ISCLS 2024

Proceedings of the 7th International Sanskrit Computational Linguistics Symposium

15–17 February, 2024

Auroville,

Puducherry, India

Sponsors

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN: 979-8-89176-027-1

Preface

Welcome to the **7th International Sanskrit Computational Linguistics Symposium (ISCLS 2024)** at Auroville, Puducherry, India. The aim of ISCLS is to bring together researchers interested in any aspects of Sanskrit Computational Linguistics. Full papers were invited on original and unpublished research on various aspects of Computational Linguistics and Digital Humanities related to Sanskrit (Classical and Vedic), Prakrit, Pali, Buddhist Hybrid Sanskrit, etc.

After a rigorous review process that constituted at least 3 reviews, a total of 10 full papers were accepted for presentation at the symposium. These include a variety of themes ranging from using web-based platforms for Sanskrit processing and teaching to distances between languages to identifying word senses, discourse relations, figures of speech, etc. In addition to full papers, the symposium features 18 demonstrations of various Sanskrit and Pali-based tools. An interesting panel discussion on “AI and Sanskrit” and 3 invited talks on aspects of Sanskrit language are part of the symposium as well.

I extend heartfelt appreciation to the Program Committee members for their active involvement in meticulously reviewing and refining the program details, contributing significantly to the success of the symposium. Their diligence and expertise ensured the high quality of the proceedings. I am also grateful to the esteemed Steering Committee for their invaluable guidance and strategic oversight throughout the planning and execution of this event. Their collective wisdom and leadership played a pivotal role in shaping the conference's direction and ensuring its alignment with our overarching objectives.

I also thank the Organizing Chair for planning and seamless execution of the symposium. Additionally, I would like to sincerely thank the Web Chair for his expertise and efforts in managing the webpage of the conference, and for creating the proceedings.

Arnab Bhattacharya
Program Committee Chair
7th ISCLS

Organization

Program Committee Chair

Arnab Bhattacharya
IIT Kanpur, India

Organising Chair

Martin Gluckman
Sanskrit Research Institute, India

Web Chair

Hrishikesh Terdalkar
IIT Kanpur, India

Steering Committee

Amba Kulkarni
University of Hyderabad, India

Arnab Bhattacharya
IIT Kanpur, India

Brendan Gillon
McGill University, Montreal, Canada

Gérard Huet
INRIA, Paris, France

Malhar Kulkarni
IIT Bombay, India

Pawan Goyal
IIT Kharagpur, India

Peter Scharf
The Sanskrit Library, USA

Technical Program Committee

Amba Kulkarni
University of Hyderabad, India

Amrith Krishna
Learno.AI, India

Arjuna S R
MAHE Bengaluru, India

Arnab Bhattacharya
IIT Kanpur, India

Brendan Gillon
McGill University, Montreal, Canada

Chaitali Dangarikar
IIT Kanpur, India

Ganesh Ramakrishnan
IIT Bombay, India

Gérard Huet
INRIA, Paris, France

Malhar Kulkarni
IIT Bombay, India

Oliver Hellwig
University of Zurich, Switzerland

Patrick McAllister
Austrian Academy of Sciences, Austria

Pavankumar Satuluri
IIT Roorkee, India

Pawan Goyal
IIT Kharagpur, India

Peter Scharf
The Sanskrit Library, USA

Sebastian Nehrdich
University of Hamburg, Germany

Shivani V
Karnataka Sanskrit University, India

Tanuja P Ajotikar
The Sanskrit Library, USA

Contents

Word Sense Alignment of Sanskrit Lexica

Dhaval Patel
Department of Sanskrit Studies,
University of Hyderabad
drdhaval2785@gmail.com

Amba Kulkarni
Department of Sanskrit Studies,
University of Hyderabad
ambakulkarni@uohyd.ac.in

Abstract

Word sense alignment is a field of study in which lexical resources or texts are aligned at the level of word sense rather than the word. The present paper tries to evaluate the possibility of mechanically aligning Sanskrit lexica at the level of word sense computationally.

1 Introduction

Sanskrit, an ancient Indian language, has been a medium of transmission of knowledge in various fields of study for centuries. Compilation of word lists in Sanskrit commenced at an early date as it was found necessary to access the old literature such as Vedic literature, while the language was undergoing some transformations with meaning shifts. The lexical resources known as kośas were developed. They are of two types – (1) Samānārthaka kośas and (2) Anekārthaka kośas. Samānārthaka kośas enlist the synonyms together. The synonyms are arranged following some theme, semantic criterion, or ontological classification scheme. For example, in the most famous samānārthaka kośa viz. Amarakośa, the words are arranged in three kāṇḍas and further within the kāṇḍas, the headwords are arranged based on either semantic or ontological properties. Anekārthaka kośas enlist different meanings of a given word. The words may or may not have any alphabetic arrangement. Both kinds of kośas were meant to be memorized, applied to texts, and cited as and when the usage of the said word in the literature was to be justified in a commentary. Therefore, the kośas were almost invariably in a verse form. Vogel (2015) has given a comprehensive coverage of these Sanskrit kośas and commentarial literature thereon.

Because of the influence of Western lexicography, a few Sanskrit-Sanskrit dictionaries like Vācaspatyam and Śabdakalpadruma were also compiled on the lines with the Western methodology of arranging headwords alphabetically and in prose form. Several bilingual Sanskrit dictionaries such as Sanskrit-English, Sanskrit-French, and Sanskrit-German were created starting from the early 19^{th} century. Almost all of the major dictionaries that are free from copyright are available on the Cologne Digital Sanskrit Dictionaries website (CDSD, 2023). This digitized data has various levels of markup. In the recent years Huet (2019) has developed a digital Sanskrit-French dictionary where the lexical items are directly linked to the inflectional and derivational morphology.

Some of these dictionaries, in addition to providing the meaning of Sanskrit words in the target language, also provide citations from Sanskrit texts. The citations in different dictionaries vary. These citations play an important role in understanding the context in which the sense is being used. Aligning the senses of different dictionaries would provide us with more than one example sentence for each sense to understand the context and the semantic criterion that decides the sense of the word in a given usage. Further, with the availability of word embeddings for words in several languages such as Hindi, English, French, German, etc. if the senses in Sanskrit bilingual dictionaries are aligned, one can take advantage of the

existing modelling of the world knowledge and the domain knowledge of other languages to disambiguate Sanskrit words. Such sense mapping would be useful in the Machine Translation system, for Information Retrieval, and even for a casual learner of the Sanskrit language. This motivated us to look at the problem of aligning various Sanskrit bilingual dictionaries according to the senses.

In what follows, we first explain the word sense alignment problem, and the challenges therein. This is followed by the discussion on the methodology followed for automatic sense alignment. In section 4 we discuss the sense alignment of two dictionaries Sanskrit-English and Sanskrit-Hindi by Apte. The results of the alignment algorithms are extended to other pairs of dictionaries, which is the topic of section 5. Finally we discuss other possible ways of alignment before concluding.

2 Word Sense Alignment

Word sense alignment, also known as sense alignment or sense mapping matches two entries from two lexical resources based on the sense the two entries express. Word sense alignment emerged out of various efforts toward the word sense disambiguation (WSD) problem. WSD is an important task for several NLP applications such as Machine Translation, Information Retrieval, Question Answering, Summarisation, and so on. At the same time, it is one of the most difficult problems in the field of Natural Language Processing (NLP). It is considered as being an AI-complete problem (Agirre and Edmonds, 2007). The difficulties arise due to poor understanding of the process involved. Various factors such as linguistic, contextual, domain-specific, cultural, and world knowledge contribute to the process of manual word sense disambiguation. In the case of resource-rich languages such as English, there are several lexical resources with varied granularity such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998), ConceptNet (Speer et al., 2017), VerbNet (Schuler, 2005) etc., and sense-tagged corpora available in digital media. This resulted in several efforts aiming at the alignment of such resources known as Word Sense Aligned (WSA) resources. The development of Euro-WordNet (Vossen, 1998) and Indo-WordNet (Bhattacharyya, 2010) are also steps towards generating Word sense-aligned lexical resources so that the sense-tagged corpus in one language can become available in another with minimum effort. In the recent years, Word Sense Alignment has gained importance. Languages with low resources would like to take advantage of the resources available in resource-rich languages, by aligning their resources to those of the rich languages. For example Salgado et al. (2020) describes the challenges of word sense alignment of Portuguese Language Resources. Joshi et al. (2012) present a heuristic approach to link English and Hindi WordNets by linking their senses. Two closely related Czech lexical resources VALLEX[1] and PDT-VALLEX[2] were aligned fully automatically (Bejcek et al., 2014). Johansson and Pina (2015) used word sense embeddings to automatically link the Swedish language banks.

During his post-doctoral fellowship in 2012 at Inria, Pawan Goyal aligned the Sanskrit Heritage dictionary with an XML version of Monier-Williams available at CDSD (Goyal et al., 2012). Goyal used the online google translator to translate the French entries into English and then aligned them with the entries in Monnier Williams' Sanskrit-English dictionary, by manually aligning the entries wherever there were ambiguities/multiple choices available. The alignment process is incremental and thus may be iterated on successive versions of the Sanskrit Heritage dictionary.

[1] http://ufal.mff.cuni.cz/vallex/2.6/
[2] http://lindat.mff.cuni.cz/services/PDT-Vallex/

2.1 Challenges

The conceptual space is a continuum that is divided into discrete units by the lexicon of a language. Since the lexicon is denumerably finite, a word represents a piece of continuous conceptual space and not a discrete point. This sometimes leads to one word representing a spectrum of meanings. Such words are termed polysemous words. Sometimes, more than one lexical unit produces the same word form. Such word forms are called homonyms. Among the homonymous and polysemous meanings, typically the homonyms are provided with different headword entries, while the polysemous meanings are clubbed under a single head. Within polysemous meanings, the granularity is decided by the lexicographer. Deciding the granularity of the meaning is not trivial. It is not at all clear when a sense of the word should be treated as a separate meaning and when it should be subsumed within an already existing meaning. Further deciding between a polysemy and homonymy is subjective due to the fuzzy boundary between them. Another factor is the inclusion of metaphoric meanings in the dictionary. Indian tradition discusses three types of meanings viz. abhidhā (literal), lakṣaṇā (metaphoric or secondary) and vyañjanā (suggestive). While it is impossible to provide the suggestive meanings, which are subjective in nature, and also depend on the context, the lexicographers do consider the secondary or suggestive meanings for inclusion in the dictionaries. Even in the case of dictionaries from the same lexicographer, the intended audience, printing or economic considerations may force the lexicographer to deal with sense granularity in different ways across different dictionaries. Therefore, the choice of sense granularity is mostly left to the discretion of the lexicographer, as has been observed through various lexical resources. Because of these reasons, the word sense mapping between different lexical resources is not trivial.

3 Methodology

Manually aligning lexical resources at the word sense level is a very laborious task. It would also require the person to be well versed in two languages e.g. aligning a Sanskrit-English and Sanskrit-Hindi dictionary would require the person to know at least English and Hindi, and preferably Sanskrit too. For a resource-starved languages like Sanskrit, this may be very costly and time-consuming.

The present work focuses on finding out the similarity between different meanings of a given word and present the human annotator with a similarity score or a confidence score, so that the annotator may devote more time to the places where the machine performs with a low confidence level. We also aim at finding a more or less language-agnostic way of automatically or semi-automatically aligning lexical resources at the word sense level so that it can be extended to other language pairs.

For any mechanical mapping between entries from two dictionaries to be successful, either both the target languages need to be the same or a model trained on both languages to identify similar concepts across both languages is needed. The first approach is simpler. Because of the advancement in machine translation technologies and publicly available resources such as Google Translate[3], it has become possible to translate various texts from one language to another. Thus, in the absence of models trained in two different languages, one can still use Google Translate to identify similar concepts across languages. The task of finding out the similarity between two documents (in our case, the meaning of the word for a given sense) is a common theme in information retrieval (IR), topic modeling, ontology matching, etc. There are various algorithms which have already been tested for the same.

