krid phase describes the derivation but proposes two inflectional analysis, namely “n. sg. loc.” and “m. sg. loc.”, the difference being the gender in the morphological analysis.

These metrics of joint probabilities are similar to First Order Markov Model except that the segments are not states and hence the probability of the segment does not depend on the previous segment. What follows is the ranking algorithm where the solutions with higher joint probabilities combined with the least number of segments are ranked higher.

5.1.5 Ranking Algorithm

The ranking algorithm starts with assigning the probabilities for each of the chunk segmentations, and then with a dovetailing mechanism, prepares a list of best n segmentation solutions where n can be adjusted internally.³ It was observed earlier that the ranking and dovetailing mechanism used in the *Reader* is slow, primitive, where the ranking criteria considers approximate *kāraka* analysis, and applicable to short and sentences devoid of huge compounds.

Initial Observations with SHMT

The initial experiments were done on the segmentation solutions proposed by the *Reader*. For a sentence, S , it produces the set of all segmentation solutions, where the i^{th} solution can be represented as $S'_i = \langle s_{i,1}, s_{i,2}, \dots, s_{i,x} \rangle$. Each segment $s_{i,j}$ ($j \leq x$) has three attributes: word $w_{i,j}$, phase $p_{i,j}$ and transition $t_{i,j}$. The second metrics (eq. 5.5) was considered for the ranking and the SHMT frequencies were used to calculate the joint unigram probabilities. A set of 21,127 test sentences (from the SHMT corpus) was run on SH's Reader, 19,494 of which were recognized by SH. Evaluation was done based on a sentence-level perfect matching metric. While the SH without the ranking produced the correct solution in the first rank for 53.51% of the test sentences, the ranking pushed it to 89.27%. Although this looks promising, we should note that the test sentences predominantly contain short and simple sandhied expressions.

Dovetailing Mechanism

For every segmentation solution identified for a chunk, we get the triplet of word, phase and transition for each of the segments of the solution. These attributes are stored in a two dimensional structure, D , where the rows correspond to chunks and

³ 1, 5, 10, 50, 100, or depending on the requirement.

each row would have the list of segmentation solutions for that particular chunk, thus justifying the segment representation $P_{w_{i,j,k}}$. For every segment, the unigram probability is extracted and saved in D , depending upon the metrics chosen. In case the segment is unrecognized or the training set does not contain a particular word, the value of $\frac{1}{C_{ref}+C_{typ}}$ is assigned as the unigram probability, where C_{ref} is the total number word references and C_{typ} is the total number of unique words in the training set. Similarly, for the morph metrics, C_{ref} is the total number of (stem, inflectional morph, base, derivational morph) tuples, and C_{typ} is the number of unique tuples.

The cumulative probabilities, along with the cumulative segmentation (string) for every chunk segmentation are calculated. The chunk segmentations are inserted into D according to the chunk probability and number of segments, so that the structure always has the chunk segmentations of a particular chunk in the order of higher probability and lesser number of segments in the segmentation. When all the chunks are processed, we get the complete structure D , with chunk-segmentation, chunk-probability, chunk segments and their attributes.

A dovetailing traversal algorithm⁴ is introduced, where all possible combinations of the chunk segmentations from D are generated resulting into a list of segmentation solutions, with a limit to the number of solutions (n). These segmentation solutions are prioritized based on the overall joint probability calculated as the product of the joint probabilities of the chunk segmentations considered for the solution. In this way, the chunk segmentations of eq. 5.2 are converted into the list of most probable segmentations of eq. 5.3. This list of n solutions contains: (segmentation, joint probability, segments) where the segments further contain the attributes. These are passed on to the third stage, display routine which enlists the segmentation solutions. If $n = 1$, then instead of the dovetailing mechanism, the first chunk segmentations of all the chunks are directly merged to form the most probable segmentation solution.

Thus, in addition to the two steps of the segmentation, two more steps had been added to rank the solutions.

1. Constructing the list of chunk segmentations with the segmentation strings, joint probabilities and segments with their attributes, and

⁴ Similar to a simple breadth-first traversal.

2. Dovetailing mechanism over the list of chunk segmentations to keep a list of best segmentations.

This makes sure that the less probable segments are omitted and only the most probable segments are considered for the final result. The same was applied with the newly created datasets (Alignments 2 and 3 and also the Hackathon dataset).

5.2 Evaluation

Three strategies were considered for evaluating the segmentation results of all these experiments: sentence-level, word-level and compound-based. In the sentence-level strategy, the performance is measured in terms of the number of complete matches of the sentences. In the word-level strategy, the macro-averaged precision, recall and F-score were taken into consideration. This involves a counter-based evaluation strategy where the counter of each of the ground truth segments (words) of a solution are compared with the counter of each of the predicted segments. Then the macro-averaged precision, recall, etc. over all the solutions are computed.⁵

In the compound-based strategy, the ranking metrics are evaluated using two parameters: *perfect match with compounds* and *number of compound matches*. The first one is a much stricter metrics where it is mandated that the predictions have the exact words and compounds (in the expected compositionality). The second one calculates the mean of the number of compounds correctly predicted. These parameters help in addressing how efficiently the metrics identify compound boundaries, in addition to resolving *sandhi* and identification of words and compound components.

Additionally, to compare the performance of the ranking metrics, two more metrics were used: Mean Rank (MR) and Mean Reciprocal Rank (MRR). And ranking was limited to the first 100 ranks when the word and word-transition metrics were used, and to the first 200 ranks when the stem-morph metrics was used. SH considers the differences in morphological analysis while generating the segmentation solutions. In the word and word-transition metrics, all the solutions having the same word-forms but different morphological analyses were collapsed together, ignoring the morphological analyses and orienting the whole outcome with segmentation-only results. While us-

⁵ The code and results of the evaluation are made available here: [sanskrit_segmentation_evaluation](#)

ing the frequencies of stem and morphological analysis, these identical segmentations which differ only in morphological analyses were also collapsed to have an equivalent comparison with the other metrics, even though it would mean losing their main criteria for ranking. In order to account for this, the ranking was limited to a larger window (200) to accept identical segmentations differing only in morphological analyses but having ranks beyond the 100th position.

5.3 Observations

The following texts were considered for the experiments on the segmentation models: *Bhagavad Gītā*, *Meghadūta*, *San̄kṣepa-Rāmāyaṇa* and test set of Hellwig and Nehrdich (2018). Additionally, *Kathāsaritsāgara*, a true unseen data from the perspective of both DCS and SH, was also chosen in order to have a fairer comparison. *Bhagavad Gītā* and *Meghadūta* were used as development corpora to update SH’s segmenter and also to validate the gold standards.

First, the existing datasets (SHMT, SIGHUM, SIGHUM* sentences re-aligned with the new alignment algorithm) are considered for generating the statistics for SH-Ranking. A comparison of SH-Ranking’s performance on a sample set (*Bhagavad Gītā* Chapter 3) using these datasets is done with the Alignment 1 dataset (partial), shown in table 5.4. We can observe the superior performance of SHMT over the other datasets. With more data available from the alignments, we can get better results.

Out of the 43 verses, the correct solution was found in the first rank for 10 verses when the SHMT dataset was used. And for 11, 9 and 5 verses respectively when the other three datasets (Alignment 1, SIGHUM*, SIGHUM) were used. Similarly, the number of verses for which the correct solution was found in ranks 2, 3, 4, 5 and 6-10 are reported in the table. The percentage of verses for which SH-Ranking proposes the correct solution within the top 10 ranks is also noted.

Thus a sentence-level evaluation on each of the selected test sets was done with the alignment datasets, followed by a comparison with the word-level evaluation. The performance on the unseen data is discussed further, followed by the evaluation based on the compound boundary identification and finally the evaluation for ranking.

Solution Rank	SHMT	Alignment 1	SIGHUM* ¹	SIGHUM
1	10	11	9	5
2	5	2	2	0
3	2	1	1	3
4	2	1	0	1
5	2	1	1	0
6-10	2	4	4	4
Coverage	53.48%	46.5%	30.7%	39.53%

¹ The sentences from SIGHUM were re-aligned with the new alignment algorithm

Table 5.4: **Performance Comparison of SH-Ranking with four datasets (SHMT, Alignment 1, SIGHUM sentences re-aligned with the new alignment algorithm, and SIGHUM sentences) on Bhagavad Gītā Chapter 3 verses**

Column 1 (Solution Rank) contains the rank of the correct solution. Column 2 contains the number of verses for which the correct solution was found in the rank (of column 1) when SHMT dataset was used to create the frequencies for the ranking algorithm. Similarly, column 3 contains the numbers when Alignment 1 dataset was used, column 4 contains the numbers when SIGHUM sentences re-aligned with the new alignment algorithm was used, and column 5 contains the numbers when the original SIGHUM dataset was used for the ranking. The overall accuracy (coverage) is also reported.

5.3.1 Sentence-level Evaluation

The main purpose of this evaluation was two fold. First, to identify the better metrics among the three viz. word and transition, word only and stem-morph, for ranking the solutions in SH. Second, to compare the performance of SH with rcNN (Hellwig and Nehrdich, 2018).⁶ In order to address the issue of compositionality, we augmented the gold data with both the compositional as well non-compositional analyses of compounds.

The performance comparison of SH-Ranking (with the word and stem-morph metrics from all the four alignments) and rcNN on the verses from *Bhagavad Gītā* (Vyāsa, 2007), *Meghadūta* (Kālidāsa, 1934) and *Saṅkṣepa-Rāmāyaṇa* (Vālmiki, 2002), and the test sentences from Hellwig and Nehrdich (2018) (henceforth referred to as rcNN-test-set) are recorded in table 5.5.⁷ The results of word-transition metrics is not reported as it has a comparatively low performance.⁸

⁶ Character-based recurrent and convolutional neural network model

⁷ The results after ignoring the compositionality of compounds are recorded in the table.

⁸ This was observed during an initial run of *Bhagavad Gītā* verses on all the three metrics where the number of sentences correctly predicted with word-transition metrics was less than that with the other two metrics by a margin of 100 sentences or more.

Table 5.5: Sentence-level performance comparison of *Bhagavad Gītā*, *Meghadūta*, *Saṅkṣepa-Rāmāyaṇa* and rcNN-test-set on SH-Ranking and rcNN models

		SH-Ranking								rcNN ⁵
		A1 ¹		A2 ²		A3 ³		A1H ⁴		
		m ⁶	w ⁶	m	w	m	w	m	w	
BG ⁸ (692)	C ¹³	93.50	96.97	94.06	97.25	93.21	<u>96.97</u>	93.93	96.97	58.23
	R-1 ¹⁴	71.53	74.27	75.29	78.47	75.00	<u>78.47</u>	68.79	70.81	58.23
MD ⁹ (230)	C	<u>88.70</u>	91.30	88.70	90.87	88.70	90.87	87.40	90.97	40.00
	R-1	<u>78.70</u>	80.43	71.30	74.35	73.04	75.22	71.30	71.30	40.00
SR ¹⁰ (84)	C	72.62	72.62	72.62	72.62	72.62	72.62	72.62	72.62	69.05
	R-1	54.76	54.76	54.76	<u>55.95</u>	53.57	53.57	<u>55.95</u>	<u>55.95</u>	69.05
r(a) ¹¹ (18,404)	C	54.80	54.87	54.79	54.88	54.78	54.76	54.87	<u>54.96</u>	82.63
	R-1	47.77	48.16	48.86	49.61	49.20	<u>49.90</u>	47.21	47.67	82.63
r(s) ¹² (597)	C	80.90	81.07	80.90	<u>81.07</u>	90.90	<u>81.07</u>	80.90	81.07	67.34
	R-1	68.68	71.02	68.51	<u>70.35</u>	68.68	<u>70.35</u>	68.17	70.02	67.34

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ rcNN - The character-based Recurrent and Convolutional neural network model by Hellwig and Nehrdich (2018)

⁶ m - morph-metrics

⁷ w - word-metrics

⁸ BG - *Bhagavad Gītā*

⁹ MD - *Meghadūta*

¹⁰ SR - *Saṅkṣepa-Rāmāyaṇa*

¹¹ r(a) - rcNN-test-set (Hellwig and Nehrdich, 2018)

¹² r(s) - A sample of rcNN-test-set

¹³ C - Coverage or Overall accuracy regardless of the rank or position

¹⁴ R-1 - Rank-1 Accuracy indicating the accuracy considering rank 1 alone

Rank-1 Accuracy (i.e., the proportion of inputs where the correct segmentation is ranked first) and Coverage (i.e., the proportion of inputs for which the correct segmentation is found anywhere in the ranked list, regardless of position) are reported in the table. The columns indicate the models used and the rows indicate the test sets. SH-Ranking is tested in two metrics: word and stem-morph. And four different datasets were used to generate the frequencies for the ranking: A1, A2, A3 and A1H (Hackathon dataset).

Bhagavad Gītā

In all the four alignments, the word-metrics performs the best when considering the correct solution in the first rank. When the stem-morph frequencies are used, there is a minor decrease in the correct segmentation at the first position (when compared to the word metrics). Despite collapsing the identical segmentations together, by ignoring the differences due to morphological analyses, the performance of the stem-morph frequencies in the first rank for the four datasets is comparatively lesser.

These observations show that for the word segmentation task, the stem and morphological information are in the first place required for validating the segments, but these do not contribute much towards ranking in comparison with the words. The stem-morph frequencies are best used when considering the task of morphological parsing rather than word segmentation alone.

The initial observations are based on the *Bhagavad Gītā* (Vyāsa, 2007) corpus of 700 verses which is available in both the segmented and unsegmented forms.

In the first run of the 700 verses, with the word-transition metrics, it was observed that the Rank-100 accuracy was 49.29. 94 verses had unrecognized chunks and for the remaining verses, the correct segmentation was either ranked beyond the 100th position or not produced at all. For handling the unrecognized chunks, SH lexicon was updated to recognize them. For the remaining verses it was observed that while SH produced compositional analyses of compounds, the gold segmentation had non-compositional analyses. Hence, the gold segmentations were augmented with compositional analyses of compounds.

rcNN on *Bhagavad Gītā*: A comparison of the rcNN model with SH-Ranking is carried out to understand the performance differences and other intricacies in the two models. Since rcNN produces only one solution while SH-Ranking produces the top n solutions, only the first solution from SH is considered for comparison.

The rcNN model recorded a correct solution for 217 of the 700 verses of *Bhagavad Gītā*. Using a post processing module for rcNN output,⁹ and considering both the compositional and non-compositional analyses as correct segmentations, the rcNN model recorded correct solutions for 403 out of 700 verses.

⁹ This post-processing module was introduced to handle terminal sandhis and homonasal consonants by modifying specific characters according to the format of the ground truth analysis.

Meghadūta

Meghadūta, is written in two parts containing 121 verses together. These were divided into two halves (hemistichs) resulting into 242 hemistichs. Excluding the 12 hemistichs having unrecognized chunks by SH, SH-Ranking (with word metrics using the Alignment 1 dataset) performed the best by producing correct segmentations for 210 hemistichs and 185 of them in first position. And rcNN was able to predict the correct segmentations for 92 hemistichs.

Saṅkṣepa-Rāmāyaṇa

Saṅkṣepa-Rāmāyaṇa is an abridged version of the epic *Rāmāyaṇa*, written by *Vālmīki*, describing the original story in just 100 verses. Here also, we excluded the 16 verses that had chunks unrecognized by SH. All the SH-Ranking metrics produced the correct segmentations for 61 verses. Of these, we find almost similar results in the first position, with a difference of just one or two sentences. rcNN produced the correct segmentations for 58 verses.

rcNN-test-set

The test set of Hellwig and Nehrdich (2018) contains over 21,000 sentences from DCS. Out of these, 18,404 could pass through SH and these were used for the comparison. rcNN outperforms all the metrics by a relatively larger margin. The main reason for the under-performance of SH-Ranking owes to the compositionality of compounds. The gold segmentations were obtained from DCS and they contain non-compositional analyses when the context requires it while SH predominantly produces compositional analyses. Since it is tedious to work on each of these sentences and modify the gold segmentations to accept both compositional and non-compositional analyses, a sample set of the test sentences (597) was taken into consideration, and was augmented with compositional analyses. The SH-Ranking produced correct segmentations for 484 sentences of which 424 were in the first position, and rcNN produced correct segmentations for 402 sentences.

Comparison of SH-Ranking with TransLIST

Sandhan et al. (2022) provided a performance comparison of rcNN and TransLIST along with various other models using two datasets: SIGHUM and Hackathon. It was observed that with both the datasets TransLIST outperformed the remaining models. SH-Ranking was tested over the Hackathon dataset. Hackathon dataset contains 90,000, 10,332 and 9,963 sentences as train, dev and test set, respectively. The 90,000 sentences from the parallel corpus of the training set were used to create the frequencies for the ranking. 9,910 sentences from the test set were considered for the performance comparison, and the sentence-based comparison is recorded in table 5.6. The morph metrics performed the best producing the correct solution for 8,430 sentences and edged the TransLIST model by 0.3%.

Table 5.6: Sentence level performance comparison of SH-Ranking, rcNN and TransLIST on the test set of Hackathon Dataset (9,910 sentences)

	SH-Ranking ¹		rcNN	TransLIST
	m ²	w ³		
Coverage	<u>94.54</u>	94.56	83.80	84.77
Rank-1 Accuracy	85.07	<u>84.79</u>	83.80	84.77

The best performing metrics / model are in bold and the second best are underlined.

¹ Alignment with Hackathon dataset

² m - morph-metrics

³ w - word-metrics

5.3.2 Word-level Evaluation

Table 5.7 gives a summary of the word-level evaluation results of SH-Ranking (with the word based and stem-morph based metrics) and rcNN over *Bhagavad Gītā*, *Meghadūta*, *San̥kṣepa-Rāmāyaṇa* and rcNN-test-set. Similar to table 5.5, the columns indicate the models (SH-Ranking and rcNN) and the rows indicate the test sets. The SH-Ranking is tested using two metrics (word and stem-morph) with the four datasets previously obtained.

Alignment 3 and alignment 1 frequencies performed the best for *Bhagavad Gītā* and *Meghadūta*, respectively, while the rcNN model performed the best for *San̥kṣepa*

Table 5.7: Word-level evaluation of SH-Ranking and rcNN models on *Bhagavad Gītā*, *Meghadūta*, *Saṅkṣepa-Rāmāyaṇa* and rcNN-test-set

		SH-Ranking								rcNN
		A1 ¹		A2 ²		A3 ³		A1H ⁴		
		m ⁵	w ⁶	m	w	m	w	m	w	
BG ⁷	P ¹²	97.75	97.79	98.08	<u>98.14</u>	98.10	98.16	97.35	97.37	95.45
	R ¹³	97.17	97.25	97.54	<u>97.64</u>	97.59	97.66	96.69	96.78	94.61
	F ¹⁴	97.45	97.51	97.80	<u>97.87</u>	97.83	97.89	97.00	97.06	94.98
MD ⁸	P	97.80	<u>97.74</u>	96.71	97.07	96.82	97.12	96.95	96.52	92.11
	R	97.98	<u>97.85</u>	97.14	97.34	97.23	97.39	97.14	97.76	90.97
	F	97.87	<u>97.78</u>	96.90	97.19	97.00	97.24	97.02	96.62	91.48
SR ⁹	P	94.05	93.14	93.49	93.17	93.59	92.61	<u>94.14</u>	93.39	96.22
	R	<u>94.86</u>	94.19	94.56	94.34	94.66	93.84	94.71	94.29	96.30
	F	<u>94.39</u>	93.60	93.96	93.69	94.06	93.16	94.37	93.77	96.23
r(a) ¹⁰	P	85.28	85.34	85.73	85.93	85.78	<u>85.96</u>	85.14	85.24	96.09
	R	87.83	87.97	88.34	88.58	88.40	<u>88.64</u>	87.62	87.80	96.37
	F	86.31	86.41	86.80	87.02	86.85	<u>87.06</u>	86.14	86.28	96.17
r(s) ¹¹	P	91.90	92.14	91.93	<u>92.24</u>	91.95	92.21	91.83	92.06	93.86
	R	93.45	93.71	93.51	<u>93.80</u>	93.52	93.78	93.38	93.60	94.61
	F	92.59	92.84	92.63	<u>92.93</u>	92.65	92.91	92.52	92.75	94.10

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ m - morph-metrics

⁶ w - word-metrics

⁷ BG - *Bhagavad Gītā*

⁸ MD - *Meghadūta*

⁹ SR - *Saṅkṣepa-Rāmāyaṇa*

¹⁰ r(a) - rcNN-test-set (all)

¹¹ r(s) - rcNN-test-set (sample)

¹² P - Precision

¹³ R - Recall

¹⁴ F - F-Score

Rāmāyaṇa. For the rcNN-test-set, the rcNN model outperformed the remaining with a huge margin. For the sample set considered from rcNN-test-set, accepting compositional analyses, the rcNN model still performed better followed by the SH-Ranking with alignment 2 dataset with a difference of less than 1.5%.

Table 5.8 gives the comparison of SH-Ranking, rcNN and TransLIST over the test set of Hackathon dataset. The morph metrics of the SH-Ranking edged over rcNN and TransLIST in terms of precision while rcNN had better recall and F-score values.

Table 5.8: Word-level evaluation of SH-Ranking, rcNN and TransLIST models on the Hackathon dataset

	SH-Ranking ¹		rcNN	TransLIST
	m ²	w ³		
P ⁴	97.43	<u>97.39</u>	97.33	97.31
R ⁵	<u>97.29</u>	97.25	98.06	97.30
F ⁶	<u>97.34</u>	97.30	97.59	97.29

The best performing metrics / model are in bold and the second best are underlined.

¹ Alignment with Hackathon dataset

² m - morph-metrics

³ w - word-metrics

⁴ P - Precision

⁵ R - Recall

⁶ F - F-Score

Upon comparing the performances of the four alignment datasets, two important observations emerge. From the sentence-level evaluation, it was noticed that the average difference between the maximum and minimum values of correct segmentations in the first position, calculated across the test sets, was 5.56%. This finding highlights the significant impact of having more data, as it effectively pushes up the correct solutions towards the top rank.

However, when considering the precision values from the word-based evaluation across the alignment datasets, the average difference (percentage) between the maximum and minimum values was only 0.99%. Similarly, for Recall and F-Score, the average differences between the maximum and minimum values were 0.96% and 0.97%, respectively. These results suggest that while expanding the datasets has not led to a substantial improvement in performance, the primary objective of elevating the correct solutions to higher rankings has been achieved.

5.3.3 Performance on true unseen data

For a fair performance comparison of SH-Ranking with rcNN and TransLIST, these models are to be trained on each of the datasets and then the evaluation should be done on a held-out test set. Due to lack of similar hardware requirements, this could not be achieved. In order to evaluate the models on a truly unseen data, the 13th *lambaka* of *Kathāsaritsāgara*, composed by *Somadeva Bhaṭṭa*, was considered. The e-text of *Kathāsaritsāgara*, *lambaka* 13, was taken from an online resource (Gretil).¹⁰ The total number of sentences (containing the hemistichs of all the verses (438) along with the prologue and epilogue of the *lambaka*) is 445. To check the correctness of Gretil's version, a comparison was done with a printed version of *Kathāsaritsāgara* (Bhatta, 1960), which resulted in corrections for 56 sentences.

Annotation: Since this text did not have any gold segmentations, an annotation task was carried out with three different annotators. The annotators were given instructions to segment the sentences along with the marking of compound boundaries. Out of the 445 sentences, Annotators 1 and 2 had 205 identical annotations¹¹ while 1 and 3 had 175 annotations in common. Annotators 2 and 3 had 155 annotations in common. The inter-annotator agreement (IAA) was identified using Cohen's Kappa score, which was calculated for two cases:

1. whether the annotators agree to resolve *sandhi* or not, and
2. whether the annotators agree on explicitly marking the compound splits or not.

The Kappa scores, shown in table 5.9, indicate that the three annotators have an almost perfect agreement in both resolving *sandhi* as well as marking the compound splits.

Two setups for evaluation were prepared. While in the first setup, each of the annotations individually was considered as gold, in the second setup, all the annotations together were considered as gold and the evaluation was run. This was to make sure that the evaluations are not biased towards any particular annotation. The observations of the sentence-level evaluation are reported in table 5.10, which also records

¹⁰ The link for the Gretil resource can be accessed from here: *Kathāsaritsāgara Gretil*

¹¹ This represents the number of sentences where a match is found for the entire sentence.

Table 5.9: Inter-Annotator Agreement between the three Annotators using Kappa Scores

IAA Case	Annotator Pairs		
	1-2	2-3	1-3
Sandhi	0.927	0.915	0.947
Compound	0.804	0.763	0.83

the evaluation over the best annotation and the evaluation on all annotations.¹² The columns indicate the segmentation models (SH-Ranking with the two metrics and four datasets, and the rcNN model) and the rows indicate the two evaluation setups.

Table 5.10: Sentence-level evaluation of SH-Ranking and rcNN models on *Kathāsarit-sāgara (lambaka 13)*, where for the SH-Ranking metrics, only the sentences recognized by SH were considered, but for rcNN, additionally all the sentences were considered

		SH-Ranking								rcNN
		A1 ¹		A2 ²		A3 ³		A1H ⁴		
		m ⁵	w ⁶	m	w	m	w	m	w	
C1 ⁷	Cv ⁹	60.45	60.22	61.12	<u>60.90</u>	61.12	<u>60.90</u>	60.00	60.00	59.55
	R-1 ¹⁰	49.21	51.01	50.11	<u>51.46</u>	49.89	51.01	46.74	47.42	59.55
C2 ⁸	Cv	45.84	45.17	46.29	45.62	<u>46.07</u>	45.62	44.94	44.94	42.02
	R-1	33.93	35.05	34.83	<u>35.28</u>	34.61	35.06	31.69	31.91	42.02

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ m - morph-metrics

⁶ w - word-metrics

⁷ C1 - All the three annotations are considered as gold

⁸ C2 - Only the best annotation is considered as gold

⁹ Cv - Coverage: Number of correct solutions obtained

¹⁰ R-1 - Rank-1 Accuracy: Number of correct solutions in the first rank (applicable to SH-Ranking results)

We observe a significant difference between the Rank-1 accuracy scores of the best performing metrics of SH-Ranking and the rcNN (8.01 and 6.47 in the two setups). In the current scenario, the evaluation is done on all the sentences which also includes

¹² The comparison was done on the various metrics of SH-Ranking and rcNN but not with TransLIST as TransLIST had errors in its code.

the sentences that have at least one chunk unrecognized by SH, amounting to 65 sentences or 14.61%. If we exclude the sentences unrecognized by SH, the number of correctly predicted segmentations of SH-Ranking does not change at all while rcNN observes a significant change in the numbers.¹³ Addressing SH to handle the unrecognized chunks corresponding to the 65 sentences would help increase the scores of SH-Ranking.

Table 5.11 records the word-level evaluation of *Kathāsaritsāgara* (*lambaka 13*). Similar to the other tables, the columns indicate the segmentation models (SH-Ranking with two metrics using 4 different datasets, and rcNN), and the rows indicate four different scenarios in which the evaluation was carried out:

1. C1 - All the three annotations are considered as gold. SH-unrecognized sentences are included.
2. C2 - Only the best annotation is considered as gold. SH-unrecognized sentences are included.
3. C3 - All the three annotations are considered as gold. SH-unrecognized sentences are excluded.
4. C4 - Only the best annotation is considered as gold. SH-unrecognized sentences are excluded.

rcNN outperforms the SH-Ranking metrics by an average F-score of 4.39% when considering all the sentences (see C1 and C2). Ignoring the sentences which are unrecognized by SH resulted in a competing performance where the rcNN edges over SH-Ranking by a margin of 0.44% on average F-score (see C3).

5.3.4 Evaluating the Ranking Algorithm

Identifying Compound Boundaries

One advantage of SH is its ability to explicitly mark the compound boundaries which are resolved during the overall segmentation process. Thus with the third evaluation,

¹³ In all the previous experiments, we excluded the sentences having chunks unrecognized by SH. In order to have a fair comparison of the models, such sentences were included during the experiments with *Kathāsaritsāgara*.

Table 5.11: Word-level evaluation of SH-Ranking and rcNN models on *Kathāsaritsāgara* (*lambaka* 13)

		SH-Ranking								rcNN
		A1 ¹		A2 ²		A3 ³		A1H ⁴		
		m ⁵	w ⁶	m	w	m	w	m	w	
C1 ⁷	P ¹¹	86.67	86.69	86.71	<u>86.84</u>	86.71	86.76	85.62	85.55	91.64
	R ¹²	85.67	85.68	85.78	<u>85.90</u>	85.79	85.82	84.60	84.52	91.75
	F ¹³	85.86	85.87	85.93	<u>86.06</u>	85.93	85.98	84.80	84.73	91.58
C2 ⁸	P	<u>81.79</u>	81.75	81.60	81.68	81.70	81.71	80.83	80.63	86.43
	R	81.06	81.06	80.96	81.08	81.07	<u>81.12</u>	80.03	79.86	86.10
	F	<u>81.06</u>	81.03	80.91	81.01	81.01	81.04	80.06	79.88	86.07
C3 ⁹	P	90.38	90.37	90.51	<u>90.57</u>	90.54	90.51	89.41	89.30	91.39
	R	91.33	91.29	91.52	<u>91.55</u>	91.55	91.49	90.28	90.15	91.65
	F	90.76	90.73	90.92	<u>90.97</u>	90.95	90.91	89.75	89.63	91.40
C4 ¹⁰	P	<u>85.14</u>	85.06	85.05	85.05	85.13	85.06	84.20	83.94	86.63
	R	86.26	86.22	86.27	86.31	86.38	<u>86.34</u>	85.25	85.01	86.32
	F	85.54	85.48	85.50	85.02	<u>85.60</u>	85.54	84.56	84.32	86.28

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ m - morph-metrics

⁶ w - word-metrics

⁷ C1 - Considering all the sentences including those unrecognized by SH, and all the three annotations are considered as gold

⁸ C2 - Considering all the sentences including those unrecognized by SH, and only the best annotation is considered as gold

⁹ C3 - Considering all the sentences excluding those unrecognized by SH, and all the three annotations are considered as gold

¹⁰ C4 - Considering all the sentences excluding those unrecognized by SH, and only the best annotation is considered as gold

¹¹ P - Precision

¹² R - Recall

¹³ F - F-Score

the parameter *perfect match metrics* (*PM* (*c*)) uses a sentence-level evaluation with an additional constraint: marking the compound splits. And, the other parameter *compound matches* (*CM*) calculates the mean of the number of compounds matched correctly.