Bär et al. (2013) have enumerated and implemented the following similarity algorithms in their software: Longest Common Substring, Greedy String Tiling (Wise, 1996), Jaro (1989),

[3]https://translate.google.com/ accessed on 20 September 2023

Jaro-Winkler (Winkler, 1990), Monge and Charles (1997), Levenshtein (1966), Jiang and Conrath (1997), Resnik (1995), Latent Semantic Analysis (Landauer et al., 1998) and Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007). As the dictionary meanings are relatively small chunks of text, with sizes ranging upto two or three sentences at maximum, structural and stylistic similarity measures mentioned in the said paper are not of much relevance to the task at hand. Semantic similarity measures presume some graph-like structure and use the structure of those graphs to find out the similarity or nearness between two nodes. These work best when there is some ontological representation of the world knowledge or some hierarchy of the word/word senses and their relationship is explicitly coded. In an alphabetically arranged dictionary, such a relationship is almost non-existent. Therefore, these measures were not tried. Latent Semantic Analysis (LSA) and explicit semantic analysis (ESA) require a lot of computational resources. Training and running these algorithms on a large corpus like two full-fledged dictionaries will be computationally too heavy. LSA may be able to identify similarities between 'child' and 'offspring', which a normal text-based scorer may miss. However, due to limited computational resources, we have not tried them either. Other than these measures, there are following string-based measures implemented by rapidfuzz library[4] - Damerau (1964), Hamming (1950), Indel, OSA, Prefix and Postfix.

The present paper focuses on the usability of string-based similarity measures for finding out the mapping of word senses. We present here our efforts towards the word sense alignment of Sanskrit lexical resources. We present three case studies. The first one is with two different target languages but the same compiler. Here we have chosen Sanskrit-English[5] and Sanskrit-Hindi dictionaries of Apte[6]. Since the second dictionary is based on the first one, the assumption is there would be a good chance of getting one-one mapping. The second pair is with the same target language but different compilers. Here we have chosen Sanskrit-English by two different compilers – Wilson (1832)[7] and Yates (1846)[8]. The third one was a pair of monolingual English lexical resources viz. Webster's Unabridged Dictionary of the English Language (Webster, 1900)[9] and English Wordnet (Miller et al., 1990)[10].

4 WSA of Sanskrit-English and Sanskrit-Hindi of Apte

Apte Sanskrit-English (AP90) dictionary (Apte, 1890) has been used in this experiment. The later 1957 version (AP) of the dictionary (Apte et al., 1957) is still under copyright. Therefore CDSD does not have its data for open usage. AP90 is not fully marked up to show different word senses separately. It has some rudimentary markup or patterns by the help of which crude parsing was done and word meanings were separated. Apte Sanskrit-Hindi (ASH) dictionary (Apte, 2007) is a Hindi translation of Apte's Sanskrit-English dictionary. It is not an exact translation. Many of the words have been omitted, and many meanings have been merged, deleted, or separated. It seems that ASH had the advantage of using the data of the 1957 edition too. Therefore, the new words or meanings added in that edition are also used in ASH. At the same time, ASH has been made more concise. Therefore, multiple meanings have been combined together. Rare meanings have been dropped altogether too. Therefore, it was not trivial to align the word senses in these two dictionaries, and hence, these were taken up to attempt word sense level alignment between them.

[4]https://pypi.org/project/rapidfuzz/

[5]https://www.sanskrit-lexicon.uni-koeln.de/scans/AP90Scan/2020/web/webtc/download.html

[6]Developed by the SHMT (Sanskrit-Hindi Machine Translation) consortium during 2008-2011, now a part of Samsaadhanii Platform at https://sanskrit.uohyd.ac.in/scl/

[7]https://www.sanskrit-lexicon.uni-koeln.de/scans/WILScan/2020/web/webtc/download.html

[8]https://www.sanskrit-lexicon.uni-koeln.de/scans/YATScan/2020/web/webtc/download.html

[9]https://www.gutenberg.org/files/29765/29765-0.txt

[10]https://github.com/fluhus/wordnet-to-json/releases/download/v1.0/wordnet.json.gz

4.1 Gold Standard Data

As there is no previously existing gold standard data regarding word sense alignment of unstructured lexical resources like a dictionary pair, a manual gold standard data was created by selecting a random starting point and taking roughly 1000 ASH entries starting therefrom (See Table 1). Corresponding entries of AP90 were also taken up (See Table 2).

Head word	Hindi sense_id (ASH)	Hindi Meaning (ASH)
आकल्पः (Ākalpaḥ)	247	आभूषण, अलंकार
आकल्पः (Ākalpaḥ)	248	वेशभूषा
आकल्पः (Ākalpaḥ)	249	रोग, बीमारी

Table 1: sample ASH entries

Head word	English sense_id (AP90)	English Meaning (AP90)
आकल्पः (Ākalpaḥ)	440	An ornament, decoration
आकल्पः (Ākalpaḥ)	441	Dress (in general), accoutrement
आकल्पः (Ākalpaḥ)	442	Sickness, disease
आकल्पः (Ākalpaḥ)	443	Adding to, increasing

Table 2: sample AP90 entries

Every word sense had been given a unique identifier for both dictionaries. A manual examination of the data was done and a manual mapping was created. As and when some parsing error was detected in the data, the same was manually corrected. Sample entries of the sense alignment of entries from ASH and AP90 are shown in Table 3.

Head word	Hindi sense_id (ASH)	English sense_id (AP90)
आकल्पः (Ākalpaḥ)	247	440
आकल्पः (Ākalpaḥ)	248	441
आकल्पः (Ākalpaḥ)	249	442
आकल्पः (Ākalpaḥ)	-	443

Table 3: Gold Standard Data for Alignment of ASH and AP90

Since the two dictionaries selected had different target languages, for aligning the entries, we decided to use Google Translate to translate the meanings of AP90 into Hindi. In Table 4, column GSH shows the Google translation of the entries in AP90 into Hindi. The task at hand is to map the English sense_id to Hindi sense_id using GSH. Please note that sometimes Google Translate does not translate some difficult words like 'accoutrement' and leave them as they are, when processed via bulk upload.

Head word	English sense_id (AP90)	English Meaning (AP90)	GSH
आकल्पः (Ākalpaḥ)	440	An ornament, decoration	एक आभूषण, सजावट
आकल्पः (Ākalpaḥ)	441	Dress (in general), accoutrement	पोशाक (सामान्य रूप से), accoutrement
आकल्पः (Ākalpaḥ)	442	Sickness, disease	रोग, रोग
आकल्पः (Ākalpaḥ)	443	Adding to, increasing	जोड़ना, बढ़ाना

Table 4: sample AP90 entries along with their Hindi Translations

Similarly, entries of ASH were translated into English with the help of Google translate. Please see column GSE of Table 5.

Head word	Hindi sense_id (ASH)	Hindi Meaning (ASH)	GSE
आकल्पः (Ākalpaḥ)	247	आभूषण, अलंकार	jewelery , Ornament
आकल्पः (Ākalpaḥ)	248	वेशभूषा	Costumes
आकल्पः (Ākalpaḥ)	249	रोग, बीमारी	Disease , Disease

Table 5: sample ASH entries along with their English Translations

In the next section, we present various algorithms, and their performance on the gold standard data.

4.2 Algorithms

Our algorithms are based on simple string-level similarity measures. We define four different units of comparison, and four different units of measure for comparison resulting in 16 different algorithms. We describe them below.

4.2.1 unit of comparison

The three basic units we propose are words, shingles (n-grams at character levels), and syllables. While glancing at the ASH entries with GSH manually, we also realized that in the case of languages like Hindi, depending upon the presence of post-positions, the last character of the word is changed as in 'baharā' (बहरा) versus 'bahare' (बहरे). Hence we decided to consider a word with the last character trimmed also as a unit of comparison.

4.2.2 measure of comparison

We have identified four different measures for calculating the similarity. Suppose the meanings from two dictionaries are stored as a list of words L_1 and L_2. l_1 and l_2 are sets of unique words amongst L_1 and L_2 respectively. In the following notation, $|A|$ denotes the cardinality of set A. The four different measures are defined as

$$m_1 = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}$$

$$m_2 = \frac{|l_1 \cap l_2|}{|l_1 \cup l_2|}$$

$$m_3 = \frac{|L_1 \cap L_2|}{|L_1|}$$

$$m_4 = \frac{|l_1 \cap l_2|}{|l_1|}$$

The phrases describing the senses are tokenised and stop-words are removed. In the case of English, all the words are converted to lower case. Let us assume the two senses that need to be aligned are 'space, place in general' and 'free space or vacuum'. As a first step the phrases are tokenised and the stop words are removed, and the words are changed to lower case. This results into two word lists

L_1 = [“space”, “place”, “general”], and
L_2 = [“free”, “space”, “vacuum”]

As there are no duplicate words in any of these two word lists, $l_1 = L_1$ and $l_2 = L_2$. Thus, for the above lists $m_1 = \frac{1}{6}$; $m_2 = \frac{1}{5}$; $m_3 = \frac{1}{3}$; $m_4 = \frac{1}{3}$

The shingles for each word are the all possible n-grams of characters. Thus,
shingles(“space”) = [“s”, “sp”, “spa”, “spac”, “space”, “p”, “pa”, “pac”, “pace”, “a”, “ac”, “ace”, “c”, “ce”, “e”].

The trimmed words are obtained by trimming the last character of the word. So the trimmed word list for L_1 is [“spac”, “plac”, “genera”]. While, we do not see this trimmed word list of any advantage in the case of English, for languages like Hindi these are useful. For example, in the word mapping the words ‘baharā’ (**बहरा**) and ‘bahare’ (**बहरे**) will not match, but after trimming the last phoneme, both the words will match ‘bahar’.

With the 4 units of comparison and 4 units of measures of similarity, there are 16 different measures for judging the similarity between the two senses. These 16 measures are shown in Table 6.

unit	m_1	m_2	m_3	m_4
word	CR1	CR2	CR3	CR4
shingles	CR5	CR6	CR7	CR8
trimmed word	CR9	CR10	CR11	CR12
syllable	CR13	CR14	CR15	CR16

Table 6: Metrics used for evaluation

A threshold of 0.2 was defined to ignore the mappings with low similarity score. Another measure delta was also calculated. It is the difference between the word sense pair across dictionaries with the highest similarity score and the second best pair. If delta is high, it means that the pair at the first rank is ahead of the second rank comfortably. A threshold value of delta was kept at 0.1.

4.3 Evaluation on Gold Data

After setting these thresholds, the comparison of the results of all algorithms was made. The gold standard comprises of 2022 word sense pairs manually validated. The results of some of the standard algorithms implemented by rapidfuzz library are shown in Table 7.

Algorithm	Pairs identified	Percentage	Algorithm	Pairs identified	Percentage
Levenshtein	1793	88.67%	Damerau	1794	88.73%
Hamming	1688	83.48%	Indel	1822	90.11%
Jaro	1816	89.91%	Jaro-Winkler	1818	89.91%
OSA	1793	88.67%	Prefix	1749	86.50%
Postfix	1690	83.58%			

Table 7: Percentage of word sense pairs correctly identified from gold standard data by already existing algorithms

With the same thresholds, the results obtained from algorithms CR1 to CR16 are shown in Table 8.

Algorithm	Pairs identified	Percentage	Algorithm	Pairs identified	Percentage
CR1	1866	92.28%	CR9	1861	92.04%
CR2	1871	92.53%	CR10	1865	92.24%
CR3	1855	91.74%	CR11	1848	91.39%
CR4	1856	91.79%	CR12	1847	91.35%
CR5	1856	91.79%	CR13	1847	91.35%
CR6	1887	**93.32%**	CR14	1878	92.87%
CR7	1858	91.89%	CR15	1841	91.04%
CR8	1863	92.14%	CR16	1851	91.54%

Table 8: Percentage of word sense pairs correctly identified from gold standard data by various algorithms

As can be seen from the results, CR6 gave the best result of all the algorithms. Therefore, the algorithm CR6 was selected out of these algorithms. CR6 makes use of shingles and hence captures various features like terminal case removal, textual similarity between tatsama words and tadbhava words, common verb or common noun in compounds etc. This may be the reason why CR6 gives better result than other algorithms.

4.4 Evaluation on complete dictionaries

CR6 was applied to the complete dictionaries ASH (D1) and AP90 (D2). As AP90 definitions are in English language and ASH definitions are in Hindi language, both were translated with the help of Google Translate and an English version of ASH (E) and a Hindi version of AP90 (H) were created. D1 and H were compared against each other (both with Hindi definitions) and D2 and E were compared against each other (both with English definitions). Having comparisons with both the languages helped in a big way. There are cases where one language is insufficient to map satisfactorily, but the other language could map without any difficulty. Let us see such a case with an example.

D1.91	जो चुराये जाने के योग्य न हो, या हटाये जाने अथवा दूर ले जाये जाने के योग्य न हो
D1.92	श्रद्धालु, निष्ठावान्
D1.93	दृढ,अविचल,अननुनेय
D1.94	पहाड़

Table 9: Entry of the word '**अहार्य**' in the ASH (D1) dictionary

D2.1916	not to be stolen, removed or carried
D2.1917	not to win (by fraud), devoted, loyal
D2.1918	firm, steadfast, hard
D2.1919	a mountain

Table 10: Entry of the word '**अहार्य**' in the AP90 (D2) dictionary.

As can be seen from the contents the four senses of D1 correspond sequentially to the four entries of D2.

Now, let us look at the Google translations of D1 into English and D2 into Hindi.

Had we used only translation of AP90 into Hindi through Google translator, and compared it with the entries in ASH, the words '**पहाड़**' (D1.94) and '**एक पर्वत**' (H.1919) will not get good similarity score. However, the same words when translated to English will be highly similar viz. 'a mountain' (D2.1919) and 'Mountain' (E.94). Therefore, the similarity score with English as the destination language will be very high. Similarly, '**श्रद्धालु, निष्ठावान्**' will not match '**समर्पित,**

E.91	Unstealable, or not capable of being removed or taken away
E.92	Devotees , loyal
E.93	Strong , motionless , irresistible
E.94	Mountain

Table 11: Entry of ASH translated to English via Google Translate (E)

H.1916	चोरी, हटाया या ले जाने के लिए नहीं
H.1917	जीतने के लिए नहीं (धोखाधड़ी से), समर्पित , वफादार
H.1918	दृढ़, अडिग, कठोर
H.1919	एक पर्वत

Table 12: Entry of AP90 translated to Hindi via Google Translate (H)

वफादार' much at character level, but 'devoted, loyal' will match 'Devotees, loyal' at character level. There are also cases where Hindi fares better. Mapping '**फँसा हुआ**' with '**फँसा हुआ**' is easier than mapping 'trapped' and 'entangled' (D1.342 and D2.556). Thus, using two languages helps us to take care of some cases where one language uses different synonyms and the other language has only one word for the concept or may have used the same word out of available synonyms.

Creating mapping with two languages also gives us some more benefits. It give us more confidence about a given mapping if both the languages give the same mapping. Based on these insights, an analysis of the mappings of gold standard data and full dictionary data was carried out. We classify the confidence levels of machine into 7 different categories. These categories are shown in Table 13 with their correspondence confidence levels.

Category	Description	Confidence
A	Both languages give above sim_threshold, and both languages give the same first match	High
B	(Language1 above sim_threshold, and Language2 gives lower similarity score) or (Language2 above sim_threshold, and Language1 gives lower similarity score)	High
C	Headword present in only one dictionary, and absent in the other	High
D	Both languages give below sim_threshold, and both languages give the same first match	Low
E	(Language1 below sim_threshold, but better than Language2) or (Language2 below sim_threshold, but better than Language1)	Low
F	Headword present in both dictionaries, but all entries of dictionary1 have already been assigned to other entries of dictionary2 or vice versa. Hence, there is no mapping	High
G	Force mapped, as this is the only remaining match	High

Table 13: Categorization of various mappings along with their confidence level

Analysis of gold standard data with these codes yielded the following results (See Table 14).

It is worth noting that the machine generated a total of 2096 mappings. The gold standard data has 2022 mappings. It is because the machine does not know what are the number of mappings present in the gold standard data. Word senses may have one-one, one-many and many-one mappings. Therefore, it is not possible to determine in advance how many word sense

Category	Pairs in the category	Percentage
A	636	30.34 %
B	221	10.54 %
C	738	35.21 %
D	60	02.86 %
E	59	02.81 %
F	360	17.18 %
G	22	01.05 %
Total	2096	100 %

Table 14: Categorization of gold standard mapping generated via algorithm CR6

mappings are to be generated. Therefore, the machine generated a total of 2096 mappings. Among these, the entries falling in the category of D and E have low confidence. Thus roughly 5-6% cases are such which are of low quality and would improve with human intervention. Rest of 94-95% cases can be mechanically aligned, saving precious resources. As the gold standard data was corrected as and when some parsing error or typographic error was seen, the error rate in gold standard data is much less. Whereas, there was no attempt made to clear these kind of errors in full dictionaries. Therefore, the error rates in the full dictionaries is more than the gold standard data.

The following are the results of application of the above methodology to full dictionaries Apte Sanskrit–Hindi (ASH) and Apte Sanskrit–English (AP90) (See Table 15).

Category	Pairs in given category	Percentage
A	51964	32.59 %
B	16655	10.44 %
C	47131	29.56 %
D	9730	06.10 %
E	6258	03.92 %
F	25710	16.12 %
G	2023	01.27 %
Total	159471	100%

Table 15: Categorization of word sense mappings generated by algorithm CR6 for Apte Sanskrit-Hindi and Apte Sanskrit-English dictionaries

Roughly 10% of cases fall under low confidence zone, which may require human intervention. Three random numbers were selected and 100 entries starting therefrom were examined for false positives. The following is the result. (See Table 16)

A	B	C	D	E	F	G	False Positives/total pairs
01	14	00	01	10	00	04	30/300

Table 16: False positives from randomly selected mappings

Thus, manual examination also yields around 10% error rate.

5 Mapping other dictionaries

Similar exercise was also tried for different dictionary pairs like (1) Apte Sanskrit-English and Monnier Williams Sanskrit-English dictionary[11] (2) Wilson Sanskrit-English and Yates Sanskrit-

[11]https://www.sanskrit-lexicon.uni-koeln.de/scans/MWScan/2020/web/webtc/download.html

English dictionary and (3) English WordNet and Webster's English dictionary. The results are shown in Table 17.

Category	Apte – MW	Wilson – Yates	WordNet – Webster
A	17.84 %	53.22 %	09.43 %
B	04.53 %	11.40 %	03.90 %
C	53.12 %	13.44 %	61.83 %
D	08.39 %	07.99 %	06.92 %
E	02.78 %	04.80 %	05.01 %
F	12.70 %	08.03 %	11.68 %
G	00.64 %	01.13 %	01.23 %

Table 17: Categorization of mappings for various dictionary pairs

Thus, in almost all dictionary pairs studied, the error rate (D+E) is roughly to the tune of 11-13%. These are the places where human annotators can make maximum impact by manual examination and correction.

6 Way ahead

We are exploring the possibility of using graph based similarity scores or semantic measures such as LSA or ESA to find out similarity in cases where text based similarity scores are below threshold. These approaches are computationally heavy and may require more computational resources. In the present case, the thresholds of similarity scores were chosen empirically or rather arbitrarily. It may be possible to learn these thresholds by optimizing its F-scores. As the gold standard (training data) is quite small, this exercise is not yet tried. Once we have large manually validated data, it will be worthwhile to find out the optimum thresholds with statistical methods.

The present methodology can be expanded to other language pairs and check whether findings in different language pairs are similar or otherwise. Effect of quality of translation services like Google Translate between different language pairs may add a cascading effect on the performance.

7 Conclusion

Undertaking the task of mapping of dictionaries at the level of word sense seems daunting at first, but after experimenting with a few dictionary pairs, it was only 11-13% of word senses that required manual examination by human expert. Once a quick implementation having an accuracy of 87-90% is created by machine, human annotators / users can be given an option to change the mapping if they feel that the mapping generated by the machine is incorrect. It holds immense potential to expand sense-mapped text resources from one language to another. It will particularly help the users of languages which are having scarce resources e.g. a Sanskrit work which has been disambiguated and sense-mapped in English with help of Sanskrit-English dictionary can be extended to the users of, say, French language by mapping Sanskrit-English dictionary to Sanskrit-French dictionary at word sense level.

References

Eneko Agirre and Philip Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

Vāmana Śivarāma Apte, Paraśurāma Kriṣṇa Gode, Cintāmaṇa Gaṇeśa Karve, and Kaśināṭha Vāsudeva Abhyankara. 1957. *Revised and Enlarged edition of Prin. V. S. Apte's The Practical Sanskrit-English Dictionary*. Prasad Prakashan, Poona.

Vaman Shivram Apte. 1890. *The Practical Sanskrit-English Dictionary, containing Appendices on Sanskrit Prosody and important Literary & Geographical names in the ancient history of India*. Shiralkar & Co. Book-sellers, Budhwar Peth, Poona.

Vaman Shivram Apte. 2007. *Sanskrit-Hindi Kośa*. Motilal Banarsidass, Delhi.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Eduard Bejcek, Václava Kettnerová, and Markéta Lopatková. 2014. Automatic mapping lexical resources: A lexical unit as the keystone. In *LREC*, pages 2826–2832.

Pushpak Bhattacharyya. 2010. Indowordnet. lexical resources engineering conference 2010 (lrec 2010). *Malta, May*.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. Dkpro similarity: An open source framework for text similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126.

CDSD. 2023. *Cologne Digital Sanskrit Dictionaries*. Cologne University, Cologne. version 2.4.123, accessed on 20 September 2023, at https://www.sanskrit-lexicon.uni-koeln.de.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In Martin Kay and Christian Boitet, editors, *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India, December. The COLING 2012 Organizing Committee.

Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

Gérard Huet. 2019. Sanskrit lexicography, past and future. In Li Wei, editor, *Research on the Language and Script in Buddhist Sutras*. Hangzhou Buddhist Academy.

Matthew A Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Richard Johansson and Luis Nieto Pina. 2015. Embedding a semantic network in a word space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1428–1433.

Salil Joshi, Arindam Chatterjee, Arun Karthikeyan Karra, and Pushpak Bhattacharyya. 2012. Eating your own cooking: automatically linking WordNet synsets of two languages. In *Proceedings of COLING 2012: Demonstration Papers*, pages 239–246.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3.4:235–244.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Alvaro Monge and Elkan Charles. 1997. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proc. of the ACM-SIGMOD Workshop on Research Issues on Knowledge Discovery and Data Mining*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.

Ana Salgado, Sina Ahmadi, Alberto Simoes, John McCrae, and Rute Costa. 2020. Challenges of word sense alignment. In *Proceedings of the LREC 2020 7th Workshop on Linked Data in Linguistics (LDL-2020)*, pages 45–51. European Language Resources Association (ELRA).

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31(1).

Claus Vogel. 2015. *Indian Lexicography*. Motilal Banarsidass, Delhi.

Piek Vossen. 1998. A multilingual database with lexical semantic networks. *Dordrecht: Kluwer Academic Publishers. doi*, 10:978–94.

Noah Webster. 1900. *Webster's unabridged dictionary of the English language*. Kikwansha.

H. H. Wilson. 1832. *A Dictionary in Sanscrit and English; Translated, Amended, and Enlarged from an Original Compilation, Prepared by Learned Natives for The College of Fort William*. Parbury, Allen & Co., London, second edition.

William E. Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. *The Educational Resource Information Center (ERIC)*.

Michael J Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education*, pages 130–134.

W. Yates. 1846. *A Dictionary in Sanscrit and English, Designed for the Use of Private Students and of Indian Colleges and Schools*. Baptist Mission Press, Calcutta.

Context and WSD: Analysing Google Translate's Sanskrit to English Output of Bhagavadgītā Verses for Word Meaning

Anagha Pradeep
Language Technology Research Centre
International Institute of Information Technology Hyderabad, India
anagha.pradeep@research.iiit.ac.in

Radhika Mamidi
Language Technology Research Centre
International Institute of Information Technology Hyderabad, India
radhika.mamidi@iiit.ac.in

Pavankumar Satuluri
Department of Humanities and Social Sciences
Indian Institute of Technology
Roorkee, India
pavankumar.satuluri@hs.iitr.ac.in

Abstract

In addition to innate human intelligence, having access to extensive context and world knowledge is a crucial factor that aids in comprehending natural language, making it smooth and effortless to understand words with multiple meanings for humans. Although machines lack intrinsic intelligence, their capacity to learn language can greatly improve with access to more data, which serves as valuable context. In Natural Language Processing (NLP), the task of identifying and attributing the right sense of a word in a given context is called Word Sense Disambiguation (WSD). WSD, as a sub-task, plays a crucial role in several NLP applications such as Machine Translation. Every language has a set of words that have multiple senses. Sanskrit, one of the ancient and classical languages of the Indian subcontinent is no exception to this. Like many other languages with a rich literary tradition, Sanskrit features a multitude of polysemous words. However, it is essential to acknowledge that the data used to train machine models on Sanskrit is considerably less compared to European and a few other Indian languages. Consequently, the task of disambiguating word senses in Sanskrit presents a highly complex challenge for machines, especially when considering the unique and rich nature of its literary language. The purpose of this paper is to delineate the potential areas where the infusion of additional data can enhance language learning, through a manual error analysis taxonomy focused on the Bhagavadgītā. Our analysis will delve into the translation outcomes produced by Google Translate, which is considered the state-of-the-art tool for handling Sanskrit and other languages with limited available resources.