While rcNN and TransLIST identify the segments, and additionally rcNN marks the sandhi location, they do not mark the compound boundaries. Hence, this evaluation is carried out only for the ranking metrics. Table 5.12 records the PM (c) and CM observations. The rows indicate the test sets and the columns indicate the two metrics of SH-Ranking using the four different datasets. Additionally, for each test set, the evaluation is done considering all the sentences (all), and excluding the sentences which have at least one chunk unrecognized by SH (rec).

For the *Bhagavad Gītā*, all the ranking metrics had a strong competition both with and without unrecognized sentences. Let us consider Alignment 3's word metrics. We can observe that 4.35% of the sentences did not have a perfect match and 12.81% of the compounds are distributed across the 4.35% sentences. This can be observed in other test sets also. With the PM (c) metrics, including the unrecognized sentences does no good as the entire sentence is considered for evaluation. But CM metrics gives us a clear picture of SH's ability to predict the compounds. While the overall performance over *Bhagavad Gītā* and *Meghadūta* are reasonably good, the performance deteriorates with *Saṅkṣepa-Rāmāyaṇa*, and further goes down with *Kathāsaritsāgara (lambaka 13)*. This shows the influence of lexicon over the process of segmentation in SH. The good performance of *Bhagavad Gītā* and *Meghadūta* is attributed to the fact that they were used as development data.

Mean Rank and Mean Reciprocal Rank

The ranks of the segmentations and the rank distribution were considered during the sentence-level evaluation. In addition to these, evaluation based on Average Rank and Mean Reciprocal Rank are proposed here. Table 5.13 shows the evaluation based on AR and MRR. The closer the AR score is to 1.0, the better the ranking algorithm is, and MRR is represented as a percentage. The columns indicate the SH-Ranking metrics (word and stem-morph) using the four different datasets for generating the frequencies, and the rows indicate the test sets.

We can observe that Alignment 3 metrics performs the best on *Bhagavad Gītā* while Alignment 1 metrics performs the best on *Meghadūta*. For *Saṅkṣepa-Rāmāyaṇa* and *Kathāsaritsāgara (lambaka 13)*, both Alignments 2 and 3 perform equally well. We can also observe the nature of the text based on the average rank. *Kathāsaritsāgara*

Table 5.12: Compound evaluation of SH-Ranking metrics on *Bhagavad Gītā*, *Meghadūta*, *Śaṅkṣepa Rāmāyaṇa* and *Kathāsaritsāgara* (Iambaka 13)

			SH-Ranking							
			A1 ¹		A2 ²		A3 ³		A1H ⁴	
			m ⁵	w ⁶	m	w	m	w	m	w
BG ⁷	all ¹³	PM (c) ¹¹	95.05	95.51	94.94	95.53	95.04	<u>95.65</u>	95.26	96.12
	(700)	CM ¹²	86.70	85.72	<u>87.74</u>	86.82	88.49	87.09	85.30	85.33
	rec ¹⁴	PM (c)	96.69	97.09	96.56	<u>97.10</u>	96.68	97.23	96.40	97.23
	(692)	CM	87.27	86.38	<u>88.28</u>	87.39	88.97	87.67	85.39	85.49
MD ⁸	all	PM (c)	84.62	85.00	84.62	<u>84.94</u>	84.62	<u>84.94</u>	84.78	<u>84.94</u>
	(242)	CM	<u>91.94</u>	92.99	88.59	90.71	88.16	90.35	89.23	89.89
	rec	PM (c)	88.39	88.70	88.39	<u>88.65</u>	88.39	88.65	88.64	<u>88.65</u>
	(230)	CM	<u>93.30</u>	94.35	89.79	91.98	89.35	91.71	90.48	91.12
SR ⁹	all	PM (c)	<u>60.20</u>	59.00	<u>60.20</u>	59.00	<u>60.20</u>	59.00	60.82	59.00
	(100)	CM	82.22	81.83	83.08	82.75	<u>83.42</u>	83.58	83.08	82.25
	rec	PM (c)	<u>71.95</u>	70.24	<u>71.95</u>	70.24	71.95	70.24	72.84	70.24
	(84)	CM	85.98	85.52	<u>86.79</u>	86.90	86.38	85.32	84.98	84.42
KSS ¹⁰	all	PM (c)	57.24	57.08	<u>57.47</u>	57.53	<u>57.47</u>	57.43	56.79	56.63
	(445)	CM	68.25	69.33	70.81	71.16	70.25	<u>71.10</u>	66.86	66.29
	rec	PM (c)	67.11	66.84	67.37	67.37	67.37	<u>67.28</u>	66.58	66.32
	(380)	CM	71.93	73.16	74.93	75.04	74.27	<u>74.98</u>	70.95	69.96

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ m - morph-metrics

⁶ w - word-metrics

⁷ BG - *Bhagavad Gītā*

⁸ MD - *Meghadūta*

⁹ SR - *Śaṅkṣepa-Rāmāyaṇa*

¹⁰ KSS - *Kathāsaritsāgara* (Iambaka 13)

¹¹ PM (c) - Perfect Match with compounds

¹² CM - Compound Matches

¹³ all - Considering all the sentences including those unrecognized by SH

¹⁴ rec - Considering all the sentences excluding those unrecognized by SH

obtained the best rank (1.4) and it should have relatively less complex words and constructions which are used quite often and hence, most of the correct solutions are on the top. On the other hand, consider *Bhagavad Gītā*. In spite of using it as a develop-

Table 5.13: Ranking evaluation of SH-Ranking metrics on *Bhagavad Gītā*, *Meghadūta*, *Śaṅkṣepa-Rāmāyaṇa* and *Kathāsaritsāgara* (lambaka 13)

		SH-Ranking							
		A1 ¹		A2 ²		A3 ³		A1H ⁴	
		m ⁵	w ⁶	m	w	m	w	m	w
BG ⁷	MRR ¹¹	78.00	76.96	80.10	79.39	80.96	<u>80.12</u>	73.78	73.14
	AR ¹²	2.20	2.27	2.11	2.16	2.07	<u>2.10</u>	2.65	2.84
MD ⁸	MRR	<u>87.14</u>	87.45	80.68	82.79	81.53	83.28	82.34	79.78
	AR	1.55	<u>1.69</u>	1.95	2.00	1.97	1.99	1.79	2.24
SR ⁹	MRR	79.70	80.35	<u>83.84</u>	84.48	80.66	80.30	80.66	79.68
	AR	1.90	1.80	1.83	1.56	1.85	<u>1.73</u>	1.90	1.85
KSS ¹⁰	MRR	81.90	83.53	85.47	86.70	85.15	<u>86.45</u>	80.18	78.61
	AR	1.57	1.49	1.47	1.40	1.46	<u>1.42</u>	1.64	1.67

The best performing metrics / model are in bold and the second best are underlined.

¹ A1 - Alignment 1

² A2 - Alignment 2

³ A3 - Alignment 3

⁴ A1H - Alignment with Hackathon dataset

⁵ m - morph-metrics

⁶ w - word-metrics

⁷ BG - *Bhagavad Gītā*

⁸ MD - *Meghadūta*

⁹ SR - *Śaṅkṣepa-Rāmāyaṇa*

¹⁰ KSS - *Kathāsaritsāgara* (lambaka 13)

¹¹ MRR - Mean Reciprocal Rank

¹² AR - Average rank

ment corpus, the average rank comes to above 2. This indicates that the less probable words and constructions are present in the text.

5.4 Representation of the Ranked Solutions

SH represents the segmentation in two modes: *Summary* and *Reader*. The Summary mode produces the graphical interface (tabulated display) with all the segments and their corresponding features. The Reader mode enlists the segmentation solutions with the segments, their features and the transition that happens after them, but this mode is restricted due to time and space constraints. To leverage both the features of the

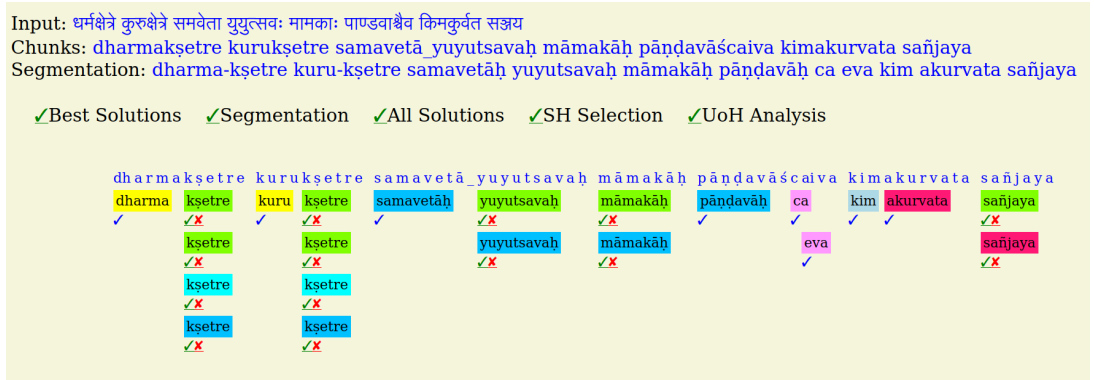
graphical interface and the list, the results of the ranking mechanism are displayed in two new modes: (1) *First*, and (2) *Best n*.

In the *first* mode, the segmentation results corresponding to the first solution are displayed. In the *best n* mode, the segmentation results corresponding to the top n solutions are displayed.

During the segmentation process, for each chunk segmentation produced, three features of each of the segments are collected: segment, phase, transition, which will be used later to generate the morphological analyses along with the multi-tags. During the ranking mechanism, while generating the list of overall segmentation solutions from these chunk segmentations, these features are cumulatively appended. In addition to it, the cumulative segmented string of the entire sentence and the cumulative joint probability of the entire segmentation solution are also generated. The joint probability depends on the metrics chosen. As of now, there are two: word and morph.¹⁴

Let us now consider the *first* mode, where the ranking is chosen with the word metrics, keeping n as 1. All the segments belonging to the first solution are collected, and their corresponding indices are sent to the tabulated display. This display is refreshed with the selected indices showing only the collected segments. For example, fig. 5.1 shows the segments of the first solution.

Figure 5.1: Ranking results with the “First solution” mode (graphical interface) using the word metrics



¹⁴ For an end user, only the word metrics is used, but the morph metrics and the word-transition metrics are present internally. However, we will discuss the results using the word and morph metrics. word-transition is ignored as its results were poor, and also the representation for word and word-transition metrics would be similar.

Figure 5.7: Ranking results with the “Best n solutions” mode using the morph metrics

Input: धर्मक्षेत्रे कुरुक्षेत्रे समवेता युयुत्सवः मामकाः पाण्डवाश्चैव किमकुर्वत सञ्जय
 Chunks: dharmakṣetre kurukṣetre samavetā yuyutsavaḥ māmakāḥ pāṇḍavāścaiva kimakurvata sañjaya

✓Best Solutions ✓All Solutions

dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	1 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	2 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	3 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	4 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	5 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	6 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	7 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	8 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	9 - UoH
dharma	ksetre	kuru	ksetre	samavetāḥ	yuyutsavaḥ	māmakāḥ	pāṇḍavāḥ	ca	eva	kim	akurvata	sañjaya	10 - UoH

In each of the modes, we saw various options at the top. Those are redirections to other modes from the current mode. One can view the *Reader* mode and also the filtered list of solutions based on primitive *kāraka* analysis in “SH Selections”. There is also a redirection to the *Saṃsādhani* platform’s dependency parser, from the graphical interface view, using the “UoH Analysis” option and for each of the solutions from the list view.

The main advantage of these modes is the reduction of possibilities. And in these modes, one can also extract the segmentations as a list of strings, in the notation of choice, using the CGI environment variables, essentially functioning as an API for the same. One can also get the segmentaion along with the morphological analysis in a JSON format. More information about how to extract results from SH’s ranking is reported in Appendix C.

However, the disambiguation based on *yogyatā* is not completely solved here. The expected segmentation can be obtained using the *first* mode, if not, then using the *best* mode, and if not, then the *All* mode. The number of solutions displayed in the *All* mode is calculated at the phase level. If we calculate the number of solutions based on

morphological analyses (multi-tags in the phase), it will be even more. Let us see the same example (Bh. 1.1 - fig. 2.1). Table 5.14 shows the possibilities in each chunk when calculated at the word, phase and morphological analysis level. We can see how an unsegmented sentence forms 320 segmentation solutions at the word level, and almost three million solutions considering the morphological analysis.

Table 5.14: Comparison of chunk possibilities (for *Bhagavad Gītā* verse 1.1) across word, phase and morphological analysis

Chunk	Number of possibilities		
	Word	Phase	Morphological Analysis
<i>dharmakṣetre</i>	2	6	17
<i>kurukṣetre</i>	2	6	13
<i>samavetā yuyutsavaḥ</i>	4	8	32
<i>māmakāḥ</i>	5	21	26
<i>pāṇḍavāścaiva</i>	2	2	2
<i>kimakurvata</i>	1	1	2
<i>sañjaya</i>	2	4	4
Total	320	48,384	2,941,592

Let us see how much the two new modes reduce (tab. 5.15). The *best n* mode with word metrics retains almost all segments except those of *māmakāḥ*, and we see 90.47% reduction with respect to phases and 92.31% with respect to morphological analysis. With the morph metrics, we observe 99.60% and 99.74% respectively. In the *first* mode with the word metrics, we see a reduction of 99.86% with respect to phases, 99.74% with respect to morphological analysis. With the morph metrics, 99.99% reduction in morphological analysis is observed and with both phases and words, it returns a unique solution. Having a solution where there is only one morphological analysis possible for each segment is difficult because, choosing the morphological analysis requires the entire sentence's context and sometimes inter-sentential relationships too.

With the given dataset, and the proposed ranking algorithm, we were able to reduce the possibilities as much as possible. In the future direction, one can explore the effect of considering both the word and the morph probabilities together, and also involve context to disambiguate the different morphological analyses of a segment. We also need to explore OOV words and how to handle them, along with efficiently handling compositionality of compounds.

Table 5.15: Comparison of overall possibilities (for *Bhagavad Gītā* verse 1.1) in the *first* and *best n* modes with each the metrics

Mode	Metrics	Number of possibilites		
		Word	Phase	Morphological Analysis
<i>first</i>	word	1	64	7,776
	morph	1	1	108
<i>best 10</i>	word	10	4,608	226,304
	morph	10	192	7,488
All	-	320	48,384	2,941,592

Chapter 6

Conclusion

The present work focused on two tasks. One, alignment of DCS and SH annotations to generate a normalized dataset, and two, ranking the solutions of SH using the normalized dataset. During the first task, several differences between DCS and SH annotations were discovered. The alignment process also helped address the limitations of DCS, validating and normalizing the annotations of both the systems. The ranking algorithm leverages the aligned datasets to produce the most probable solutions at the top. The experiments with various test sets helped discover several anomalies of SH which were corrected further to enhance the system. In this conclusion, the key contributions and inferences from the alignment and ranking experiments are recorded first followed by a discussion on the future aspects of the work.

6.1 Key Contributions

6.1.1 New Alignment of DCS and SH - Normalized dataset

Taking insights from the initial effort towards the alignment of DCS and SH annotations (Krishna et al., 2016), along with its limitations, the present work attempts to create larger normalized datasets. It differs from the previous attempt in three aspects:

1. updating SH's engine to produce all the analysis instead of relying on web-scraping techniques,
2. creation of a dedicated morphological analysis mapper that bridges SH analysis with the DCS analysis, and
3. using linguistic features and auxiliary information like look-up tables for mapping pronoun-stems, compound-stems and derived verbs (secondary conjugations).