1 Introduction

Language is the foremost factor that sets apart humans from other beings. Humans possess the innate ability to understand, analyze and express thoughts through languages. This also means that the process of disambiguating a word having multiple senses is fundamentally natural and effortless in humans, specifically considering the access humans have to context and world knowledge. An exponential growth in the domain of Artificial Intelligence (AI) with an aim of having machines that can simulate human behavior in analyzing and interpreting natural languages is evidently witnessed. Natural Language Processing (NLP) applications like Machine Translation (MT) deal with language complexities among which subtasks such as Word Sense Disambiguation (WSD) are also dealt with. WSD has been a long standing problem in the domain of Computational linguistics and NLP. It is necessary to determine the meaning of each

word in a context, in order to make sense of the text (Itankar and Raza 2020). Nonetheless, it must be acknowledged that this pursuit is far from a straightforward undertaking for a machine. Language modeling has been very helpful in advancing machines in language intelligence. Pre-trained Language Models (PLMs) such as Transformers use large scale unlabelled corpora for their training and have shown great progress in various NLP tasks (Zhao et al. 2023). In 2022, Google Translate, which uses the Transformer architecture, expanded its language repertoire to include Sanskrit and 23 other languages bringing the total number of translatable languages to 133[1]. Similar to numerous other languages, Sanskrit, the ancient classical language of the Indian subcontinent, includes a wide range of words with multiple senses. The unique nature of the language within the realm of literature adds an additional layer of complexity in disambiguating the word senses. Google enlists languages that have been trained using monolingual data and zero-resource MT as "long-tail languages" among which Sanskrit is one. These are the languages that do not have as much corpora for training the language model as do most European languages and a few Indian languages such as Hindi and Tamil (Bapna et al. 2022). This paper attempts to outline the importance of increased context in the form of training data for Sanskrit, through a manual error analysis of word meanings in the translation output of Bhagavadgītā (BhG). It proposes a taxonomy for word meaning errors and further explores the potential areas where an increase in contextual knowledge could bring about an enhancement in ambiguity resolution.

The paper is structured as follows: We will begin by discussing our rationale for selecting the BhG for translation in the subsequent section. This will be succeeded by a concise review of pertinent prior research. Moving on to the fourth section, we will provide a detailed account of the experiment and its subsequent analysis. Finally, in the fifth section, we will offer insights into additional observations gleaned from extensive experiments.

2 Motivation to choose Bhagavadgītā

BhG is a part of the well-known Indian epic, the Mahabharata. It can be found in the Bhīṣma-parva which is the sixth book of the epic. This text containing seven hundred verses, is a philosophical masterpiece that presents a conversation between Lord Kṛṣṇa and Arjuna, the Pāṇḍava prince on the battlefield of Kurukṣetra. On seeing his own relatives and loved ones on the opposing side of the battle, Arjuna is faced with a moral conflict about participating in the battle. In this moment of crisis, Lord Kṛṣṇa imparts spiritual wisdom to address Arjuna's concerns and inspires him to fulfill his duty (Mukundananda 2022). Numerous thinkers and scholars, spanning from Adi Shankaracharya (H. K. Goyandka 2015) to modern figures like Alladi Mahadeva Sastry (Sastry 2004), Mahatma Gandhi (Gandhi 2014), Swamy Dayananda Saraswati (Saraswati 2007), Eknath Easwaran (Easwaran 2009), and many more, have found profound fascination with this revered philosophical masterpiece, each offering their own unique interpretations. The impetus for this study arises from our curiosity to witness how a machine would undertake the translation of such a profound text. It is fascinating to observe where the machine excels and where it encounters challenges in the process.

3 Related Work

In 2022, Google Translate expanded its language repertoire to include Sanskrit. During the same year, a paper was published, focusing on the semantic and sentiment analysis of BhG translations in English using a language framework based on BERT (Chandra and Kulkarni 2022). This study employed three distinct translations by experts to develop a framework for analyzing the semantic and sentiment aspects of selected verses from the BhG. Subsequently, another paper evaluating the performance of Google Translate in translating the BhG into English was published in 2023. The output from Google Translate was compared to expert translations using sentiment and semantic analysis via BERT-based language models, serving as a continuation of the previous research (Shukla et al. 2023). As far as our knowledge extends, these studies

[1]https://blog.google/products/translate/24-new-languages/accessed on August 5th 2023

represent the primary investigations into the evaluation of machine translation outputs of the BhG. It is also important to note that the error analysis of semantic relations mentioned in the papers above involves an automated process of measuring semantic similarity between Google's English translation output and expert human English translations. And is therefore crucial to recognize that this approach significantly differs from the methodology employed in the present study.

In addition to these studies, (Popović 2020) proposes a new method of manual evaluation for Machine Translation output. They adhere to an issue marking strategy rather than a scoring or classifying one. The paper studies Croatian and Serbian outputs of IMDb (Internet Movie Database) movie reviews and Amazon product reviews from three different online MT systems viz: Google Translate[2], Amazon Translate[3] and Bing[4]. It concludes that out of the three MT systems Google generates the most comprehensible translations for both target languages. Usage of MT systems for literary works is not very common yet. This is possibly because of the unique structure of the language that literature carries. The usage of figurative devices, metaphors, idioms, irony and so on, makes it hard for machines to translate, often leading to literal translations that are incorrect. Nevertheless, there have been some studies in this area. (Omar and Gomaa 2020) evaluate the usefulness of applying machine translation systems to literature with a view of identifying challenges that may have negative impacts on the reliability of machine translation systems. The study uses two MT systems Google Translate and Q Translate[5] to translate two short stories Harry Potter and The Black Cat from English to Arabic. Although they conclude on how both MT systems have performed badly at various levels, we can observe that the error rate in Google Translate was comparatively lesser. In general, the evaluation of MT systems whether automatic or manual is done on the entire translation covering adequacy, comprehensibility, and grammaticality (Popović 2020). However, our work focuses only on errors in word meanings and offers a taxonomy of these errors.

4 Experiment

4.1 Data Collection

The Sanskrit verses of the BhG utilized in this study as the source language were obtained from a Sanskrit computational toolkit named Samsaadhanii[6]. For the translation aspect, the translation outputs were self-acquired through the Google Translate API[7], a widely employed tool for automated translation. This combination of original Sanskrit verses and translated content forms the foundation of our dataset for analysis[8].

4.2 Analysis

Words are the fundamental building blocks of any text, and their accurate interpretation plays a key role in understanding the text as a whole as well as preserving its essence. Therefore, the aim of this study is to observe errors in word meanings and categorize them. To achieve this a comprehensive manual examination of each word in the Google translation output of BhG was conducted to ascertain its alignment with the intended meaning of the corresponding Sanskrit word. Two reference translations of the BhG namely Jayadayal Goyandaka's "Bhagavad Gīta Tattvavivecanī" (J. Goyandka 2011) and Swami Mukundananda's "Bhagavad Gita The Song of God" (Mukundananda 2022) were employed for this process. Additionally, support was derived from the Monier Williams Sanskrit English dictionary (Monier-Williams 1899) to augment the accuracy of the analysis. Given that the scope of this taxonomy-based analysis was specifically

[2] https://translate.google.co.in/

[3] https://ai-service-demos.go-aws.com/translate

[4] https://www.bing.com/translator

[5] https://qtranslate.en.softonic.com/?ex=CS-1680.3

[6] https://sanskrit.uohyd.ac.in/Corpus/

[7] https://cloud.google.com/translate

[8] Translations of all the BhG verses were obtained through Google Translate API on 31st January 2023

confined to word meanings, the errors identified were categorized into four distinct groups, which are:

1. Errors arising from polysemous words.
2. Errors stemming from compounding.
3. Errors originating from words that are knowledge sensitive.
4. Errors springing from incorrect meaning attribution.

We will go through each of these categories of errors in detail with examples for a clearer understanding.

Errors arising from polysemous words:

Words that have multiple senses are known as polysemous words (Agirre and Edmonds 2007). Given such a word, context plays a crucial role in disambiguating which sense of the word is to be taken. For instance, the word हरि(*hari*) in Sanskrit has fourteen meanings such as snake, lion, Vishnu (lord), sun, moon, air, Yama (god of death), Indra (lord of deities), rays, horse, parrot, monkey, frog and the colour yellow[9]. Given the word हरि(*hari*), the machine should be able to aptly choose the right sense of the word based on its context among various possible interpretations. This task is known as Word Sense Disambiguation (Navigli 2009). The outputs received on Google Translate for the following inputs are given below[10].

Sanskrit Sentence	**English Translation**
हरिः खादति	**Hari eats**
हरिः कदलीफलं खादति	**Hari eats** bananas
हरिः वृक्षेऽस्मिन् कदलीफलं खादति	The **monkey** is **eating** bananas on this tree

Table 1: Sanskrit Sentences and English Translations

The outputs clearly indicate that when provided with additional contextual information, the machine's ability to disambiguate the sense of the polysemous word "हरि"(*hari*) becomes more refined. This enhanced disambiguation capability demonstrates the importance of context in NLP tasks. Nonetheless, within the translation outputs we generated, it is important to note that 139 out of the 700 verses contained errors attributable to the incorrect selection of senses for polysemous words. We further categorized this list of verses into two. The former comprised verses in which the error-identified polysemous word's meaning could potentially be adjusted by altering the context. In contrast, the latter encompassed verses in which the meaning of polysemous words remained unchanged regardless of contextual modifications.
An example for the first category is:

अपि चेदसि पापेभ्यः सर्वेभ्यः पापकृत्तमः। सर्वं ज्ञानप्लवेनैव वृजिनं सन्तरिष्यसि ।।4.36।।

Translation: Even if you are the most sinful of all sinners You will cross all troubles by the float of knowledge. 4.36

In this verse, the word प्लव(*plava*) has multiple senses such as float, frog, monkey, boat, sheep, enemy and so on[11].
The sense to be taken here is "boat". However, the machine also recognizes the sense "boat" given an alteration in the context.
रामः प्लवेन नदीं तरति

[9] `https://sanskrit.uohyd.ac.in/scl/amarakosha/frame.html`
[10] Translations for all the sample sentences were obtained between 15th August and 8th September 2023
[11] The dictionary entries for the various words discussed throughout this paper are given in Appendix A

Translation: Rama crosses the river by boat.

Similarly, another example for the first category is a very famous verse from the Gītā.

यदा यदा हि धर्मस्य ग्लानिर्भवति भारत। अभ्युत्थानमधर्मस्य तदात्मानं सृजाम्यहम् ॥4.7॥

Translation: Whenever there is a loss of religion, O Bhārata, When irreligion arises, I create Myself. 4.7

The word धर्म(*dharma*) has been wrongly identified with the sense religion. The appropriate sense to be used in this context is "righteousness".
Google Translate can also identify other senses of this word given different contexts such as:
उष्णत्वम् अग्नेः धर्मः
Translation: Heat is the righteousness of fire.
कुम्भकारस्य धर्मः कुम्भकरणम्
Translation: The duty of the potter is to make pottery.

Let us now go through examples for the second category.

किं कर्म किमकर्मेति कवयोऽप्यत्र मोहिताः। तत्ते कर्म प्रवक्ष्यामि यज्ज्ञात्वा मोक्ष्यसेऽशुभात् ।।4.16।।

Translation: Even the poets are confused here as to what is action and what is inaction I will tell you that action which, knowing it, you will be freed from evil. 4.16

In the above verse, the word कवि(*kavi*) is polysemous. It has several meanings including a wise person, a poet, names of several deities, and so on. The sense that fits well in this context is "a wise person or a thinker". Although the context was modified, the machine did not recognize the other senses associated with this word.
कविः सर्वदा सन्मार्गे एव चलति
Translation: The poet is always on the right path.
साध्वसाधुविवेचने कविः निपुणः
Translation: The poet is adept at distinguishing between right and wrong.

The same is the case of the word दैव(*daiva*). Despite the multiple senses such as celestial, divine, royal, destiny, and others that the word carries, the translation output irrespective of the context has only been "destiny".