It considered the recent changes in both DCS and SH and discussed in detail about the challenges faced during the alignment process. The reasons for mismatches were discussed where lexicon mismatch and compositionality of compounds were found to be the primary causes. The morphological analysis mapper built in this process was additionally extended to map two other existing tagsets (SLP¹ and SCL).

The normalized dataset is created specifically for the tasks of Word Segmentation and Morphological Parsing. While the DCS sentences along with their annotations were directly used, the normalized datasets also come with some of the limitations of DCS. Non-uniform sentence boundaries restrict this dataset from usage in downstream NLP tasks like dependency analysis, although dedicated fields have been provided for the Universal dependency-based POS tags and dependency relations.

However, the available features of the normalized dataset proved sufficient enough to be used for a statistical approach to the segmentation problem in Sanskrit. With more analysis on the remaining sentences of DCS, along with updates to SH's lexicon for the unrecognized stems, named entities and possible compounds with non-compositional analysis, this dataset can be extended further.

6.1.2 Improvement to the Heritage Segmenter

A new ranking algorithm is integrated into the SH system as a dovetailing mechanism on top of its segmentation process. With the ranking metrics being the joint probability of the unigram frequencies of words and morphological analyses obtained from the normalized dataset, the ranking algorithm significantly helped the correct solution to move into the top few. Additionally, this process helped truncate the entire list of possible solutions to a limited set without much loss of recall.

The performance comparison of the different ranking metrics showed that the word metrics was sufficient to push the most probable solution(s) to the top layer, and that the stem-morphological analysis metrics did not produce much improvement for the word segmentation task, indicating that these are best used with the morphological parsing task, viz. to pick out the intended morphological analysis in context from the possible morphological analyses.

¹ <https://sanskritlibrary.org/helpmorphids.html>

The comparison with neural architectures showed on-par performance of the ranking mechanism. A cleaner comparison would have been to train all the models on the same dataset, and then evaluate on a held-out dataset of the same resource. But these could not be achieved due to several technical difficulties including lack of similar hardware requirements. However, the experiments on the Hackathon Dataset, and the true unseen data from *Kathāsaritsāgara* attempts towards a fairer comparison of SH-Ranking with the neural architectures.

Four evaluation strategies were introduced, of which two (sentence and word level using Macro-averaged Precision, Recall and F-Score) were already used previously in the various segmentation models and the other two (compound and ranking) are novel evaluation strategies introduced in the present work.

The alignment processes have proven effective in pushing the correct solutions towards the top ranking. But they have not been as helpful in identifying the segments. Increasing the dataset size, also does not produce a significant improvement to the ranking mechanism. Here, the compositionality of compounds is a major challenge, making the non-determinism in the *Sandhi* resolution within a compound more difficult to handle than the non-determinism in the *Sandhi* resolution between words.

Finally, we can presume that the segmentation models produce near-perfect results for sentences without marking the word boundaries of the compounds. But these are still affected by contextual and discourse level ambiguities (for example, *śveto dhāvati*). Marking the compound boundaries along with its constituency analysis is an important task. For example, the ratio of number of sentences with compounds to those without any compound is very high. *Bhagavad Gītā* has at least one compound in 629 / 700 verses, *Saṅkṣepa-Rāmāyaṇa* 93 / 100 verses and *Meghadūta* 241 / 242 verses. Even the prose uses compounds very often. More importantly, *sandhi* happens both across words and across compound components and the rules of *sandhi* are very much the same for both.

6.1.3 Integrating the updates with *Samśādhanī* tools

The *Samśādhanī* platform² hosts various tools for processing Sanskrit texts of which the *Sandhi-Vicchedikā* (*Sandhi* Splitter) and *Anusāarakam*³ make use of the segmented solutions from SH.

The best possible solutions from SH-Ranking are fed to the *Sandhi-Vicchedikā* and *Anusāarakam* via a pipeline established as a collaborative effort between these two systems to enable cross-platform analysis of sentences (Huet and Kulkarni, 2014). While *Sandhi-Vicchedikā* produces the first solution, and provides a link to redirect to all possible solutions, the *Anusāarakam* takes the segments along with all possible morphological analysis of each of the segments and performs the dependency analysis on the segmented sentence (Kulkarni, 2021).

6.2 Future Work

The intention behind this work was to provide a dataset which is rich in morphological and lexical analyses, and which provides as much details as possible regarding the sentences and words. The alignment process did pave the way to building such a dataset which was used on the existing Segmenter to improve it further. Thus, the resultant dataset could give rise to a homogeneous gold corpus which could in turn be used for various subsequent tasks of sentential analysis. What follows is an account on the future aspects of the work and possible directions for further research.

- **Alignment of the entire DCS dataset:** While we currently have a dataset comprising of almost a third of the DCS, this is less in number when compared to the amount of data available for other languages. One aspect of future scope is to consider the mismatches observed during the alignment and resolving them. This involves updating the SH's lexicon, addressing more linguistic considerations and handling compositionality efficiently.
- **Ranking based on an integrated approach:** An integrated approach using word probabilities to rank the solutions based on word forms and then using the stem

² www.sanskrit.uohyd.ac.in/scl

³ A machine translation and accessor system from Sanskrit to other languages like Hindi, Marathi and Telugu.

and morph probabilities to rank the morph possibilities among each of the segmentations could perform better. This handles non-determinism at two levels - word and morphological analysis. The phase-level non-determinism can also be considered but the dataset should include the SH phase details too.

- **OOV words:** Out-of-vocabulary words are to be handled effectively. Updating the lexicon is one approach. While it is time consuming, it is also prone to explode the lexicon with over-generation of lexicon entries.⁴
- **Compositionality of Compounds:** It is crucial to distinguish the *Sandhi* between the compounds and the *Sandhi* between words as *Sandhi* resolution in a compound requires the compositionality of the compound according to the context in addition to the *Sandhi* rules and lexical and morphological information. Therefore, it is essential to direct further attention towards determining the compositionality of compounds and exploring how it can contribute to word segmentation.

Compositionality of compounds poses problems both in analysis and evaluation. Two levels of evaluation needs to be introduced: syntactic and semantic. While the former ensures that the compound is split with correct components, the latter deals with the constituency analysis taking into account the compositionality based on context. Compounds are to be treated like Multi-word expressions (MWE) as we find various similarities between the two including collocation of components based on semantic relations and strict preference of the components' order. Constant et al. (2017) proposes three categories of compounds based on compositionality, namely, fully compositional, in which case they are not MWEs (e.g., *paper card*), conventionalized, in which case they are statistically idiomatic MWEs (e.g., *credit card*) or non-compositional MWEs (e.g., *green card*). Similar to this, one might also look into which component contributes more to the compound formation, essentially looking at the head word of the compound. This would be one of the measures of annotating the compositionality of a compound.

⁴ For compounds, if the non-compositional analysis is intended and it has to be updated in the lexicon, then adding all such possible compounds that stem out of existing lexicon items would result in over-generation of lexicon entries.

- **Segmentation and Morphological Analysis in Vedic context:** What we have seen majorly pertains to the division called Classical Sanskrit. The other division, Vedic Sanskrit is much older, and contains several peculiarities that impact both segmentation and morphological analysis. Thus, a study on how our existing tools fare on Vedic texts and how these can be developed further to incorporate these peculiarities can help us either towards building an integrated system that solves the NLP problems of both Classical and Vedic Sanskrit, or towards building a system dedicated to Vedic processing.

Bibliography

- Aldous, D. J. (1985). Exchangeability and related topics. In Hennequin, P. L., editor, *École d'Été de Probabilités de Saint-Flour XIII — 1983*, pages 1–198, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Aralikatte, R., Gantayat, N., Panwar, N., Sankaran, A., and Mani, S. (2018). 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. Association for Computational Linguistics. <https://aclanthology.org/D18-1530>.
- Bharati, A., Kulkarni, A., Sheeba, V., and Vidyapeetha, R. (2006). Building a wide coverage Sanskrit morphological analyzer: A practical approach.
- Bhardwaj, S., Gantayat, N., Chaturvedi, N., Garg, R., and Agarwal, S. (2018). SandhiKosh: A benchmark corpus for evaluating Sanskrit sandhi tools. In Calzolari, N., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Hasida, K., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S., and Tokunaga, T., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA). <https://aclanthology.org/L18-1712>.
- Bhatta, S. (1960). *Kathāsaritsāgara (with Hindi Translation)*, volume 3. Vihar Rashtrabhasha Parishad, Oxford.
- Chormai, P., Prasertsom, P., Cheevaprawatdomrong, J., and Rutherford, A. (2020). Syllable-based neural Thai word segmentation. In Scott, D., Bel, N., and Zong, C., editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4619–4637, Barcelona, Spain (Online). International Committee on Computational Linguistics. <https://aclanthology.org/2020.coling-main.407>.

- Constant, M., Eryigit, G., Monti, J., van der Plas, L., Ramisch, C., and Todirascu, A. (2017). Survey: Multiword expression processing: A Survey. *Computational Linguistics*, 43(4):837–892. <https://aclanthology.org/J17-4005>.
- Dave, S., Singh, A. K., A.P., D. P., and Lall, P. B. (2021). Neural Compound-Word (Sandhi) Generation and Splitting in Sanskrit Language. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)*, CODS-COMAD '21, page 171–177, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3430984.3431025>.
- Gao, J., Li, M., Wu, A., and Huang, C.-N. (2005). Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach. *Computational Linguistics*, 31(4):531–574. <https://aclanthology.org/J05-4005>.
- Goldwater, S., Griffiths, T. L., and Johnson, M. (2006). Contextual Dependencies in Unsupervised Word Segmentation. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics. <https://aclanthology.org/P06-1085>.
- Goldwater, S., Johnson, M., and Griffiths, T. (2005). Interpolating between types and tokens by estimating power-law generators. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2005/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf.
- Goyal, P. and Huet, G. (2016). Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Language Modelling*, 4(2):145–182. <https://jlm.ipipan.waw.pl/index.php/JLM/article/view/108>.
- Goyal, P., Huet, G., Kulkarni, A., Scharf, P., and Bunker, R. (2012). A Distributed Platform for Sanskrit Processing. In *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India. The COLING 2012 Organizing Committee. <https://aclanthology.org/C12-1062>.

- Gupta, A. (2012). *Sanskrit Compound Processor*. PhD thesis, Department of Sanskrit Studies, School of Humanities, University of Hyderabad, Hyderabad.
- Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using Networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.
- Haruechaiyasak, C., Kongyoung, S., and Dailey, M. (2008). A comparative study on Thai word segmentation approaches. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 1, pages 125–128.
- Hellwig, O. (2009). SanskritTagger, a stochastic lexical and pos tagger for Sanskrit. *Lecture Notes in Artificial Intelligence*, page 266–277.
- Hellwig, O. (2010). *The Digital Corpus of Sanskrit (DCS)*. <http://www.sanskrit-linguistics.org/dcs/>.
- Hellwig, O. (2015a). Morphological disambiguation of classical sanskrit. In Mahlow, C. and Piotrowski, M., editors, *Systems and Frameworks for Computational Morphology*, pages 41–59, Cham. Springer International Publishing.
- Hellwig, O. (2015b). Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit. <https://api.semanticscholar.org/CorpusID:203632660>.
- Hellwig, O. (2016). Detecting Sentence Boundaries in Sanskrit Texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 288–297, Osaka, Japan. The COLING 2016 Organizing Committee. <https://www.aclweb.org/anthology/C16-1028>.
- Hellwig, O. and Nehrdich, S. (2018). 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, Brussels, Belgium. Association for Computational Linguistics. <https://www.aclweb.org/anthology/D18-1295>.

- Huet, G. (2003). Lexicon-directed Segmentation and Tagging of Sanskrit. In *XIIIth World Sanskrit Conference, Helsinki, Finland. Final version in Themes and Tasks in Old and Middle Indo-Aryan Linguistics*, Eds. Bertil Tikkannen and Heinrich Hettrich., pages 307–325, Delhi. Motilal Banarsidass.
- Huet, G. (2005a). A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Funct. Program.*, 15(4):573–614. <https://doi.org/10.1017/S0956796804005416>.
- Huet, G. (2005b). A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger. *J. Functional Programming*, 15,4:573–614. yquem.inria.fr/~huet/PUBLIC/tagger.pdf.
- Huet, G. (2007). Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In *Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries*, New York, NY, USA. ACM. yquem.inria.fr/~huet/PUBLIC/IWRIDL.pdf.
- Huet, G. (2009). Sanskrit Segmentation. In *Proceedings of the South Asian Languages Analysis Roundtable XXVIII*.
- Huet, G. (2024). Hoisting the colors of Sanskrit. In Bhattacharya, A., editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 39–51, Auroville, Puducherry, India. Association for Computational Linguistics. <https://aclanthology.org/2024.iscls-1.4>.
- Huet, G. and Kulkarni, A. (2014). Sanskrit linguistics web services. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 48–51.
- Huet, G. and Lankri, I. (2018). Preliminary design of a Sanskrit corpus manager. In Huet, G. and Kulakrni, A., editors, *Proceedings of Computational Sanskrit and Digital Humanities, 18th WSC*. <https://open.library.ubc.ca/cIRcle/collections/70440/items/1.0391834>.
- Huet, G. and Razet, B. (2015). Computing with Relational Machines. *Mathematical Structures in Computer Science*, pages 1–20.

- Hyman, M. D. (2008). From Pāṇinian Sandhi to Finite State Calculus. In *Proceedings, 2nd International Sanskrit Computational Linguistics Symposium*, pages 253–265.
- Kālidāsa (1934). The Meghadūta. Gopal Narayen & Co. Book-sellers.
- Kitagawa, Y. and Komachi, M. (2018). Long Short-Term Memory for Japanese Word Segmentation. In *Proceedings of the 32nd Pacific Asia Conference on Language*, pages 279–288, Hong Kong. Information and Computation.
- Krishna, A., Gupta, A., Garasangi, D., Satuluri, P., and Goyal, P. (2020a). Keep it Surprisingly Simple: A Simple First Order Graph Based Parsing Model for Joint Morphosyntactic Parsing in Sanskrit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4791–4797, Online. Association for Computational Linguistics. <https://aclanthology.org/2020.emnlp-main.388>.
- Krishna, A., Santra, B., Bandaru, S. P., Sahu, G., Sharma, V. D., Satuluri, P., and Goyal, P. (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*, pages 2550–2561, Brussels, Belgium. Association for Computational Linguistics. <https://aclanthology.org/D18-1276>.
- Krishna, A., Santra, B., Gupta, A., Satuluri, P., and Goyal, P. (2020b). A Graph-Based Framework for Structured Prediction Tasks in Sanskrit. *Computational Linguistics*, 46(4):785–845. <https://aclanthology.org/2020.cl-4.4>.
- Krishna, A., Santra, B., Satuluri, P. K., Bandaru, S. P., Faldu, B., Singh, Y., and Goyal, P. (2016). 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. The COLING 2016 Organizing Committee. <https://aclanthology.org/C16-1048>.
- Krishna, A., Satuluri, P., and Goyal, P. (2017). A Dataset for Sanskrit Word Segmentation. In Zaimis, E., editor, *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*,

- volume 42, pages 105–114, Vancouver, Canada. Association for Computational Linguistics. <https://aclanthology.org/W17-2214>.
- Krishnan, S. and Kulkarni, A. (2019). Sanskrit Segmentation Revisited. In *Proceedings of the 16th International Conference on Natural Language Processing*, pages 105–114, International Institute of Information Technology, Hyderabad, India. NLP Association of India. <https://aclanthology.org/2019.icon-1.12>.
- Krishnan, S., Kulkarni, A., and Huet, G. (2023). Validation and Normalization of DCS corpus and Development of Sanskrit Heritage Engine’s Segmenter. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*, pages 38–58, Canberra, Australia (Online mode). Association for Computational Linguistics. <https://aclanthology.org/2023.wsc-csdh.3>.
- Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In Blanco, E. and Lu, W., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics. <https://aclanthology.org/D18-2012>.
- Kulkarni, A. (2019). *Sanskrit Parsing based on the theories of Śābdabodha*. IAS, Shimla and D K Printworld.
- Kulkarni, A. (2021). Sanskrit Parsing Following Indian Theories of Verbal Cognition. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(2). <https://doi.org/10.1145/3418061>.
- Kulkarni, A. and Shukl, D. (2009). Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.
- Kumar, A., Kulkarni, A., and Shailaja, N. (2024). START: Sanskrit teaching; annotation; and research tool – bridging tradition and technology in scholarly exploration. In Bhattacharya, A., editor, *Proceedings of the 7th International Sanskrit Computational Linguistics Symposium*, pages 113–124, Auroville, Puducherry, India. Association for Computational Linguistics. <https://aclanthology.org/2024.iscls-1.9>.

- Kumar, A., Mittal, V., and Kulkarni, A. (2010). Sanskrit compound processor. In Jha, G. N., editor, *Sanskrit Computational Linguistics*, pages 57–69, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Lao, N. and Cohen, W. W. (2010). Relational Retrieval Using a Combination of Path-Constrained Random Walks. *Mach. Learn.*, 81(1):53–67. <https://doi.org/10.1007/s10994-010-5205-8>.
- Leskovec, J. and Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.*, 8(1). <https://doi.org/10.1145/2898361>.
- Li, J. and Gırrbach, L. (2022). Word Segmentation and Morphological Parsing for Sanskrit. *CoRR*, abs/2201.12833. <https://arxiv.org/abs/2201.12833>.
- Li, X., Meng, Y., Sun, X., Han, Q., Yuan, A., and Li, J. (2019). Is Word Segmentation Necessary for Deep Learning of Chinese Representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252, Florence, Italy. Association for Computational Linguistics. <https://aclanthology.org/P19-1314>.
- Ma, J., Ganchev, K., and Weiss, D. (2018). State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics. <https://aclanthology.org/D18-1529>.
- Mittal, V. (2010). Automatic Sanskrit Segmentizer Using Finite State Transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90, Uppsala, Sweden. Association for Computational Linguistics. <https://aclanthology.org/P10-3015>.
- Monier-Williams, S. M. (1899). *A Sanskrit-English Dictionary*. Oxford University Press, Oxford.
- Natarajan, A. and Charniak, E. (2011). s^3 - Statistical Sandhi Splitting. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 301–308,

- Chiang Mai, Thailand. Asian Federation of Natural Language Processing. <https://aclanthology.org/I11-1034>.
- Nguyen, D., Thin, D. V., Nguyen, K. V., and Nguyen, N. L. (2020). Vietnamese Word Segmentation with SVM: Ambiguity Reduction and Suffix Capture. *CoRR*, abs/2006.07804. <https://arxiv.org/abs/2006.07804>.
- Pillai, P. K. N. (1941). THE ṚGVEDA PADAPĀṬHA—A STUDY WITH SPECIAL REFERENCE TO THE ṚGVEDA PRĀTISĀKHYA. *Bulletin of the Deccan College Research Institute*, 2(3/4):247–257. <http://www.jstor.org/stable/42931274>.
- Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401.
- Prince, A. and Smolensky, P. (1993). John Wiley & Sons, Ltd, Center for Cognitive Science, Rutgers University, Piscataway.
- Reddy, V., Krishna, A., Sharma, V., Gupta, P., M R, V., and Goyal, P. (2018). Building a Word Segmenter for Sanskrit Overnight. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA). <https://aclanthology.org/L18-1264>.
- Riley, M., Allauzen, C., and Jansche, M. (2009). OpenFst: An open-source, weighted finite-state transducer library and its applications to speech and language. In Chelba, C., Kantor, P., and Roark, B., editors, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 9–10, Boulder, Colorado. Association for Computational Linguistics. <https://aclanthology.org/N09-4005>.
- Sandhan, J., Behera, L., and Goyal, P. (2023). Systematic investigation of strategies tailored for low-resource settings for low-resource dependency parsing. In Vlachos, A. and Augenstein, I., editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2164–2171, Dubrovnik, Croatia. Association for Computational Linguistics. <https://aclanthology.org/2023.eacl-main.158>.

- Sandhan, J., Singha, R., Rao, N., Samanta, S., Behera, L., and Goyal, P. (2022). TransLIST: A Transformer-Based Linguistically Informed Sanskrit Tokenizer. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6902–6912, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. <https://aclanthology.org/2022.findings-emnlp.513>.
- Satuluri, P. (2015). *Sanskrit Compound Generation: With a Focus on the Order of Operations*. PhD thesis, Department of Sanskrit Studies, School of Humanities, University of Hyderabad, Hyderabad.
- Scharf, P. and Hyman, M. (2009). *Linguistic Issues in Encoding Sanskrit*. Motilal Banarsidass, Delhi.
- Shao, Y., Hardmeier, C., and Nivre, J. (2018). Universal Word Segmentation: Implementation and Interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435. <https://aclanthology.org/Q18-1030>.
- Thang, D. Q., Phuong, L. H., Huyen, N. T. M., Tu, N. C., Rossignol, M., and Luong, V. X. (2008). Word Segmentation of Vietnamese Texts: A Comparison of Approaches. In *LREC’08*, Marrakech, Morocco.
- Vālmīki (2002). *Saṅkṣepa Rāmāyaṇam*. Rashtriya Sanskrit Samsthan.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- Vyāsa (2007). *Śrīmad Bhagvad Gītā (Padaccheda-Anvaya)*. Geeta Press.
- Xue, N. (2003). Chinese word segmentation as character tagging. In *Computational Linguistics and Chinese Language Processing*, volume 8(1), pages 29–48.
- Zhao, H., Huang, C.-N., and Li, M. (2006). An improved Chinese word segmentation system with conditional random field. In Ng, H. T. and Kwong, O. O.,

editors, *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165, Sydney, Australia. Association for Computational Linguistics. <https://aclanthology.org/W06-0127>.

Appendix A

DCS-SH Alignment

The alignment of the Digital Corpus of Sanskrit (DCS) and Sanskrit Heritage Segmenter (SH) brought out the core differences between the two systems which were briefly discussed in Chapter 3 along with possible solutions to handle them. A detailed discussion with various scenarios is presented in this section.

DCS has around 644,000¹ annotated sentences where about 22,000 were found with corrupted or incomplete data. 622,000 sentences were valid and these have 4.5 million word references in total. And there are 90,544 unique stems (including homonymous stems). If we ignore the differences due to homonymy indices, there are 82,879 unique stems in total.

Of the word references, 802,097 segments do not have any morphological analysis at all. It was observed that there are 5,813 such unique entries and some of these are “indeclinables”. From a list of all indeclinables, taken from *Samśādhani* tools², a comparison was done to check these segments, and is presented in table A.1.

The differences between DCS and SH according to their morphological analyses is recorded ahead.

Morphological Analysis

A general overview of all the parameters in DCS’ morphological analysis is presented in tables A.2 (noun), A.3, A.4 (verb), and A.5 (primary derivative). With these parameters, DCS has various combinations to produce many of the available morphological analyses, with a few exceptions. An analysis of these morphological analyses is being done here along with a comparison with the morphological analyses of SH and *Samśādhani* tools.

¹ This is the number of sentences at the time of this writing. As DCS is developed regularly, these statistics may change subsequently.

² <https://sanskrit.uohyd.ac.in/scl/>

Table A.1: DCS Word references without morphological analysis

Type	No. of References
Total	802,097
Indeclinables	724,948
Initial compound components	28,003
Nominal stems	32,145
Words	14,173
Words with <i>tasil</i> -suffix ¹	2,787
Others	41

¹ *tasil* is a secondary derivative (*taddhita*) suffix, used to generate either an adverbial form or an indeclinable word. For example, *viśeṣa* → *viśeṣataḥ* - (ind.) especially, (adv.) in particular

Table A.2: DCS Morphological Parameters - Noun

Case	Reference	Gender	Reference	Number	Reference
Cpd	Compound (iic.)	Masc	Masculine	1	Singular
Nom	Nominative	Fem	Feminine	2	Dual
Acc	Accusative	Neut	Neuter	3	Plural
Ins	Instrumental				
Abl	Ablative				
Dat	Dative				
Gen	Genitive				
Loc	Locative				
Voc	Vocative				

Primary Nouns

For the noun and pronominal forms, a direct one-one map was obtained between the morphological analyses of DCS and SH. DCS records the inflectional morphological

Table A.3: DCS Morphological Parameters - Verbs (Part 1)

Tense	Reference	Mood	Reference	Formation	Reference
Pres	Present	Ind	Indicative	peri	Periphrastic
Perf	Perfect	Imp	Imperative	root	Aorist-root
Impf	Imperfect	Opt	Optative	them	Aorist-thematic
Fut	Future	Sub	Subjunctive	red	Aorist-reduplicated
Aor	Aorist	Inj	Injunctive	s	Aorist-s
Plp	Pluperfect	Cond	Conditional	is	Aorist-is
		Prec	Precative / Benedictive	sis	Aorist-sis
				sa	Aorist-sa

Table A.4: DCS Morphological Parameters - Verbs (Part 2)

Person	Reference	Number	Reference	Voice	Reference
1	First	Sing	Singular	Pass	Passive
2	Second	Dual	Dual		
3	Third	Plur	Plural		

Table A.5: DCS Morphological Parameters - Derivatives

VerbForm	Reference
PPA	Past Active Participle
PPP	Past Passive Participle
Ger	Future Passive Participle
Part	Present Participle (Active, Middle, Passive)
Inf	Infinitives
Abs	Absolutives

analysis for primary nouns as: “Case=<>|Number=<>|Gender=<>”. The values for these three parameters are according to the entries in table A.2. All the three parameters are recorded for nouns while the gender parameter is left empty for the pronominals. For non-final components of the compound words, DCS uses the value “Cpd”.

SH has similar analyses. The eight case values corresponding to the values of DCS are: [“nom.”, “acc.”, “i.”, “dat.”, “abl.”, “g.”, “loc.”, “voc.”]. The Number values are: [“sg.”, “du.”, “pl.”] and the Gender values are [“m.”, “f.”, “n.”, “*”] (* for pronominals and numerals). The non-final components of compounds are assigned the analysis *iic*.

Derivatives

The primary derivatives (*kr̥dantas*) of DCS have the base derivational analysis concatenated to the inflectional analysis (table A.5). SH provides the derivational analysis separately (table A.6). A sample of the annotations of primary derivations in DCS and SH is compared in the table A.7.

To summarize, we can observe the following when we compare the morph representations of the primary derivatives of DCS and SH:

1. For Present and Future participles, DCS marks two parameters: Tense and VerbForm, and additionally a third parameter, Voice, for Passive voice. SH, on the other hand, provides the information on *Pada* (*parasmai* - Active or *ātmane* - Middle), and Conjugation (Primary, Causative, Desiderative or Intensive). For the Present Participles, it also gives the *Gaṇa* or class information. For Future Passive Participles, it provides information regarding the specific suffix which is added (*yat* - 1, *anīyar* - 2, *tavya* - 3).
2. For Past participles, DCS has only one parameter VerbForm (PPP or PPA). SH additionally gives the Conjugation information (“ca.”, “des.”, “int.”) whenever applicable.
3. For Absolutives and Infinitives, DCS has one Parameter (VerbForm=Abs or Inf), and SH produces inf. or abs. The conjugation information is additionally present in SH’s analysis. And SH treats the Absolutives and Infinitives as indeclinables.
4. The primary derivative analyses excluding Absolutives and Infinitives amounts to 66. This includes past-active, past-passive, perfect-active, perfect-middle, future-active, future-middle, future-passive (3), present-active, present-middle

Table A.6: SH Morphological Parameters - Derivatives

Derivational Analysis	Reference
ppr. [x] ¹ ac.	Present Active Participle
ppr. [x] md.	Present Middle Participle
ppr. ps.	Present Passive Participle
ppa.	Past Active Participle
pp.	Past Passive Participle
ppf. ac.	Perfect Active Participle
ppf. md.	Perfect Middle Participle
pfu. ac.	Future Active Participle
pfu. md.	Future Middle Participle
pfp. [1]	Future Passive Participle (<i>yat</i>)
pfp. [2]	Future Passive Participle (<i>anīyar</i>)
pfp. [3]	Future Passive Participle (<i>tavya</i>)

¹ *x* refers to one of the 10 classes *gaṇas*

and present-passive participles. The present active and middle participles are additionally annotated with the *gaṇa* of the original verb which creates 20 distinct analyses entries for their primary conjugation forms. The secondary conjugation possibilities (3 - causative, desiderative, intensive) are also annotated for all the participles. These combine with all the previously mentioned noun forms (97 - one compound form along with 24 forms each of Masculine, Feminine, Neuter and Pronominal). This amounts to a maximum of 6,402 possible *kṛdanta* (primary derivative) analyses from SH.

5. But for Present, Past-Passive and Future Participles, DCS also specifies them separately without any inflectional analysis. This causes mismatch with SH as SH always has the inflectional analysis.

Table A.7: DCS - SH Primary Derivatives comparison				
Example	DCS		SH	Reference
	Tense	VerbForm		
<i>bhuktavat</i>	-	PPA	ppa.	Past Active Participle (ktavatu)
<i>upadiṣṭāḥ</i>	-	PPP	ca. pp.	Causative Past Passive Participle (ṇijanta kta)
<i>gacchan</i>	Pres	Part	ppr. [1] ac.	Present Active Participle (śatṛ)
<i>locamānaḥ</i>	Pres	Part	ppr. [1] md.	Present Middle Participle (śānac)
<i>vakṣyamāna</i>	Fut	Part	pfu. md.	Future Middle Participle (luḍādeśa)
<i>vijñeyam</i>	-	Ger	pfp. [1]	Future Passive Participle (yat)
<i>sāadhanīyam</i>	-	Ger	ca. pfp. [2]	Causative Future Passive Participle (ṇijanta anīyar)
<i>kartavyam</i>	-	Ger	pfp. [3]	Future Passive Participle (tavya)
<i>śodhayitvā</i>	-	Abs	ca. abs.	Absolutive (ktvā, lyap, ṇamul)
<i>śrotum</i>	-	Inf	inf.	Infinitive (tumun)

6. DCS provides the VerbForm as PPA, PPP, Ger and Part with Voice as Passive for Participles. In association with the inflectional analysis (noun), we get a maximum of 485 possible analyses, of which only 335 are used in the existing dataset. Of all the morphological analysis extracted from DCS, we have a unique list of 683 morph analyses. Of these, such *kṛdantas* are found in 335 analysis. So, out of the 6, 402 possible forms from SH, we have 335 in the DCS.