दैवमेवापरे यज्ञं योगिनः पर्युपासते। ब्रह्माग्नावपरे यज्ञं यज्ञेनैवोपजुह्वति ।।4.25।।

Translation: Other yogis worship destiny as the sacrifice Others offer the sacrifice in the Brahma-agni by the sacrifice itself. 4.25

निर्मलरात्रौ वयं आकाशे ताराचन्द्रादिकं दैववस्तु द्रष्टुं शक्नुमः
Translation: On a clear night we can see the stars and moon and other objects of destiny in the sky.
दैववृक्षेति सुप्रसिद्धः कल्पवृक्षः प्रार्थितं सर्वं दास्यति
Translation: The Kalpa tree, well known as the tree of destiny, will give everything asked for.
In the three example cases that we have seen, the senses of the polysemous word दैव(*daiva*) to be taken in line with the context are celestial and or divine.

Upon examining examples for both scenarios, we observed that in the first case, the machine was able to discern the various senses of polysemous words, only when the context was modified. However, in the second case, even with altered context, the sense remained consistent. As per our observation, we hypothesize that both these cases lack sufficient examples in the training data, that cover a wide range of senses for each of the polysemous words. Thus, inducing additional data through which the machine can learn the distinct senses as well as diverse contexts would be of paramount importance to enhance the performance of the system.

Errors stemming from compounding:

In classical Sanskrit, compounding is a prevalent linguistic feature where multiple words are merged to create a single lexeme. The purpose of this practice is to achieve brevity in language, aiming to express the intended meaning using fewer words. Nevertheless, grasping the meaning of compounds can be intricate due to the fact that they often possess multiple potential interpretations, contingent upon contextual cues (Krishna et al. 2016). In our examination of the translation output of the BhG, we have identified instances where meaning errors arose either from an incorrect interpretation of the compound or from the absence of an interpretation altogether. It is important to note that these errors were relatively infrequent. In the examples that follow, we will delve into instances of both types of errors.

अदृष्टपूर्वं हृषितोऽस्मि दृष्ट्वा भयेन च प्रव्यथितं मनो मे। तदेव मे दर्शय देव रूपं प्रसीद देवेश जगन्निवास ॥11.45॥

Translation: I was delighted to see something I had never seen before and my mind was overwhelmed with fear Show me that very form, O Lord of the gods, O inhabitant of the universe. Have mercy on me. 11.45

The compound जगन्निवास (*jagannivāsa*) has been translated as "inhabitant of the universe", which is indeed one of the possible interpretations. However, in the given context where Arjuna is addressing Lord Krishna and praising him, a more suitable interpretation would be "the one in whom the world resides" (जगतः निवासः). Referring to Lord Krishna as the one in whom the world resides often carries a deeper and more profound connotation, emphasizing his divine glory, whereas the former interpretation may not capture this essence as effectively. It is important to note that various interpretations of the compound are dependent on the primary derivative suffix.

In the verse,

आब्रह्मभुवनाल्लोकाः पुनरावर्तिनोऽर्जुन। मामुपेत्य तु कौन्तेय पुनर्जन्म न विद्यते ॥ 8.16॥

Translation: O Arjuna, the worlds return to the Abrahma world. But having attained Me, O son of Kunti, there is no rebirth. 8.16

we see that the compound आब्रह्मभुवनात् (*ābrahmabhuvanāt*) has been translated as Abrahma world, while the compound is to be interpreted as "upto the world of Brahma". This shows that the machine has not been able to interpret the compound.

Upon observation, we noticed that the occurrence of errors related to compounding in the verses was relatively infrequent. This is likely because the training data treated compounds as single words and therefore identified and translated them accurately. To substantiate our observation, we quote the following examples:

Sanskrit Sentence	**English Translation**
अहं **रामालयं** गच्छामि	I am going to **Ramalaya**
अहं **विद्यालयं** गच्छामि	I am going to **school**
गणेशः **लम्बोदरः** अस्ति	Ganesha is **tall**
भीमः **वृकोदरः** अस्ति	Bhima is a **wolf-belly**

Table 2: Sanskrit Sentences and English Translations

Albeit, even in cases where the machine is trained on various interpretations of a compound, determining which interpretation is appropriate for a specific context can still be a challenging task.

Errors originating from words that are knowledge sensitive:

From our observation errors stemming from words that are sensitive to cultural, societal, or philosophical nuances are quite common in machine-generated content. Machines often struggle to recognize the senses of such words, particularly when those words hinge on a deep grasp of human culture, history, or philosophy. These nuances can be challenging for machines to capture accurately. BhG being a profound philosophical text, that is a part of ऐतिहासिक (*aitihāsika*) text is susceptible to such errors. Here are a couple of instances that we may look through.

पाञ्चजन्यं हृषीकेशो देवदत्तं धनञ्जयः। पौण्ड्रं दध्मौ महाशङ्खं भीमकर्मा वृकोदरः ।।1.15।।

Translation: Hrishikesha married Panchajanya and Arjuna married Devadatta Bhimakarma blew the great conch of Paundra 1.15

From the translation, it is apparent that the words पाञ्चजन्य, देवदत्त, पौण्ड्र (*pāñcajanya, devadatta, pauṇḍra*) have been incorrectly recognized as names of entities and place, instead of being correctly identified as the names of the conches used by Kṛṣṇa, Arjuna, and Bhīma. This misinterpretation occurs due to the machine's limitation in historical and contextual knowledge. Another observation regarding the translation is that the verse does not contain a verb related to "marriage", yet the translation includes this verb twice. This discrepancy may be attributed to instances of similar sentences in the training data.

Similarly, in the verse:

प्रकृतिं पुरुषं चैव क्षेत्रं क्षेत्रज्ञमेव च। एतद्वेदितुमिच्छामि ज्ञानं ज्ञेयं च केशव ॥13.1॥

Translation: Nature, the person, the field, and the knower of the field. I wish to know this knowledge and the knowable, O Kesava. 13.1

Although the words have been translated to their corresponding senses, the philosophical nuances they carry often remain unrepresented in these. Take, for example, the word प्रकृति (*prakṛti*); it does not refer to nature in a general sense but is intricately linked to the specific concept of material nature discussed in the Sāṅkhya philosophy. The machine frequently struggles to convey these intricacies due to its inherent limitations.

Errors springing from incorrect meaning attribution:

Out of the 700 verses in the BhG, it was found that 25 verses had word translations that were completely unrelated to their actual word meanings. Unlike polysemous words where at least one of the listed senses was considered, in these instances, the translations provided did not align with any of the listed meanings for the respective words. Let us take a look at a few sample cases.

अयनेषु च सर्वेषु यथाभागमवस्थिताः। भीष्ममेवाभिरक्षन्तु भवन्तः सर्व एव हि ।।1.11।।

Translation: Situated in their proper places in all the moons May you all protect Bhishma alone. 1.11

The word अयन (*ayana*) refers to path in this context. It is wrongly translated as "moon", which does not correspond to any of the listed senses of the word.

Similarly, the word परित्राण (*paritrāṇa*), which means "to protect", was incorrectly translated in the following verse.

परित्राणाय साधूनां विनाशाय च दुष्कृताम्। धर्मसंस्थापनार्थाय सम्भवामि युगे युगे ।।4.8।।

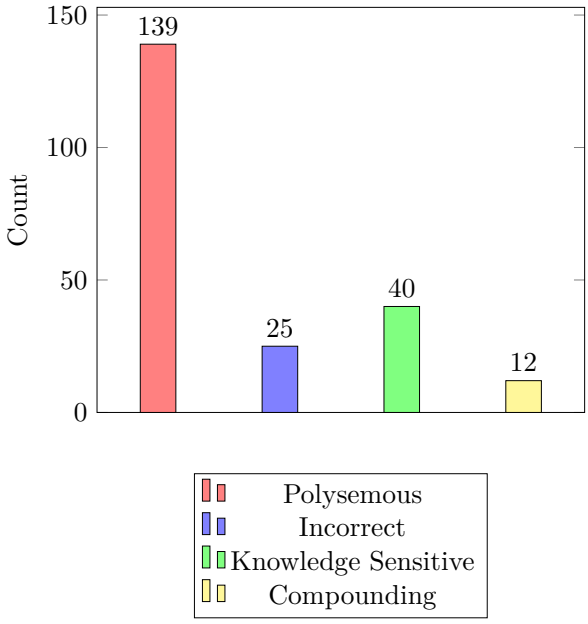


Figure 1: Bar chart depicting the number of verses in each category

Translation: For the salvation of the righteous and the destruction of the wicked I am able to establish righteousness in every age. 4.8

The sense "salvation" has not been enlisted as one of the senses of the word परित्राण (*paritrāṇa*). We could not draw any conclusions regarding the reasons for these incorrect translations, except for the possibility that the training data might have contained errors.

5 Additional Experiments

In addition to the previously mentioned error categories, we also identified a few other factors that influenced the translation of word meanings. These factors are listed below, along with examples.

5.1 Sandhi

Sandhi encompasses sound or form alterations occurring either at morpheme or word boundaries (Hyman 2007). In Table 3, we have listed a few examples illustrating how the presence or absence of sandhi influenced the translation of word meanings.

Sanskrit Sentence/Verse	**English Translation**
हरिः **वृक्षेऽस्मिन्** कदलीफलं खादति	The **monkey** is eating bananas on this tree
हरिः **वृक्षे अस्मिन्** कदलीफलं खादति	**Hari** is eating bananas on this tree
न च मां तानि कर्माणि निबध्नन्ति धनञ्जय। **उदासीनवदासीनमसक्तं** तेषु कर्मसु ॥9.9॥	Nor do those actions bind Me, O Arjuna **Indifferent as if indifferent**, unattached to those actions.
न च मां तानि कर्माणि निबध्नन्ति धनञ्जय। **उदासीनवत् आसीनम्** असक्तं तेषु कर्मसु ॥9.9॥	Nor do those actions bind Me, O Arjuna **Seated as if indifferent**, unattached to those actions.
चातुर्वर्ण्यं मया सृष्टं गुणकर्मविभागशः। तस्य कर्तारमपि मां **विद्ध्यकर्तारमव्ययम्**॥4.13॥	I have created the four varnas according to the divisions of virtue and action Know Me, **the inexhaustible doer, to be the doer** of it
चातुर्वर्ण्यं मया सृष्टं गुणकर्मविभागशः। तस्य कर्तारम् अपि मां **विद्धि अकर्तारम् अव्ययम्**॥4.13॥	I have created the four varnas according to the divisions of virtue and action Know Me also as **the doer of it, the inexhaustible non-doer**.

Table 3: Sanskrit Sentences and English Translations

5.2 Playing with words

In some cases, we also observed how the absence and replacement of words in a sentence has rendered changes in the meanings of the polysemous words in the sentence. Table 4 lists instances of a few examples.

ये यथा मां प्रपद्यन्ते तांस्तथैव भजाम्यहम्। मम वर्त्मानुवर्तन्ते मनुष्याः पार्थ सर्वशः ।।4.11।।

Translation: I reward those who worship Me in the same way that they worship Me. Men follow My path in every way, O Arjuna.

In the case of the word प्रपद्यन्ते (*prapadyante*), we needed its specific sense of "surrender" in the given context. To explore if we could achieve the desired meaning through some modifications, we initially transformed the metrical form into a sentence: ये यथा मां प्रपद्यन्ते तान् तथा अहं भजामि इति भगवान् कृष्णः भगवद्गीतायाम् अवदत्. This sentence was translated as "Lord Krishna said in the Bhagavad Gita that I worship those who worship Me in the same way as they worship Me".
To experiment, we decided to remove certain words from the sentence to see if we could obtain the intended sense. Surprisingly, when we removed the word अहम् (*aham*), the desired output was achieved: भगवान् कृष्णः भगवद्गीतायां ये यथा मां प्रपद्यन्ते तान् तथा भजामि इति अवदत्.
Translation: Lord Krishna said in the Bhagavad Gita that I worship those who surrender to Me in the same way.
However, it is to be noted that when the verse was transformed into prose, the meaning of the word भजामि (*bhajāmi*) was also inadvertently altered.

Although this was not an example from the Gītā, we encountered something similar with the word हरि (*hari*) as well.