Compounds

The non-final components of a compound word are analysed as “Case=Cpd” by DCS and “iic.” by SH. The non-final compound components, which are primary derivatives, are additionally provided with the derivational morph analysis with the parameter VerbForm, in DCS. And SH provides the derivational analysis separately like pp., ppr. pfp., etc. Sometimes the derivational analysis is not provided by the DCS and

multiple mappings are hence possible. For certain roots, their primary and causative conjugation word forms are identical. In such cases, SH produces both the analysis of primary as well as the conjugated form, while DCS does not produce such additional information, leading to a one to many mapping from DCS to SH. This can be observed in the sentence *nimīlitākṣo mūrdhaste śiro roge na dhārayet* and table A.8 for the word *nimīlita*.

Table A.8: DCS-SH Mapping Example 3

Word	Stem	Case	Number	Gender	VerbForm	SH Morph
<i>nimīlita</i>	<i>nimīlay</i>	Cpd	-	-	PPP	{pp.} <i>ni-mīl</i> {iic.} {ca. pp.} <i>ni-mīl</i> {iic.}
<i>akṣaḥ</i>	<i>akṣa</i>	Nom	1	Masc	-	{m. sg. nom.}
<i>mūrdha</i>	<i>mūrdhan</i>	-	-	-	-	{iic.}
<i>sthe</i>	<i>stha</i>	Loc	1	Masc	-	{m. sg. loc.}
<i>śiro</i>	<i>śiras</i>	Gen	3	Masc	-	{m. pl. g.}
<i>roge</i>	<i>roga</i>	Loc	1	Masc	-	{m. sg. loc.}
<i>na</i>	<i>na</i>	-	-	-	-	{ind.}
Word	Stem	Tense	Mood	Person	Number	SH Morph
<i>dhārayet</i>	<i>dhāray</i>	Pres	Opt	3	Sing	<i>dhr</i> {ca. opt. ac. sg. 3}

Verbs

Verbs in SH: The following are the important observations on the morphological analysis of verbs in SH:

1. A verb has four parameters: <conjugation, paradigm, number, person>.
2. **conjugation** has four possibilities:
 - (a) Primary → “”
 - (b) Causative → “ca.”
 - (c) Desiderative → “des.”
 - (d) Intensive → “int.”
3. **paradigm** has been handled in three ways:

-
- (a) Present Tense
 - (b) Conjugated
 - (c) Periphrastic Future
4. For the Present tense, there are four modes:
- (a) Present → “pr.”
 - (b) Imperative → “imp.”
 - (c) Optative → “opt.”
 - (d) Imperfect → “impft.”
5. SH refers to the remaining tenses as Conjugated forms where the following are considered as the tenses:
- (a) Future → “fut.”
 - (b) Perfect → “perf.”
 - (c) Aorist → “aor.”
 - (d) Injunctive → “inj.”
 - (e) Conditional → “cond.”
 - (f) Benedictive → “ben.”
 - (g) Subjunctive → “subj.”

DCS - Verbs: DCS uses 4 primary parameters (with 2 additional parameters(*) for special cases) for annotating a verb:

1. Tense → Present, Perfect, Imperfect, Future, Aorist, Pluperfect
2. Mood → Indicative, Imperative, Optative, Subjunctive, Injunctive, Conditional, Precative
3. *Formation → Periphrastic, root-aorist, reduplicated-aorist, is-aorist, thematic-aorist, sa-aorist, s-aorist, sis-aorist
4. *Voice → Passive
5. Person → 1, 2, 3
6. Number → Singular, Dual, Plural

A combination of the Tense and Mood parameters is mapped to the 10 *lakāras* of Sanskrit. And they can also be mapped with SH terms, with a few exceptions. Table A.9 is used as a reference to map the two systems. It contains the 10 *lakāras* with their sub-types as well.

Table A.9: Reference for the Tense-Mood combinations

<i>lakāra</i>	Reference
<i>laṭ</i> (<i>vartamāna</i>)	present-indicative
<i>loṭ</i> (<i>ājñārtha</i>)	present-imperative
<i>laṇ</i> (<i>anadyatana bhūta</i>)	imperfect-indicative pluperfect-indicative
<i>luṇ</i> (<i>adyatana bhūta</i>)	aorist-indicative
<i>vidhiliṇ</i>	present-optative
<i>āśīrliṇ</i>	aorist-optative benedictive
<i>liṭ</i> (<i>parokṣe</i>)	perfect-indicative
<i>liṭ</i> (<i>anuprayoga</i>)	periphrastic-perfect
<i>luṭ</i> (<i>anadyatana bhaviṣyat</i>)	periphrastic-future
<i>lṛṭ</i> (<i>adyatana bhaviṣyat</i>)	future-indicative
<i>lṛṇ</i> (<i>bhaviṣyat</i>)	future-conditional
<i>leṭ</i>	present-subjunctive perfect-subjunctive aorist-subjunctive
<i>laṇ</i> (<i>a-abhāva</i>)	present-injunctive
<i>liṭ</i> (<i>a-abhāva</i>)	perfect-injunctive
<i>lṛṇ</i> (<i>a-abhāva</i>)	future-injunctive
<i>luṇ</i> (<i>a-abhāva</i>)	aorist-injunctive

In the recent developments to the DCS dataset, the Injunctive (Inj) mood is termed as Jussive (Jus) and the Perfect and Aorist Tenses have been clubbed together as Past. This resulted into a larger number of many to one mapping from SH's analyses to

DCS analysis. Hence, the recent developments have been neglected temporarily and the previous version of the DCS is taken into consideration for alignment.

Observations on the morphological analyses of DCS are as follows:

- DCS has Present and Imperfect as Tenses and Imperative and Optative as moods which combine with Present, Future and Aorist Tenses.
- DCS has Future, Perfect and Aorist as Tenses, and Injunctive, Conditional, Precative (Benedictive) and Subjunctive as moods with the following combinations:
 1. Injunctive → Present and Aorist
 2. Subjunctive → Present and Aorist
 3. Conditional → Future
 4. Precative → Aorist
- There are two analyses for precatives: Aorist-Optative and Aorist-Precative in DCS. Temporarily, it is assumed (after checking a few examples) that benedictives are precatives and aorist-optatives are incorrect morph analysis which are to be verified manually.

Table A.10 shows the resulting comparison between the tense and mood combinations and SH's morph analyses. Table A.11 shows the analyses which require further verification as these are unrecognized by SH.

A combination of all of these parameters with "Voice", "Person", and "Number" are generated to form the overall mapper for morphological analysis of verbs. A few observations while generating this mapper are recorded here.

1. SH has three additional information in "*Pada*", "Secondary Conjugation", and "*Gaṇa*" while DCS does not have these.
2. Present Tense:
 - (a) The *Gaṇa* information is present for the Present Tense with multiple moods (ind, imp, opt, inj, subj).
 - (b) SH fails to analyse most of the injunctives and subjunctives.
 - (c) Mapping between DCS and SH was obtained directly for all the other cases in Present tense.
3. Imperfect:

Table A.10: DCS Tense-Mood combinations - comparison with SH

Example	Tense	Mood	Formation	SH	Reference
<i>abhūt</i>	Aor	Ind	is	{aor. [1] ac. sg. 3}	root-aorist-indicative
<i>avocat</i>	Aor	Ind	them	{aor. [2] ac. sg. 3}	a-aorist-indicative
<i>ajigrahat</i>	Aor	Ind	red	{aor. [3] ac. sg. 3}	reduplicated-aorist-indicative
<i>adrākṣam</i>	Aor	Ind	s	{aor. [4] ac. sg. 1}	s-aorist-indicative
<i>avadhīt</i>	Aor	Ind	is	{aor. [5] ac. sg. 3}	iṣ-aorist-indicative
<i>ajñāsīt</i>	Aor	Ind	sis	{aor. [6] ac. sg. 3}	siṣ-aorist-indicative
<i>adhukṣan</i>	Aor	Ind	sa	{aor. [7] ac. pl. 3}	sa-aorist-indicative
<i>bhaviṣyathaḥ</i>	Fut	Ind		{fut. ac. du. 2}	future-indicative
<i>gantā</i>	Fut	Ind	peri	{per. fut. ac. sg. 3}	periphrastic-future
<i>asthāsyat</i>	Fut	Cond		{cond. ac. sg. 3}	conditional
<i>uvāca</i>	Perf	Ind		{pft. ac. sg. 3}	perfect-indicative
<i>āhvayāmāsa</i>	Perf	Ind	peri	{per. perf. ac. sg. 3}	periphrastic-perfect
<i>abhāṣata</i>	Impft	Ind		{impft. [1] md. sg. 3}	imperfect-indicative ¹
<i>vada</i>	Pres	Imp		{imp. [1] ac. sg. 2}	present-imperative ¹
<i>praṇamāmi</i>	Pres	Ind		{pr. [1] ac. sg. 1}	present-indicative ¹
<i>bodhayet</i>	Pres	Opt		{ca. opt. ac. sg. 3}	present-optative ¹
<i>bhūyāt</i>	Aor	Prec		ben. ac. sg. 3	aorist-precative aorist-benedictive

¹ for all *gaṇa*

(a) There is only one mood associated with imperfect tense - ind.

(b) The *Gaṇa* information is available here too in SH.

4. Future:

(a) There are three types of the future tense: *lṛt* (future-indicative), *luṭ* (periphrastic-future), and *lṛñ* (future-conditional) and for all the three, there is a one-to-one mapping between DCS and SH.

Table A.11: DCS Tense-Mood combinations - unrecognized by SH or missing in DCS

Example	Tense	Mood	Formation	SH	Reference
<i>ūhiṣeyāḥ</i>	Aor	Opt	-	?	aorist-optative
<i>kṛdhi</i>	Aor	Imp	-	?	aorist-imperative
<i>kṛthā</i>	Aor	Inj	-	?	aorist-injunctive
<i>karat</i>	Aor	Sub	-	?	aorist-subjunctive
<i>vāvr̥dhasva</i>	Perf	Imp	-	?	perfect-imperative
<i>ninīyāt</i>	Perf	Opt	-	?	perfect-optative
<i>sus̥ptā</i>	Perf	Inj	-	?	perfect-injunctive
<i>babhūyāt</i>	Perf	Sub	-	?	perfect-subjunctive
<i>yajāt</i>	Pres	Sub	-	subj.	present-subjunctive ¹
<i>āvavṛtran</i>	Plp	Ind	-	?	pluperfect-indicative
<i>janat</i>	Pres	Inj	-	inj.	present-injunctive ¹

¹ for all *gaṇa*

5. Perfect Tense:

- (a) Perfect Indicative and Periphrastic Perfect are directly mapped.
- (b) For Perfect Imperative, Optative, Injunctive and Subjunctive, almost all the examples are not analysed by SH.

6. Pluperfect:

- (a) These are not implemented by SH. So all the sentences which have words in plp form are ignored during the alignment.

7. Aorist:

- (a) The seven types of Aorist formation and their indicators (numbers) in SH are:
 - i. root → 1 → *sic-luk*
 - ii. a (thematic) → 2 → *añ*
 - iii. red (reduplicated) → 3 → *cañ*
 - iv. s → 4 → *sic*
 - v. is → 5 → *seṭ-sic*

vi. sis $\rightarrow$ 6 $\rightarrow$ seṭ-sic (*ās-anta-dhātu*)

vii. sa $\rightarrow$ 7 $\rightarrow$ ksa

- (b) Aorist Optative is analysed separately, in addition to Aorist Precative in DCS. And SH analyses both of these as “ben.” (Benedective - *āśīrlin*), causing a two-to-one mapping from DCS to SH. Hence, temporarily Aor-Opt is not considered for the alignment and these are considered for manual verification.

The injunctive and subjunctive cases of Aorist tense are not uniformly analysed by SH and hence a possible mapper for the same could not be recorded.

The following are the morph analyses which couldn't be mapped or analysed as sufficient examples could not be validated with the SH analysis and hence require manual verification in the examples of DCS:

- Perfect-Imperative
- Perfect-Optative
- Perfect-Injunctive
- Perfect-Subjunctive
- Aorist-Imperative
- Aorist-Optative
- Aorist-Injunctive
- Aorist-Subjunctive

Comparison with other formats

A comparison of the morphological analyses from SH, *Samśādhani* tools (SCL), and Sanskrit Library format (SLP) is provided by The Sanskrit Library³. This has been updated further here according to the recent changes in SCL and SH, and the DCS' morphological analysis is also integrated with this. A sample for the DCS-SH comparison is presented in Tables A.12 and A.13. A reference to map SCL and SLP is presented in table A.14.

³ <https://sanskritlibrary.org/helpmorphids.html>

Table A.12: DCS-SH Morph Comparison - Nouns

Example	Case	Number	Gender	SH	Reference
<i>bandhuḥ</i>	Nom	1	Masc	m. sg. nom.	masculine nominative singular
<i>nāsikābhyām</i>	Abl	2	Fem	f. du. abl.	feminine ablative dual
<i>kāryeṣu</i>	Loc	3	Neut	n. pl. loc.	neuter locative plural

Table A.13: DCS-SH Morph Comparison - Verbs

Example	Tense	Mood	Person	Number	Voice / Formation	SH	Reference
<i>praṇamāmi</i>	Pres	Ind	1	Sing	-	pr. [1] ac. sg. 1	*present-indicative class1 active first-person singular
<i>prapadye</i>	Pres	Ind	1	Sing	Pass	pr. ps. sg. 1	*present-indicative passive first-person singular
<i>āstam</i>	Impf	Ind	2	Dual	-	impft. [2] ac. du. 2	*imperfect-indicative class2 active second-person dual
<i>kariṣyatha</i>	Fut	Ind	2	Plur	-	fut. ac. pl. 2	*future-indicative active second-person plural
<i>gantā</i>	Fut	Ind	3	Sing	peri	per. fut. ac. sg. 3	periphrastic future-indicative active third-person singular
<i>dadhiṣe</i>	Perf	Ind	2	Sing	-	pft. md. sg. 2	*perfect-indicative middle first-person dual
<i>abhiṣicyasva</i>	Pres	Imp	1	Plur	Pass	imp. ps. sg. 2	*present-imperative passive first-person plural
<i>abhaviṣyaḥ</i>	Fut	Cond	2	Sing	-	cond. ac. sg. 2	*(future) conditional second-person singular
<i>adhāḥ</i>	Aor	Ind	2	Sing	root	aor. [1] ac. sg. 2	*aorist-indicative root-aorist active second-person singular
<i>syātām</i>	Pres	Opt	3	Dual	-	opt. [2] ac. du. 3	*present-optative class2 active third-person dual

Table A.14: SCL-SLP Morph Comparison

SCL	SLP	Reference
<i>pum;1;eka</i>	m1s	masculine nominative singular
<i>stri;5;dvi</i>	f5d	feminine ablative dual
<i>napum;7;bahu</i>	n7p	neuter locative plural
<i>kartari;laṭ;u;eka; parasmaipadī;bhvādi;</i>	pre[1] a1s	*present-indicative class1 active first-person singular
<i>karmaṇi;laṭ;u;eka; ātmanepadī;divādi;</i>	pre p1s	*present-indicative passive first-person singular
<i>kartari;laṇ;ma;dvi; parasmaipadī;adādi;</i>	ipf[2] a2d	*imperfect-indicative class2 active second-person dual
<i>kartari;lrṭ;ma;bahu; parasmaipadī;;</i>	fut a2p	*future-indicative active second-person plural
<i>kartari;luṭ;pra;eka; parasmaipadī;;</i>	pft a3s	periphrastic future-indicative active third-person singular
<i>kartari;liṭ;u;xvi; ātmanepadī;;</i>	prf m1d	*perfect-indicative middle first-person dual
<i>karmaṇi;loṭ;u;bahu; ātmanepadī;;</i>	ipv p1p	*present-imperative passive first-person plural
<i>kartari;lrñ;ma;eka; parasmaipadī;;</i>	con a2s	*conditional active second-person singular
<i>kartari;luñ;ma;eka; parasmaipadī;;</i>	aor[root] a2s	*aorist-indicative root-aorist active second-person singular
<i>kartari;vidhiliṇ;pra;dvi; parasmaipadī;adādi;</i>	pop[2] a3d	*present-optative class3 middle third-person dual

Appendix B

Resources

The list of resources associated with the alignment of the Digital Corpus of Sanskrit with the Sanskrit Heritage Segmenter, the normalized dataset generated from the alignment and the developments to the Sanskrit Heritage Segmenter with the help of a ranking mechanism are enlisted and explained here.

Extracting DCS and SH analyses

The DCS data is extracted from:

<https://github.com/OliverHellwig/sanskrit>

The DCS analysis were converted and normalized into a standard format considering terminal sandhi, homonasal conversions etc. The scripts for the same are available here:

https://github.com/SriramKrishnan8/dcs_interface

The list of all DCS sentences, those recognized by SH, and their corresponding SH analyses are recorded here:

https://github.com/SriramKrishnan8/sh_analyses

DCS-SH Alignment

The codebase for the implementation of the alignment is available here:

https://github.com/SriramKrishnan8/dcs_sh_alignment

This provides the details of all the steps involved along with links to the pre-requisites, namely DCS analysis in the JSON format and SH analysis in the GraphML format for the first alignment and tsv format for the second and third alignments, and the corresponding normalized datasets obtained from the three alignments, along with the Hackathon dataset.¹. Additionally, the following conversion tables used during the alignment process for stems and morphological analyses are also released:

1. DCS' CNG value to SH's morphological analysis (Alignment 1),
2. DCS-SH morphological analysis mapper (Alignment 2 and 3),
3. DCS-SH-SCL morphological analysis mapper,
4. DCS-SH causative to root mapping, and
5. DCS-SH pronoun mapping.

The scripts to extract the statistics from the alignment datasets are available here:

https://github.com/SriramKrishnan8/alignment_dataset

Segmentation Evaluation

The evaluation scripts and the results are available here:

https://github.com/SriramKrishnan8/sanskrit_segmentation_evaluation.

Both sentence-level and word-level evaluation methods have been described and recorded here. The results contain the following:

1. macro-averaged precision, recall, f-score,
2. perfect match,
3. number of compound matches.
4. mean rank,

¹ All the necessary resources are compiled and saved here: <https://drive.google.com/drive/u/2/folders/1VmffgzbcTyg9cJ6o4eT0FYInfG07Jl-H>

5. mean reciprocal rank, and

6. position distribution

The results are obtained with and without considering compound boundary markings. Both the gold data and the predictions according to the various metrics of SH-Ranking and rcNN are recorded.

Appendix C

Word Segmentation and Morphological Analysis using SH

The results of the ranking-based segmentation with morphological analysis of each of the segments can be accessed from the Sanskrit Heritage Segmenter, either through an API directly or through a python package that handles various scenarios like transliteration, encapsulating results as structured JSON, error handling etc. The details of the API and the package are described ahead.

APIs for SH Results

SH segmenter can be run in six modes, viz. First, Best, All, Tagging, Parsing and Analysis. The ‘First’ and ‘Best’ correspond to the ranking mechanism and APIs have been made available for these two.¹ ‘All’ mode presents all possible segments in a graphical interface with the choice of selection left to the users. The URL for the ‘All’ mode is:

<https://sanskrit.inria.fr/cgi-bin/SKT/sktgraph.cgi>

Its corresponding environment variables are enlisted in table C.1.

SH is a lexicon-directed segmenter containing two lexicons: Heritage and Monier-Williams, and either option can be chosen. The input can either be a sentence or a word. The ‘Input Level’ parameter (‘st’) toggles the engine between functioning as a segmenter (*t*) for a sandhied sentence and as a morphological analyser (*f*) for a word. For a sandhied sentence as the input, the ‘Input Type’ parameter (‘us’) should be *t* and it does segmentation and morphological analysis together. On the other hand, a segmented sentence can also be provided as the input, where ‘us’ should be *f*, and

¹ The manual for the Sanskrit Heritage Platform is available here: <https://sanskrit.inria.fr/manual.html>

Table C.1: Parameters for the Graphical Summary Mode (All)

Parameter	Env Var	Options	Details
Lexicon	lex	SH MW	Sanskrit-French Heritage Sanskrit-English Monier-Williams
Input Level	st	t f	Sentence Word
Input Type	us	t f	Segmented Input Sandhied Input
Input font	t	WX SL KH DN RN VH	WX SLP1 (Sanskrit Library Platform) Kyoto-Harvard Devanagari IAST Velthuis
Output font	font	deva roma	Devanagari Output Roman (IAST) Output
Segmentation Mode	mode	g	Graphical Summary (All)

it performs the morphological analysis of each of the segments. For segments which are compounds, it does both segmentation and morphological analysis. The input and output fonts are mandatory.

For the first and best modes, all the parameters in table C.1 except ‘mode’ are applicable. Table C.2 records the additional parameters exclusively introduced for the ranking-based segmentation. Its corresponding URL is:

<https://sanskrit.inria.fr/cgi-bin/SKT/sktgraph2.cgi>.

The ‘pipeline’ parameter was introduced to establish a connection between the *Samśādhaniī* tools² and the SH Segmenter. When ‘pipeline’ is *t*, with the segmentation modes as *s*, then we get the segmentation as a JSON object. For example, the following is produced for the word *rāmālayaḥ*:

```
{
  "input": "rAmAlayaH",
  "segmentation": [
```

² <https://sanskrit.uohyd.ac.in/scl/>

Table C.2: Additional Parameters for the Ranking modes (First and Best)

Parameter	Env Var	Options	Details
Segmentation Mode	mode	f	All possible segments from solution 1
		b	All possible segments from the solutions 1-10
		s	First segmentation (string)
		l	List of top 10 segmentations
Frequency Mode	fmode	w	word
		t	transition
		s	stem
		m	morphological analysis
		x	word-transition bigram
		n	stem-morphological analysis bigram
Pipeline	pipeline	t	Segmented Input
		f	Sandhied Input
Stemmer	stemmer	t	Sentence
		f	Word

```

    "rAma-AlayaH"
  ]
}

```

With the segmentation mode as *l*, the following is produced:

```

{
  "input": "rAmAlayaH",
  "segmentation": [
    "rAma-AlayaH",
    "rAma-alayaH",
    "rAma AlayaH",
    "rAma alayaH",
    "rAmA layaH",
    "rAmA AlayaH",
    "rAmA alayaH"
  ]
}

```

With the segmentation mode as *f*, the first solution *rAma-AlayaH* and all the possible morphological analyses of the corresponding segments are produced in an XML

format. Similarly, with the segmentation mode as *b*, all the possible morphological analyses of the segments corresponding to the top 10 solutions are produced in the same XML format.

The ‘stemmer’ mode is introduced to produce both the segmentation and morphological analysis in JSON format. With the ‘stemmer’ as *t* and segmentation mode as *s* or *f*, the following is produced:

```
{
  "input": "rAmAlayaH",
  "segmentation": [
    "rAma-AlayaH"
  ],
  "morph": [
    {
      "word": "rAma-",
      "derived_stem": "rAma",
      "base": "",
      "derivational_morph": "",
      "inflectional_morphs": [
        "iic."
      ]
    },
    {
      "word": "AlayaH",
      "derived_stem": "Alaya",
      "base": "",
      "derivational_morph": "",
      "inflectional_morphs": [
        "m. sg. nom."
      ]
    },
    {
      "word": "AlayaH",
```

```

        "derived_stem": "Ali",
        "base": "",
        "derivational_morph": "",
        "inflectional_morphs": [
            "m. pl. nom."
        ]
    },
    {
        "word": "AlayaH",
        "derived_stem": "Ali",
        "base": "",
        "derivational_morph": "",
        "inflectional_morphs": [
            "m. pl. voc."
        ]
    }
]
}

```

With the segmentation mode as *l* or *b*, the output will have the list of top 10 segmentations for the ‘segmentation’ key, and the dictionary entries, which encapsulate the morphological analyses, of all possible segments corresponding to the top 10 segmentations will be produced for the ‘morph’ key.

Additionally, the frequency mode parameter (‘fmode’) can be altered based on the level of comparison for the ranking metrics. For all the experiments, the fmodes *w* (word), *x* (word-transition) and *n* (stem-morph) were considered. Thus, with the segmentation mode as *f* and frequency mode as *n*, we will be able to get the best possible segmentation with the best possible morphological analysis for each of the segments.

Python package for SH Results

While SH is available as a web application, its repositories can also be installed as a software.³ Binaries from these repositories that aid in performing segmentation and morphological analysis have been used to create a python package that additionally addresses the transliterations, bulk processing of the tasks, conversion to JSON format and error handling. It also handles certain input characters which are not processed by the SH engine. This package is available here:

https://github.com/SriramKrishnan8/sandhi_vicchedika.

Segmentation can be achieved through the script ‘sandhi_vicchedika.py’ by running the following command:

```
python3 sandhi_vicchedika.py <input_encoding> <output_encoding>
<segmentation_mode> <result_mode> [-t text] [-i input_file] [-o output_file]
```

The input arguments are input encoding (WX, DN, RN, SL, KH, VH), output encoding (deva, roma), segmentation mode (word, sent) and result mode (first, best). The output is a list of segmentations.

Morphological analysis can be achieved through the script ‘pada_vishleshika.py’ by running the following command:

```
python3 pada_vishleshika.py <input_encoding> <output_encoding>
<result_mode> [-t text] [-i input_file] [-o output_file]
```

The input arguments are the same as above except the segmentation mode as this performs both segmentation and morphological analysis and produces the output in JSON format. For example, running the command: ‘python3 pada_vishleshika.py WX roma best -t “rAmovanaṅgacCawī”’ yields the following result:

```
{
  "input": "rāmovanaṅgacchatī",
  "status": "success",
```

³ https://gitlab.inria.fr/huet/Heritage_Platform.git and https://gitlab.inria.fr/huet/Heritage_Resources.git

```
"segmentation": [
  "rāmaḥ vanam gacchati"
],
"morph": [
  {
    "word": "rāmaḥ",
    "stem": "",
    "root": "rā#1",
    "derivational_morph": "",
    "inflectional_morphs": [
      "pr. [2] ac. pl. 1"
    ]
  },
  {
    "word": "rāmaḥ",
    "stem": "rāma",
    "root": "",
    "derivational_morph": "",
    "inflectional_morphs": [
      "m. sg. nom."
    ]
  },
  {
    "word": "vanam",
    "stem": "vana",
    "root": "",
    "derivational_morph": "",
    "inflectional_morphs": [
      "n. sg. acc.",
      "n. sg. nom."
    ]
  },
  {
    "word": "gacchati",
    "stem": "gacch",
    "root": "gac",
    "derivational_morph": "",
    "inflectional_morphs": [
      "pr. [2] ac. 3sg."
    ]
  }
]
```

```

    {
      "word": "gacchati",
      "stem": "",
      "root": "gam",
      "derivational_morph": "",
      "inflectional_morphs": [
        "pr. [1] ac. sg. 3"
      ]
    },
    {
      "word": "gacchati",
      "stem": "gacchat",
      "root": "gam",
      "derivational_morph": "ppr. [1] ac.",
      "inflectional_morphs": [
        "n. sg. loc.",
        "m. sg. loc."
      ]
    }
  ],
  "source": "SH"
}

```

The output JSON object contains the input string, status (success, unrecognized, failure, timeout, etc.), segmentations (number of segmentations is one for the first mode and ten for the best mode), morphological analysis (list of morphological analyses of all possible segments corresponding to the segmentations, each containing word, stem, root, derivational morphological analysis and list of inflectional morphological analyses). ‘pada_vishleshika.py’ can also be used as a standalone morphological analyser that produces all possible morphological analyses for a given word.



SI No. 14420

Notification No. 1

हैदराबाद विश्वविद्यालय

UNIVERSITY OF HYDERABAD

(A Central University Established by an Act of Parliament)

P.O. CENTRAL UNIVERSITY, GACHIBOWLI, HYDERABAD - 500 046 (INDIA)

SEMESTER GRADE TRANSCRIPT**REGULAR EXAMINATION**

REG. NO. 19HSPH02
NAME OF THE STUDENT SRIRAM K
MONTH AND YEAR OF NOV 2019
COURSE Ph.D. SANSKRIT STUDIES
PARENT'S NAME J KRISHNAN / MALATHI KRISHNAN

**SEMESTER 1**

COURSE NO	TITLE OF THE COURSE	CREDITS	RESULTS
SK830	DISSERTATION RELATED READINGS	4	PASS

SEMESTER GRADE POINT AVERAGE (SGPA) :10.0

(In words) :TEN POINT ZERO

**DEPUTY REGISTRAR
(ACAD & EXAMS)**

Date Of Result Notification : Dec 13, 2019

Date Of Download : Dec 4, 2024

This is system generated certificate issued by O/o The Controller of Examinations, University of Hyderabad



SI No. 40445

Notification No. 2

हैदराबाद विश्वविद्यालय

UNIVERSITY OF HYDERABAD

(A Central University Established by an Act of Parliament)

P.O. CENTRAL UNIVERSITY, GACHIBOWLI, HYDERABAD - 500 046 (INDIA)

SEMESTER GRADE TRANSCRIPT**REGULAR EXAMINATION**

REG. NO. 19HSPH02
NAME OF THE STUDENT SRIRAM K
MONTH AND YEAR OF JUN 2020
COURSE Ph.D. SANSKRIT STUDIES
PARENT'S NAME J KRISHNAN / MALATHI KRISHNAN

**SEMESTER 2**

COURSE NO	TITLE OF THE COURSE	CREDITS	RESULTS
SK831	DISSERTATION RELATED READINGS - II	4	PASS

SEMESTER GRADE POINT AVERAGE (SGPA) :10.0

(In words) :TEN POINT ZERO

**DEPUTY REGISTRAR
(ACAD & EXAMS)**Date Of Result Notification : **Mar 25, 2021**Date Of Download : **Dec 4, 2024**

This is system generated certificate issued by O/o The Controller of Examinations, University of Hyderabad



SI No. 41308

Notification No. 1

हैदराबाद विश्वविद्यालय

UNIVERSITY OF HYDERABAD

(A Central University Established by an Act of Parliament)

P.O. CENTRAL UNIVERSITY, GACHIBOWLI, HYDERABAD - 500 046 (INDIA)

SEMESTER GRADE TRANSCRIPT**REGULAR EXAMINATION**

REG. NO. 19HSPH02
NAME OF THE STUDENT SRIRAM K
MONTH AND YEAR OF DEC 2020
COURSE Ph.D. SANSKRIT STUDIES
PARENT'S NAME J KRISHNAN / MALATHI KRISHNAN

**SEMESTER 3**

COURSE NO	TITLE OF THE COURSE	CREDITS	RESULTS
SK801	RESEARCH METHODOLOGY	4	PASS

SEMESTER GRADE POINT AVERAGE (SGPA) :10.0

(In words) :TEN POINT ZERO

**DEPUTY REGISTRAR
(ACAD & EXAMS)**Date Of Result Notification : **Mar 18, 2021**Date Of Download : **Dec 4, 2024**

This is system generated certificate issued by O/o The Controller of Examinations, University of Hyderabad



Sriram Krishnan <sriramk8@gmail.com>

Your ICON 2019 Submission (Number 37)

Organizers, ICON 2019 <icon2019@softconf.com>

Fri, Oct 25, 2019 at 6:27 PM

Reply-To: icon2019@softconf.com

To: sriramk8@gmail.com

Cc: iconnlp@iiit.ac.in

Dear Sriram Krishnan:

On behalf of the ICON 2019 Program Committee, We are delighted to inform you that the following submission has been accepted to appear at the conference:

Sanskrit Segmentation revisited

The Program Committee worked very hard to thoroughly review all the submitted papers. Please repay their efforts, by following their suggestions when you revise your paper. Also, please make sure that the length of your paper is NOT MORE than 10 pages.