Sanskrit Sentence	English Translation
हरिः वृक्षेऽस्मिन् स्थित्त्वा खादति	The monkey is standing on this tree and eating
हरिः वृक्षेऽस्मिन् स्थित्त्वा पश्यति	The monkey stands in this tree and watches
हरिः वृक्षेऽस्मिन् उपविश्य खादति	The monkey is sitting in this tree and eating
हरिः वृक्षेऽस्मिन् उपविश्य पश्यति	**Hari** is sitting in this tree and watching
हरिः वृक्षेऽस्मिन् स्थित्त्वा पश्यति	The monkey stands in this tree and watches
हरिः वृक्षेऽस्मिन् गत्त्वा खादति	The monkey goes to this tree and eats
हरिः वृक्षेऽस्मिन् गत्त्वा पश्यति	The monkey goes to this tree and looks

Table 4: Sanskrit Sentences and English Translations

It was interesting to observe how making seemingly random changes to words could bring out sense modifications. Nonetheless, despite our efforts, we could not discern a logical pattern or inference point to explain why these alterations were occurring.

6 Conclusion and Future Work

The objective of this study was to underscore the significance of context, in the form of extensive data, in improving the performance of language models for translating word meanings in Sanskrit. To achieve this goal, we assessed Google Translate's output of BhG from Sanskrit to English and developed a taxonomy of errors. Notably, our taxonomy revealed that the most frequent errors arose from polysemous words, highlighting the critical role of WSD in NLP tasks like machine translation.

However, we also observed a significant limitation: the state-of-the-art PLMs lack access to substantial Sanskrit-language data, unlike many European languages and a few other Indian languages. To address this, we illustrated how more context could potentially reduce errors stemming from polysemous words.

In addition to polysemy, we identified three other error categories: compounding, knowledge-sensitive words, and incorrect meaning attribution. While increased contextual learning might

mitigate these errors to some extent, they remain complex due to the inherent challenge of instilling machines with world knowledge.

Our study further delved into the functioning of Google Translate through experiments. However, we were unable to derive logical inferences regarding why and how the mere presence or absence of certain factors influenced the machine's translations of word meanings.

In conclusion, this study provides a broad exploration based on our observations and experiments with Google Translate. With the disclosure of the data used to train language models, future research can yield more specific linguistic-based solutions to enhance performance in Sanskrit translation tasks.

Acknowledgement

I would like to thank Prof. Amba Kulkarni for the Sanskrit verses of the Bhagavadgītā (BhG) utilized in this study and Mr. Sriram Krishnan for his valuable input and guidance.

References

Agirre, Eneko and Philip Edmonds (2007). *Word sense disambiguation: Algorithms and applications*. Vol. 33. Springer Science & Business Media.

Bapna, Ankur et al. (2022). "Building machine translation systems for the next thousand languages". In: *arXiv preprint arXiv:2205.03983*.

Chandra, Rohitash and Venkatesh Kulkarni (2022). "Semantic and sentiment analysis of selected Bhagavad Gita translations using BERT-based language framework". In: *IEEE Access*.

Easwaran, Eknath (2009). *The Bhagavad Gita*. Jaico Publishing House.

Gandhi, Mahatma (2014). *The Bhagavad Gita*. Jaico Publishing House.

Goyandka, Hari Krishnadas (2015). *Srimad Bhagavad Gita with Shankara Bhashya*. Gita Press Gorakhpur.

Goyandka, Jayadayal (2011). *Śrīmadbhagavadgītā Tattvavivecanī (English Commentary)*. Gita Press, Gorakhpur, India.

Hyman, Malcolm D (2007). "From pāṇinian sandhi to finite state calculus". In: *International Sanskrit Computational Linguistics Symposium*. Springer, pp. 253–265.

Itankar, Prashant Y and Nikhat Raza (2020). "Ambiguity Resolution: An Analytical Study". In.

Krishna, Amrith et al. (2016). "Compound type identification in sanskrit: what roles do the corpus and grammar play?" In: *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pp. 1–10.

Monier-Williams (1899). *A Sanskrit-English Dictionary: Etymologically and Philologically Arranged with Special Reference to Cognate Indo-European Languages*. Motilal Banarsidass.

Mukundananda, Swami (2022). *BHAGAVAD GITA: THE SONG OF GOD*. Rupa & Company.

Navigli, Roberto (2009). "Word sense disambiguation: A survey". In: *ACM computing surveys (CSUR)* 41, pp. 1–69.

Omar, Abdulfattah and Yasser Gomaa (2020). "The machine translation of literature: Implications for translation pedagogy". In: *International Journal of Emerging Technologies in Learning (iJET)* 15, pp. 228–235.

Popović, Maja (2020). "Informative manual evaluation of machine translation output". In.

Saraswati, Swamy Dayananda (2007). *Śrīmad Bhagavad Gītā*. Arsha Vidya Research and Publication Trust.

Sastry, Alladi Mallesh (2004). *The Bhagavad Gita*. Samata Books.

Shukla, Akshat et al. (2023). "An evaluation of Google Translate for Sanskrit to English translation via sentiment and semantic analysis". In: *arXiv preprint arXiv:2303.07201*.

Zhao, Wayne Xin et al. (2023). "A survey of large language models". In: *arXiv preprint arXiv:2303.18223*.

Appendix

The appendix is structured as follows: In Sec. A we present details about the dictionary entry[12] for words.

A Dictionary

1. **plava** (H1) [Printed book page 715,1] plava a plavaka &c. See col. 2. [ID=141442] (H2) [Printed book page 715,2] plava b mf(ā)n. swimming, floating, ŚāṅkhGṛ. ; Suśr. [ID=141469] sloping towards, inclined, Hariv. ; Var. ; Hcat. (in astrol. applied to a constellation situated in the quarter ruled by its planetary regent, Var. , Sch.) [ID=141470] transient, MuṇḍUp. [ID=141471] plava mn. (ifc. f(ā).) a float, raft, boat, small ship, RV. &c. &c. [ID=141472] plava m. a kind of aquatic bird (= gātra-samplava, kāraṇḍava, jala-vāyasa, jala-kāka or jala-kukkuṭa, L.), VS. &c. &c. [ID=141473] a frog, L. [ID=141474] a monkey, L. [ID=141475] & sheep, L. an arm, L. [ID=141476] a Caṇḍāla, L. [ID=141477] an enemy, L. [ID=141478] Ficus Infectoria, L. [ID=141479] a snare or basket of wicker-work for catching fish, L. [ID=141480] the 35th (or 9th) year in a cycle of Jupiter, VarBṛS. [ID=141481] plava m. swimming, bathing (ifc. f(ā).), MBh. ; R. ; Kathās. [ID=141482] plava m. flooding, a flood, the swelling of a river, MBh. ; MārkP. [ID=141483] the prolated utterance of a vowel (= pluti), L. [ID=141484] protraction of a sentence through 3 or more Ślokas (= kulaka), L. [ID=141485] sloping down or towards, proclivity, inclination, L. [ID=141486] (in astrol.) = plava-tva, VarBṛS. , Sch. [ID=141487] a kind of metre, Col. [ID=141488] N. of a Sāman (also with vasiṣṭhasya), ĀrṣBr. [ID=141489] jumping, leaping, plunging, going by leaps or plunges, R. (cf. comp. below) [ID=141490] returning, L. [ID=141491] urging on L. [ID=141492] plava n. Cyperus Rotundus or a species of fragrant grass, Suśr. [ID=141493]

2. **dharma** (H2) [Printed book page 510,3] 1. dharma m. (rarely n. g. ardharcādi; the older form of the RV. is dharman, q.v.) that which is established or firm, steadfast decree, statute, ordinance, law [ID=99903] usage, practice, customary observance or prescribed conduct, duty [ID=99904] right, justice (often as a synonym of punishment) [ID=99905] virtue, morality, religion, religious merit, good works (dharmeṇa ind. or °māt ind. according to right or rule, rightly, justly, according to the nature of anything; cf. below; °mesthita mfn. holding to the law, doing one's duty), AV. &c. &c. [ID=99906] Law or Justice personified (as Indra, ŚBr. &c.; as Yama, MBh. ; as born from the right breast of Yama and father of Śama, Kāma and Harṣa, ib. ; as Viṣṇu, Hariv. ; as Prajā-pati and son-in-law of Dakṣa, Hariv. ; Mn. &c.; as one of the attendants of the Sun, L. ; as a Bull, Mn. viii, 16 ; as a Dove, Kathās. vii, 89 , &c.) [ID=99907] the law or doctrine of Buddhism (as distinguished from the saṅgha or monastic order, MWB. 70) [ID=99908] the ethical precepts of Buddhism (or the principal dharma called sūtra, as distinguished from the abhi-dharma or 'further dharma' and from the vinaya or 'discipline', these three constituting the canon of Southern B°, MWB. 61) [ID=99909] the law of Northern B° (in 9 canonical scriptures, viz. Prajñā-pāramitā, Gaṇḍa-vyūha, Daśa-bhūmīśvara, Samādhirāja, Laṅkāvatāra, Saddharma-puṇḍarīka, Tathā-gata-guhyaka, Lalita-vistara, Suvarṇa-prabhāsa,ib. 69) [ID=99910] nature, character, peculiar condition or essential quality, property, mark, peculiarity (= sva-bhāva, L. ; cf. daśa-dh°arma-gata, ŚBr. &c. &c.; upamānopameyayor dh°, the tertium comparationis, Pāṇ. ii, 1, 55, Sch.) [ID=99911] a partic. ceremony, MBh. xiv, 2623 [ID=99912] sacrifice, L. [ID=99913] the ninth mansion, Var. [ID=99914] an Upaniṣad, L. [ID=99915] associating with the virtuous, L. [ID=99916] religious abstraction, devotion, L. [ID=99917] = upamā, L. (cf. above) [ID=99918] a bow, Dharmaś. [ID=99919] [Printed book page 1329,2] a thing, Sukh. i [ID=99919.1] [Printed book page 510,3] a Soma-drinker, L. [ID=99920] N. of the 15th Arhat of the present Ava-sarpiṇī, L. [ID=99921] of a son of Anu and father

[12] https://www.sanskrit-lexicon.uni-koeln.de/scans/MWScan/2020/web/webtc/indexcaller.php

of Ghṛta, Hariv. [ID=99922] of a s° of Gāndhāra and f° of Dhṛta, Pur. [ID=99923] of a s° of Haihaya and f° of Netra, BhP. [ID=99924] of a s° of Pṛthu-śravas and of Uśanas, ib. [ID=99925] of a s° of Su-vrata, VP. (cf. dharma-sūtra) [ID=99926] of a s° of Dīrgha-tapas, VāyuP. [ID=99927] of a king of Kaśmīra, Rāj. iv, 678 [ID=99928] of another man, ib. vii, 85 [ID=99929] of a lexicographer &c. (also -paṇḍita, -bhaṭṭa and -śāstrin), Cat. [ID=99930] dharma [cf. Lat. firmus, Lith. dermé.] [ID=99930.05] (H2C) [Printed book page 510,3] dharmeṇa ind., according to right or rule, rightly, justly, according to the nature of anything [ID=99930.1] dharmāt ind., according to right or rule, rightly, justly, according to the nature of anything [ID=99930.2] (H2) [Printed book page 512,3] 2. dharma Nom. P. °mati, to become, law, Vop. [ID=100471] 3. dharma in comp. for °man, q.v. [ID=100472] (H1) [Printed book page 513,1] dharma a See p. 510, col. 3. [ID=100518]

3. **kavi** (H1) [Printed book page 264,2] kavi mfn. (√1. kū cf. 2. kava, ākūta, ākūti, kāvya, Naigh. iii, 15 ; Nir. xii, 13 ; Uṇ. iv, 138) gifted with insight, intelligent, knowing, enlightened, wise, sensible, prudent, skilful, cunning [ID=46509] kavi (is), m. a thinker, intelligent man, man of understanding, leader [ID=46510] a wise man, sage, seer, prophet [ID=46511] a singer, bard, poet (but in this sense without any technical application in the Veda), RV. ; VS. ; TS. ; AV. ; ŚBr. i, 4, 2, 8 ; KaṭhUp. iii, 14 ; MBh. ; Bhag. ; BhāgP. ; Mn. vii, 49 ; R. ; Ragh. [ID=46512] N. of several gods, (esp.) of Agni, RV. ii, 23, 1; x, 5, 3; iii, 5, 1; i, 31, 2; 76, 5 [ID=46513] of Varuṇa, Indra, the Aśvins, Maruts, Ādityas [ID=46514] of the Soma [ID=46515] of the Soma priest and other sacrificers [ID=46516] (probably) N. of a particular poet [ID=46517] cf. aṅgiras (Mn. ii, 151) and uśanas (Bhag. x, 37) [ID=46518] of the ancient sages or patriarchs (as spirits now surrounding the sun) [ID=46519] of the Ṛbhus (as skilful in contrivance) [ID=46520] of Pūṣan (as leader or guider) [ID=46521] N. of a son of Brahmā, MBh. xiii, 4123, 4142-4150 [ID=46522] of Brahmā, W. [ID=46523] of a son of Bhṛgu and father of Śukra, MBh. i, 2606 (cf. 3204; BhāgP. iv, 1, 45 and Kull. on Mn. iii, 198) [ID=46524] that of Śukra (regent of the planet Venus and preceptor of the demons), Rājat. iv, 495 [ID=46525] of the planet Venus, NBD. [ID=46526] of the sons of several Manus, Hariv. ; BhāgP. ; VP. [ID=46527] of a son of Kauśika and pupil of Garga, Hariv. [ID=46528] of a son of Ṛṣabha, BhāgP. [ID=46529] of Vālmīki, L. [ID=46530] a keeper or herd, RV. vii, 18, 8 [ID=46531] (fig.) N. of the gates of the sacrificial enclosure, TS. v, 11, 1, 2 (cf. kavaṣ) [ID=46532] the sun, W. [ID=46533] of various men [ID=46534] the soul in the Sāṃkhya philosophy Comm. [ID=46535] a cunning fighter, L. [ID=46536] an owl, L. [ID=46537] kavi (is or ī, W.), f. the bit of a bridle, L. [ID=46538] the reins (cf. kavikā), W. [ID=46539] a ladle (cf. kambi), L. [ID=46540]

4. **daiva** (H1) [Printed book page 497,2] 1. daiva or daiva mf(ī)n. (fr. deva) belonging to or coming from the gods, divine, celestial, AV. ; Br. ; Mn. ; MBh. &c. [ID=96788] sacred to the gods (-tīrtha n. the tips of the fingers, Mn. ii, 59 ; cf. s.v.; °vīdik f. the north, L. ; cf. 2. diś) [ID=96788.05] royal (vāc), Rājat. v, 205 [ID=96788.1] depending on fate, fatal, Kāv. [ID=96788.15] daiva m. (with or without vivāha) a form of marriage, the gift of a daughter at a sacrifice to the officiating priest, Mn. iii, 21 ; 28 [ID=96788.2] the knowledge of portents, Śaṃk. [ID=96788.25] patr. of Atharvan, ŚBr. [ID=96788.3] pl. the attendants of a deity, TāṇḍBr. xvii, 1, 1 [ID=96788.35] (H1B) [Printed book page 497,2] daivī (ī), f. a woman married according to the Daiva rite, Viṣṇ. xxiv, 30 [ID=96788.4] a division of medicine, the medical use of charms, prayers &c., W. [ID=96788.45] (H1B) [Printed book page 497,2] daiva n. a deity (cf. kula-), BhP. iii, 1, 35 &c. [ID=96788.5] (scil. karman, kārya &c.) a religious offering or rite, Yājñ. ; MBh. [ID=96788.55] daiva n. divine power or will, destiny, fate, chance (°vāt ind. by chance, accidentally), AV. ; Mn. ; MBh. &c. [ID=96788.6] (H1C) [Printed book page 497,2] daivāt (°vāt), ind., by chance, accidentally [ID=96788.7] (H2) [Printed book page 497,3] 2. daiva Vṛddhi form of deva in comp. [ID=96855]

5. **ayana** (H2) [Printed book page 84,2] ayana a mfn. going, VS. xxii, 7 ; Nir. [ID=14674] ayana n. walking, a road, a path, RV. iii, 33, 7 &c. (often ifc. cf. naimiṣāyana, puruṣāyaṇa, praśamāyana, samudrāyaṇa, svedāyana), (in astron.) advancing, precession, Sūryas. [ID=14675] (with gen. [e.g. angirasām, ādityānām, gavām, &c.] or ifc.) 'course, circulation', N. of various periodical sacrificial rites, AV. ; ŚBr. &c. the sun's road north and south of the equator, the half year, Mn. &c., the equinoctial and solstitial points, Var-BṛS. &c. [ID=14676] way, progress, manner, ŚBr. [ID=14677] place of refuge, Mn. i, 10 [ID=14678] a treatise (śāstra cf. jyotiṣām-ayana), L. [ID=14679] (H1) [Printed book page 84,3] ayana b See √ay, col. 2. [ID=14744.1]

6. **paritrāṇa** (H3) [Printed book page 595,3] pari-trāṇa n. rescue, preservation, deliverance from (abl.), protection or means of protection, refuge, retreat, Mn. ; MBh. &c. [ID=117701] self-defence, L. [ID=117702] the hair of the body, L. [ID=117703] moustaches, Gal. [ID=117704]

7. **prapad** (H1) [Printed book page 682,1] 1. pra-√2. pad Ā. -padyate (ep. also P.), to fall or drop down from (abl.), throw one's self down (at a person's feet), MBh. ; to go forwards set out for, resort to, arrive at, attain, enter (with acc., rarely loc.), AV. &c. &c.; to fly to for succour, take refuge with (acc.), TS. &c. &c.; to fall upon, attack, assail, RV. ; AV. ; to come to a partic. state or condition, incur, undergo (acc.), MBh. ; Kāv. &c.; (with an adv. in sāt), to become, e.g. sarpasāt pra-√pad, to bec° a serpent, Bhaṭṭ. ; to obtain, gain (patini, 'as husband'), partake of, share in (acc.), ib. ; to adopt or embrace (a doctrine), Rājat. ; to undertake, commence, begin, do, MBh. ; Kāv. ; to form (a judgement), MBh. ; to assume (a form), Kathās. ; to enjoy (pleasure), R. ; to take to (dat.), Hariv. ; to come on, approach, appear, AV. ; R. ; Hariv. ; to take effect, succeed, MBh. ; to turn out (anyathā, 'differently' i.e. without any effect or consequence), Hariv. ; to admit (a claim), R. : Caus. -pādayati, °te, to cause to enter, introduce into (acc. or loc.), Br. : Desid. P. pitsati, to wish to enter, ŚBr. ; Ā. -pitsate (cf. Pāṇ. vii, 4, 54), to be going to incur or undertake, Daś. [ID=135115] (H2) [Printed book page 682,1] 2. pra-pad f. away, AitBr. [ID=135118] N. of partic. sacred texts, Br. ; GṛŚrS. [ID=135119] (H1) [Printed book page 682,2] 3. pra-pad f. (fr. 3. pad) the fore part of the foot, AV. [ID=135140]

Linguistically Mapping Aśoka: A Dialectometric Approach to the Major Rock and Major Pillar Edicts

Patrick Zeitlhuber
University of Vienna
patrick.zeitlhuber@gmail.com

Abstract

The edicts of the emperor Aśoka were inscribed in different Middle Indo-Aryan language varieties as well as Greek and Aramaic in the 3rd century BCE. These Middle Indo-Aryan varieties have been variously categorized into three or four dialect groups. In this paper, these classifications are reassessed by applying methods of dialectometry. Dialectometry is a branch of quantitative linguistics which aims at measuring the differences between languages and language varieties. I will examine Aśoka's Major Rock and Major Pillar Edicts by calculating the Levenshtein distance and aggregating the results. This is followed by hierarchical clustering and multidimensional scaling in order to determine the most suitable grouping of these language varieties. After triangulating the results, the dialect classification will be projected onto a geographical map, therefore showing the clear regional distribution of these dialect groups.

Keywords: Aśoka, dialectometry, quantitative linguistics, Middle Indo-Aryan, inscriptions

1 Introduction

The edicts of the emperor Aśoka constitute the earliest extant evidence of written culture in South Asia. They were issued in the years after Aśoka's coronation, which is commonly dated to 268 BCE. Strikingly, these edicts were not inscribed in Sanskrit, but in different language varieties of Middle Indo-Aryan (MIA) as well as Greek and Aramaic and served diverse purposes and functions.

Figure 1 shows a map of the 42 edict sites,[1] which extend over the territory of the four modern states of Pakistan, India, Nepal, and Afghanistan.[2] Of these, the inscriptions in Afghanistan as well as in Taxilā in Pakistan were written in either Greek, Aramaic or both. 174 edicts were composed in various MIA language varieties. Commonly, these inscriptions are divided into Major Rock Edicts, Minor Rock Edicts, Major Pillar Edicts, Minor Pillar Edicts, Cave Sites, and various edicts (comprising the Pāṅgurāria Separate Pillar Edict and the Bhābrū Stone Inscription).

The MIA varieties in the inscriptions show clear dialectal differences. That becomes even more obvious as the edicts are often transfers of a certain text from one variety into another. The original language is certainly an administrative language from the Eastern part of Aśoka's realm (2003, 165-166). Schneider (1978) even attempted to reconstruct the original archetype of the Major Rock Edicts in this administrative language.

[1] A list of the abbreviations for all the Aśokan edict sites is provided in appendix A.

[2] The maps in this paper were created with `QGIS`, which is free and open-source. The coordinates for the Aśokan sites were taken from Falk (2006; 2013) and verified and amended, if necessary, with Google Maps. For better geographical referencing, river courses have been marked, which, however, represent the modern conditions.

At least to my knowledge, the MIA language of the Aśokan edicts has no commonly excepted denomination apart from "Aśokan inscriptional language" and similar ones. In this paper, I opt to call them—in accordance with the nomenclature of most other MIA languages—*Āśokī*.

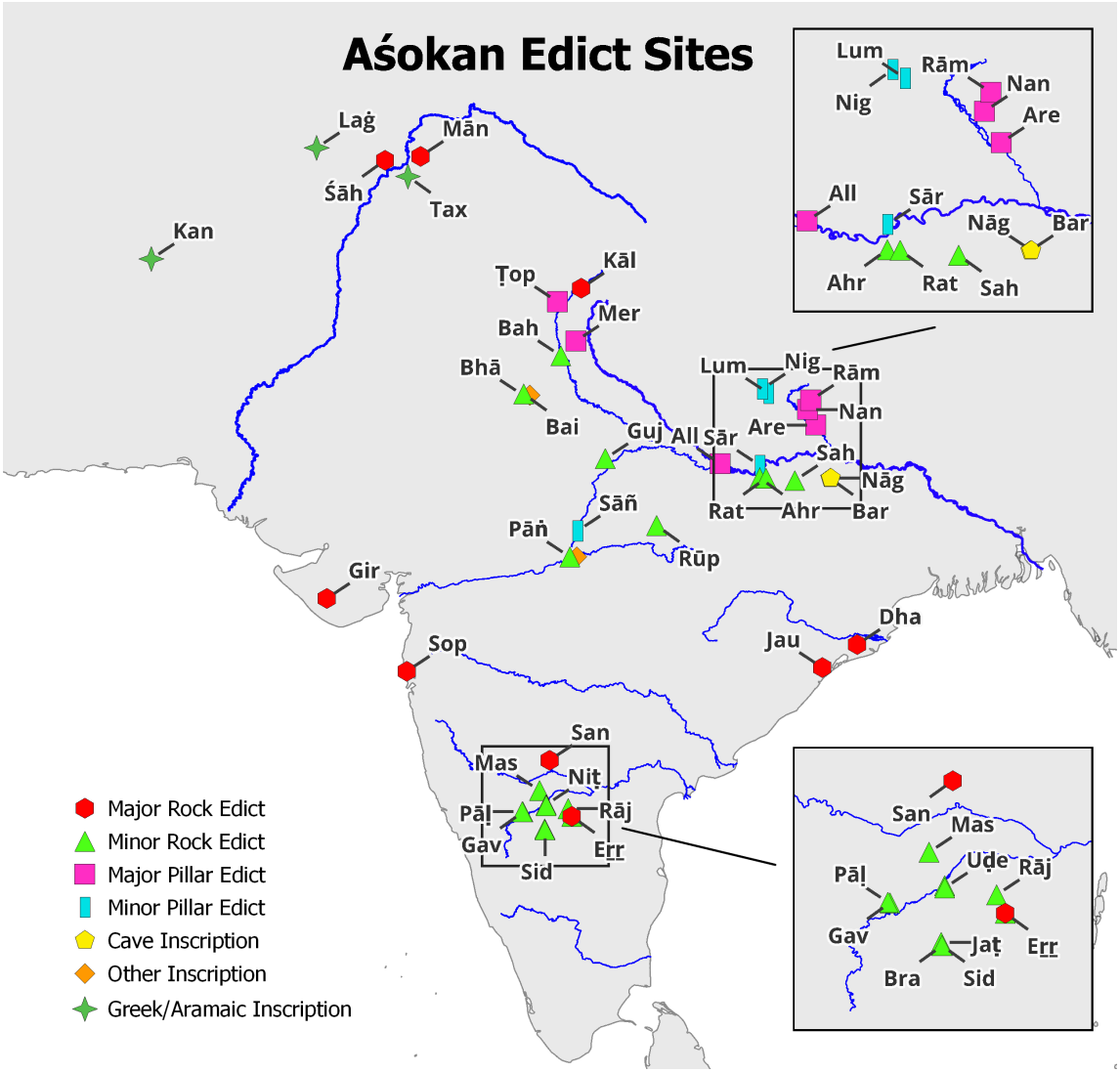


Figure 1: map of the Aśokan edicts

The classifications of the Āśokī varieties in the literature are based on qualitative linguistic analysis and vary between three and four dialect groups. Researchers picked certain linguistic characteristics which they deemed representative and distinctive, on which grounds they grouped these language varieties together. The following tables provide a sample of these different classifications.