When you are finished, you can upload your final manuscript at the following site:

<https://www.softconf.com/icon2019/papers/>

You will be prompted to login to your START account. If you do not see your submission, you can access it with the following passcode:

37X-G4J4H9H3G6

Alternatively, you can click on the following URL, which will take you directly to a form to submit your final paper (after logging into your account):

<https://www.softconf.com/icon2019/papers/user/scmd.cgi?scmd=aLogin&passcode=37X-G4J4H9H3G6>

The reviews and comments are attached below. Again, try to follow their advice when you revise your paper.

The instructions for the camera ready copy and a sample paper will be made available on the conference website. Please follow the instructions and upload your paper by November 15, 2019 positively.

Congratulations on your fine work. If you have any additional questions, please feel free to get in touch.

Best Regards,
Organizers, ICON 2019
ICON 2019

=====

ICON 2019 Reviews for Submission #37

=====

Title: Sanskrit Segmentation revisited
Authors: Sriram Krishnan and Amba Kulkarni

=====

REVIEWER #1

=====

What is this paper about, and what contributions does it make?

The paper describes the problem of Sanskrit segmentation. Specifically, the paper takes Sanskrit Heritage Reader as the base segmenter, which provides all the solutions but does not have a statistical method to rank these. The paper proposes a simple statistical method that achieves good accuracy.

What strengths does this paper have?

The paper is nicely written, does not assume any background on the problem and describes the approach in detail.
The method is easy and simple to implement to get a reasonable results.



Sriram Krishnan <sriramk8@gmail.com>

WSC 2023 Final Author Notification

WSC 2023 Conference Secretariat <sanskrit@kaigi.com.au>
 Reply-To: WSC 2023 Conference Secretariat <sanskrit@kaigi.com.au>
 To: Sriram Krishnan <sriramk8@gmail.com>

Thu, Sep 15, 2022 at 5:19 PM



www.wsc2021.com.au • CANBERRA
 The Australian National University
 9-13 January 2023

Final Author Notification WSC 2023

ID: 1650

Dear Mr. Sriram Krishnan,

Thank you for submitting an abstract for *The 18th World Sanskrit Conference*, now WSC 2023, to be held online from 9-13 January 2023.

Please see below the status of all abstracts submitted in earlier and current rounds. If one or more of your papers is accepted you will receive further guidance shortly on the next steps and requirements. Please view the **Paper Status** row to determine the status of your abstract(s).

Your abstract(s)

Title	Validation and Normalization of DCS corpus and Development of the Sanskrit Heritage Engine's Segmenter
Paper Status	Individual Paper Accepted
Presentation Type	Full paper
Theme	Computational Sanskrit and Digital Humanities, संस्कृतं विज्ञानतान्त्रिकी च
Presenting Author	Mr. Sriram Krishnan Affiliations: University Of Hyderabad
Biography	Sriram Krishnan, B. Tech (Computer Science and Engineering), MA (Sanskrit), from SASTRA University, pursuing his Ph.D. in Sanskrit Computational Linguistics at the Department of Sanskrit Studies, University of Hyderabad, is working on Sanskrit Sentence segmentation and compound analysis, and has presented two papers on parsing, segmentation at ISCLS-2019 and ICON-2019.

These status' are all accepted:

- Individual Paper Accepted
- Individual Paper Accepted 2022
- Amended and resubmitted 2022
- Individual Paper Accepted - no change

These status' are NOT accepted:

- Not Accepted
- Not Accepted 2022

You may also see:

- Under Review 2022 (the reviews for your section are not yet complete)
- Withdrawn

Please note there has been a large number of high quality abstracts and it is not possible to accommodate all abstracts in the program. We are unable to provide individual feedback regarding the reviewer results for each abstract.

Accepted authors are required to register for the conference. Not accepted authors are also encouraged to register and participate in the conference. Please click here to register if you haven't already done so: [WSC 2023 Online Registration](#).

Please do not forward this link to others - it is unique, personalised with your details and linked to your abstract.

If you have any questions about the conference or your paper status please do not hesitate to contact us. We look forward to seeing you online in January 2023.

Kind regards,



WSC 2023 Conference Secretariat
Kaigi Conferencing and Events
Level 1, The Realm
[18 National Circuit](#)
[Barton ACT 2600](#)
Phone: 02 6198 3218
Email: sanskrit@kaigi.com.au



Sriram Krishnan <sriramk8@gmail.com>

Language Resources and Evaluation: Decision on your manuscript

Language Resources and Evaluation <Lavanya.Ravi@springernature.com>

Tue, Jan 16, 2024 at 4:53 PM

To: sriramk8@gmail.com

Ref: Submission ID 5ffda941-9b25-4793-bd49-d99cb1a26c55

Dear Dr Krishnan,

Re: "Normalized Dataset for Sanskrit Word Segmentation and Morphological Parsing"

We're delighted to let you know that your manuscript has been accepted for publication in Language Resources and Evaluation.

Prior to publication, our production team will check the format of your manuscript to ensure that it conforms to the journal's requirements. They will be in touch shortly to request any necessary changes, or to confirm that none are needed.

Checking the proofs

Once we've prepared your paper for publication, you will receive a proof. At this stage, for the main text, only errors that have been introduced during the production process, or those that directly compromise the scientific integrity of the paper, may be corrected.

As the corresponding (or nominated) author, you are responsible for the accuracy of all content, including spelling of names and current affiliations.

To ensure prompt publication, your proofs should be returned within two working days.

Publication policies

Acceptance of your manuscript is conditional on all authors agreeing to our publication policies at:

<https://www.springernature.com/gp/policies/editorial-policies>

Language Resources and Evaluation is a hybrid journal. This means when the journal accepts research for publication, the article may be published using either immediate gold open access or the subscription publishing route. For further information please visit <https://www.springernature.com/gp/open-research/about/green-or-gold-routes-to-OA/hybrid-options>

Once again, thank you for choosing Language Resources and Evaluation, and we look forward to publishing your article.

Kind regards,

Sara Goggi
Editor
Language Resources and Evaluation

Reviewer Comments:

Reviewer 1

The revision has greatly improved in clarity and my previous doubts have been addressed. I thus recommend acceptance.

In particular, the revision

- improved upon explaining its relation to previous work and discussing the limitations of previous work in detail;
- makes Sanskrit examples more accessible by giving English translations;
- greatly improved the explanation of the evaluation. The role of the ranking mechanism is now clear to me and also the comparison to baselines appears to be fair.

Minor changes also improve clarity and in summary I find the methods and contribution of this paper easy to follow. The presented dataset makes a valuable contribution to the availability of digital Sanskrit resources, which are

25/11/2024, 14:04

Gmail - Language Resources and Evaluation: Decision on your manuscript

expected to be of great use both for developing data-driven models as well as for use in large-scale linguistic analysis.

Reviewer 3

The authors have taken care of my comments in this revised version of the manuscript. I also see that the manuscript as a whole has improved significantly.

Source details

Language Resources and Evaluation

Formerly known as: Computers and the Humanities

Years currently covered by Scopus: from 1996 to 2002, from 2005 to 2024

Publisher: Springer Nature

ISSN: 1574-020X

Subject area: Arts and Humanities: Language and Linguistics Social Sciences: Linguistics and Language
Social Sciences: Education Social Sciences: Library and Information Sciences

Source type: Journal

[View all documents >](#) [Set document alert](#)  [Save to source list](#)

CiteScore 2023
6.5 ⓘ

SJR 2023
0.786 ⓘ

SNIP 2023
2.259 ⓘ

CiteScore CiteScore rank & trend Scopus content coverage

CiteScore 2023 

6.5 = $\frac{1,222 \text{ Citations } 2020 - 2023}{187 \text{ Documents } 2020 - 2023}$

Calculated on 05 May, 2024

CiteScoreTracker 2024 ⓘ

6.9 = $\frac{1,234 \text{ Citations to date}}{180 \text{ Documents to date}}$

Last updated on 05 November, 2024 • Updated monthly

CiteScore rank 2023 ⓘ

Category	Rank	Percentile
Arts and Humanities		
Language and Linguistics	#30/1088	97th
Social Sciences		
Linguistics and Language	#34/1167	97th

[View CiteScore methodology >](#) [CiteScore FAQ >](#) [Add CiteScore to your site !\[\]\(df47d6bec273bbb8b349135fff3a20f7_img.jpg\)](#)



Normalized dataset for Sanskrit word segmentation and morphological parsing

Sriram Krishnan¹ · Amba Kulkarni¹ · Gérard Huet²

Accepted: 16 January 2024

© The Author(s), under exclusive licence to Springer Nature B.V. 2024

Abstract

Sanskrit processing has seen a surge in the use of data-driven approaches over the past decade. Various tasks such as segmentation, morphological parsing, and dependency analysis have been tackled through the development of state-of-the-art models despite working with relatively limited datasets compared to other languages. However, a significant challenge lies in the availability of annotated datasets that are lexically, morphologically, syntactically, and semantically tagged. While syntactic and semantic tags are preferable for later stages of processing such as sentential parsing and disambiguation, lexical and morphological tags are crucial for low-level tasks of word segmentation and morphological parsing. The Digital Corpus of Sanskrit (DCS) is one notable effort that hosts over 650,000 lexically and morphologically tagged sentences from around 250 texts but also comes with its limitations at different levels of a sentence like chunk, segment, stem and morphological analysis. To overcome these limitations, we look at alternatives such as Sanskrit Heritage Segmenter (SH) and *Saṃsādhani* tools, that provide information complementing DCS' data. This work focuses on enriching the DCS dataset by incorporating analyses from SH, thereby creating a dataset that is rich in lexical and morphological information. Furthermore, this work also discusses the impact of such datasets on the performances of existing segmenters, specifically the Sanskrit Heritage Segmenter.

Keywords Word segmentation · Morphological parsing · Datasets · Sanskrit computational linguistics

The details of the DCS-SH alignment (code, data and references) are released publicly and can be accessed here: https://github.com/SriramKrishnan8/dcs_sh_alignment.

Extended author information available on the last page of the article

Justification for Similarity Index > 10 %

Mr. Sriram K's Ph.D. thesis reported a similarity index of 38%. Of these, the following three resources majorly constitute 33% of the similarity index:

1. The journal publication entitled "Normalized Dataset for Sanskrit Word Segmentation and Morphological Parsing" which forms the major part of the 3rd, 4th and 5th chapters of his thesis. The full citation for the journal paper and link are provided below:
Krishnan S., Kulkarni A., and Huet, G. (2024) Normalized Dataset for Sanskrit Word Segmentation and Morphological Parsing. In: Language Resources and Evaluation, Springer Nature, ISSN 1574-0218. <https://link.springer.com/article/10.1007/s10579-024-09724-0>
2. The second resource is the "assets.researchsquare.com" where a copy of the journal paper was uploaded, after submitting it to the journal "Language Resources and Evaluation", for wider dissemination of the research findings among the peer group.
3. The third resource is a repository of the research lab INRIA, Paris, France, where the third author of the above journal paper is an emeritus director of research.

These three similarity indices account for the 33% of the similarity out of 38%.

Amba P.

Signature of the Supervisor

(Amba Kulkarni)

Professor
Department of Sanskrit Studies
University of Hyderabad
P.O. Central University
Hyderabad-500 046, TS

Sanskrit Segmentation and Morphological Analysis: Normalized Datasets and a Ranking-based approach

by Sriram K

Submission date: 11-Dec-2024 08:01AM (UTC+0530)

Submission ID: 2548514431

File name: Sriram_K.pdf (1.64M)

Word count: 40080

Character count: 207377

Sanskrit Segmentation and Morphological Analysis: Normalized Datasets and a Ranking-based approach

ORIGINALITY REPORT

38%

SIMILARITY INDEX

28%

INTERNET SOURCES

32%

PUBLICATIONS

1%

STUDENT PAPERS

PRIMARY SOURCES

1

Sriram Krishnan, Amba Kulkarni, Gérard Huet.
"Normalized dataset for Sanskrit word
segmentation and morphological parsing",
Language Resources and Evaluation, 2024
Publication

21%

Amba P

2

assets.researchsquare.com
Internet Source

10%

Amba P

3

gallium.inria.fr
Internet Source

2%

Amba P

4

aclanthology.org
Internet Source

1%

Professor
Department of Sanskrit Studies
University of Hyderabad
P.O. Central University
Hyderabad-500 046, TS

5

pdffox.com
Internet Source

1%

6

www.arxiv-vanity.com
Internet Source

1%

7

deepai.org
Internet Source

<1%

8

arxiv.org
Internet Source

<1 %

9

ltc.amu.edu.pl

Internet Source

<1 %

10

export.arxiv.org

Internet Source

<1 %

11

sanskrit.inria.fr

Internet Source

<1 %

12

jlm.ipipan.waw.pl

Internet Source

<1 %

13

yquem.inria.fr

Internet Source

<1 %

14

dspace.uohyd.ac.in

Internet Source

<1 %

15

"Advances in Cybernetics, Cognition, and Machine Learning for Communication Technologies", Springer Science and Business Media LLC, 2020

Publication

<1 %

16

Sushant Dave, Arun Kumar Singh, Dr. Prathosh A.P., Prof. Brejesh Lall. "Neural Compound-Word (Sandhi) Generation and Splitting in Sanskrit Language", Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management

<1 %

of Data (8th ACM IKDD CODS & 26th
COMAD), 2021

Publication

17

Amrita Varshini E R, Jayashree Nair. "Word Segmentation and Sandhi Resolution on Ayurveda Classical Scriptures", 2023 International Conference on Inventive Computation Technologies (ICICT), 2023

Publication

<1 %

18

anthology.aclweb.org

Internet Source

<1 %

19

hal.archives-ouvertes.fr

Internet Source

<1 %

20

"Chinese Computational Linguistics", Springer Science and Business Media LLC, 2019

Publication

<1 %

21

Yingfeng Chen, Rui Wang, Mengjun Ming, Shi Cheng, Yiping Bao, Wensheng Zhang, Chi Zhang. "Constraint multi-objective optimal design of hybrid renewable energy system considering load characteristics", Complex & Intelligent Systems, 2021

Publication

<1 %

22

web.archive.org

Internet Source

<1 %

23

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, Pawan Goyal. "A Graph-

<1 %

Based Framework for Structured Prediction Tasks in Sanskrit", Computational Linguistics, 2021

Publication

24

S. M. H. Mirbagheri. "3D computer simulation of melt flow and heat transfer in the lost foam casting process", International Journal for Numerical Methods in Engineering, 10/07/2003

Publication

<1 %

25

Siyuan Frank Yang, Wei-Ting Kary Chien. "Electromigration Lifetime Optimization by Uniform Designs and a New Lifetime Index", IEEE Transactions on Reliability, 2015

Publication

<1 %

Exclude quotes On

Exclude matches

< 14 words

Exclude bibliography On