Table 1 shows the dialect assessment by Salomon (1998, 73-76) and Oberlies (2003, 165). Both describe three dialects with the same members. The Northwestern group is constituted by Śāh and Mān, the Western by Gir and Sop, and the Eastern by Kāl, Dha, Jau, and E̲r̲r̲ (and for Oberlies also together with all the other Āśokan inscriptions).

	Northwestern	**Western**	**Eastern**
Salomon (1998)	Śāh, Mān	Gir, Sop	Kāl, Dha, Jau, E<u>rr</u>
Oberlies (2003)	Śāh, Mān	Gir, Sop	Kāl, Dha, Jau, E<u>rr</u> "all other rock edicts, pillar edicts" (p. 165)

Table 1: classifications into 3 dialects

Four dialects with slightly different members are postulated by Sen (1960, 7–11), Misra & Misra (1982, 9-10), and Bubeník (1996, 8), which can be seen in table 2. The names of the dialects are slightly different with each of these researchers. The terms in the first row before the comma are used by Sen and Misra & Misra, after the comma by Bubeník. All of them agree that the Northwestern (North-West) group is formed by Śāh and Mān and the Southwestern (West) by Gir. They disagree, however, on the exact classification of the other inscriptions. Unfortunately, Sop and E<u>rr</u>, which form a separate dialect for Bubeník, are not mentioned by Sen and Misra & Misra. Bubeník groups those two together, whereas they are clearly separated by Salomon and Oberlies. Otherwise, Sen and Misra & Misra separate Kāl, Dha, and Jau, which Bubeník considers members of the same dialect.

	Northwestern, North-West	**Southwestern, West**	**Middle Eastern, South/South-West**	**Eastern, Center/East**
Sen (1960)	Śāh, Mān	Gir	Kāl, Ṭop, Nāg[3]	Dha, Jau "all the Minor Rock Edicts and Pillar Edicts, the Cave Inscriptions" (p. 11)[4]
Misra & Misra (1982)	Śāh, Mān	Gir	Kāl, Ṭop, Nāg[5]	Dha, Jau
Bubeník (1996)	Śāh, Mān	Gir	Sop, E<u>rr</u>	Kāl, Dha, Jau

Table 2: classifications into 4 dialects

In this paper, I will apply quantitative methods to reassess these dialect classifications. To be precise, I will draw upon the methods of dialectometry, which is a well-established methodology, first and foremost in Romance and German(ic) variationist linguistics. Dialectometry was devised in the 1970s and 80s out of a desire to reassess prevailing dialect classifications, which were often based on small sets of subjectively chosen linguistic features, neglecting the major part of the concerned varieties. Nerbonne (2009, 177) summarizes it by stating:

> By focusing exclusively on single features or small combinations of these, variationists, including dialectologists, sometimes fail to isolate signals of provenance clearly. The signals are often so complex, even misleading, that they resist analysis using simple, single-featured methodologies.

[3]Sen also mentions the Jogīmara Cave inscriptions here. Even though they are from the Mauryan period, they are not Aśokan (Salomon, 1998, 76).

[4]Sen also mentions the Mahāsthān stone plaque inscription, the Sohgaurā copper-plate inscription, and the Hāthīgumphā inscriptions of Khāravela here. The former two are from the Mauryan period, but they are not Aśokan. The latter are from the Śuṅga period even (Salomon, 1998, 76, 142).

[5]Misra & Misra follow Sen in listing the Jogīmara Cave inscriptions here.

The aim of dialectometry is to make language measurable. This is achieved by quantifying linguistic differences either between dialects and varieties of one language or between related languages. By and large, two major schools of thought can be differentiated: the "Salzburg school" established by Hans Goebl (2010) and the "Groningen school" centered around John Nerbonne (2010). The main differences regard certain epistemological and methodological approaches. Both schools work predominantly with data from linguistic atlases.

The Salzburg school takes data and taxates it according to linguistic phenomena on the levels of phonetics, morphology, syntax, and lexicon. The similarity between these taxates is calculated with different algorithms (Goebl, 2010). The Groningen school uses predominantly the Levenshtein distance either in its original version or with various modifications. A taxation is not necessary but the data need to be arranged so only appropriate linguistic items are compared (Nerbonne, 2010). Common to both schools is that the attained measurements are arranged in a distance (or similarity) matrix which is the basis for further analyses. Hierarchical clustering has proven to be a useful method for both. The results are then projected onto a geographical map. Proponents of the Salzburg school create maps by using different clustering methods, Euclidian proximity, skewness, arithmetic mean, standard deviation etc. (Scherrer and Stoeckle, 2016; Goebl, 2010). Apart from clustering, the Groningen school applies multidimensional scaling and bipartite spectral graph partitioning (Nerbonne, 2010; Heeringa, 2004; Wieling and Nerbonne, 2011).

Especially in German variationist linguistics, different dialectometric approaches have been developed which are not based on distance matrices, e.g. factor analysis and principal component analysis (Pickl and Pröll, 2019). Of course, the methods of dialectometry are not restricted to horizontal variation, i.e. language variation in space. It is also possible to measure differences and distances between vertical varieties like dialects, regiolects, and standard language (Kehrein, 2012). This is an all but exhaustive list of all the different approaches that have been applied in dialectometry.

2 From the Data to the Map

2.1 Data Preparation

In order to determine the number of Āśokī dialects, I chose dialectometry as a viable methodology. After the digitization of the relevant data, I arranged the texts of the Major Rock Edicts (MaRE) and Major Pillar Edicts (MaPE) in a data frame as correspondence sets so each row equals one location and each cell in a column contains all variants of a certain variable from that location (see table 3 as an example). As the name of the variable has no influence on the distance measurements, the Sanskrit equivalent of the MIA wordforms serve as a reference point. Multiple variants are indicated by using | as delimiter.

The MaREs of Sopārā and Sannati had to be excluded as these are only extant in fragments. For the remaining MaREs and the MaPEs, I selected all the wordforms that are attested in at least 75 % of these edict sites. This is less based on statistical reasons than on practicality. This approach led to 66 wordforms that were suitable for further comparison. The next step was the philological and linguistic interpretation of these tokens.

Apart from Śāh and Mān, which were inscribed in Kharoṣṭhī, all the edicts were written in Brāhmī. The Aśokan Brāhmī script indicates vowel length but not geminate consonants. Moreover, anusvāra is often omitted. Judging from the inscriptions, the law of two morae (von Hinüber, 2001, 117-118) had already had its effect on the Āśokī dialects. In contrast to Old Indo-Aryan (OIA), in MIA no long vowel could precede a geminate or a consonant cluster. OIA VCC resulted either in MIA VC or VCC.[6] Both possibilities are attested in different lexemes at different Aśokan sites. For the linguistic interpretation of the data, this means that whenever a word in the Brāhmī edicts was written with a short vowel followed by a single consonant sign and this particular form can be traced back to OIA VCC, it can safely be assumed that

[6] V = long vowel, V = short vowel, C = consonant

Skt	bhavati	rājā	kariṣyanti
All	-	lājā	kacchanti
Are	hoti	lāja|lājā	kacchanti
Dha	hoti	lājā|lāja	kacchanti
E<u>rr</u>	hoti|hoti	lāja|lājā	kacchanti
Gir	bhavati|hoti	rājā	kāsanti|kassanti
Jau	hoti	lājā	kacchanti
Kāl	hoti	rājā|lājā	kacchanti
Mān	hoti|bhoti	rājā	kaṣṣanti
Mer	hoti	lājā|lāja	-
Nan	hoti	lāja	kacchanti
Rām	hoti	lāja	kacchanti
Śāh	bhoti|hoti	rāyā|rājā	kaṣṣanti
Ṭop	hoti	lājā|lājā	kacchanti

Table 3: example correspondence set

that consonant has to be understood as a geminate. However, the opposite may also be the case when a consonant cluster following an etymologically short vowel got simplified and the vowel underwent compensatory lengthening, e.g. OIA *varṣeṣu* > *vāsesu* (Gir MaRE03), next to *vassesu* (Kāl MaRe03). Especially the variety of Gir was very prone to this kind of sound change.

Yet another difficulty presents itself with regard to the Kharoṣṭhī inscriptions of Śāh and Mān. Just like Brāhmī, geminates were not indicated and anusvāra often omitted (or sometimes added in unetymological positions). Apart from that, Aśokan Kharoṣṭhī does not designate the quality of vowels. When it comes to sound clusters like OIA ∀CC or ∀CC, they appear in Kharoṣṭhī as VC. It is, therefore, impossible to tell whether the vowel was shortened or the consonant degeminated. In agreement with the phonetic interpretations in the "Dictionary of Gāndhārī" on `gandhari.org` (Baums and Glass, 2002 ongoing), these cases were treated as retaining the etymological vowel length.

Another peculiarity worth mentioning are the inscriptions from Kāl. In these, the signs for *s*, *ṣ*, and *ś* are used without any clear distinction. Bubeník (1996, 9) claims, "The three sibiliants of OIA survive [...] to a certain degree in the Center (Ka)", i.e. Kāl. I tend to disagree with this statement. Sometimes the sibilant signs appear in etymological positions but in most cases there is no obvious reason. It is likely that the scribe considered these signs to be graphical variants and used them indiscriminately or according to taste to represent one and the same sibilant phoneme.

Further challenges for the linguistic interpretation concern scribal errors, orthographic peculiarities, and inconsistent spellings. Moreover, it is imperative that only cognates are compared with each other which will be elaborated on in the next section.

2.2 Distance Measurement

For the calculation of the linguistic distances between the language varieties of the Aśokan sites, the package `dialectR` for the software `R` (Shim and Nerbonne, 2022) was utilized. The function `distance_matrix` allows the creation of a distance matrix by applying the Levenshtein distance.

The Levenshtein distance (or: edit distance) measures the number of modifications that are necessary to transform one string into another by either insertion, deletion, or substitution (Kruskal, 1983, 215-219). It is the main method of measurement used by the Groningen school of dialectometry. Nerbonne (2010, 481) states that "[e]arlier work in dialectometry analyzed the data at a nominal level, where each pair of linguistic items was measured as the same or different, while the application of Levenshtein distance allows numeric characterizations per

pair of pronunciations to be obtained." Discussing the advantage of this kind of measurement, Heeringa (2004, 24) argues that "[t]he Levenshtein distance is completely objective, and its results are verifiable, an advantage it shares with other computational methods, in contrast to dialect maps based on tribes and intuition", if "the data used consists of representative samples of the varieties."

Another important aspect regarding distance measurements is the fact that only cognates in different varieties should be compared. It would be possible to use the Levenshtein distance to calculate the difference between two words that are etymologically unrelated. However, this would yield methodologically and epistemologically incorrect results. Therefore, it is a prerequisite that already the data preparation is carried out with sound philological and linguistic knowledge.

In order to illustrate a measurement with the Levenshtein distance, the variable *YATHĀ* will serve as an example in table 4:

Gir (MaRE12)	y	a	th	ā	
E̲r̲r̲ (MaRE12)		a	th	a	
	1			1	= 2

Table 4: Levenshtein distance example

Two modifications are necessary to get from *yathā* to *atha*: the deletion of word-final *y* and the substitution of *ā* by *a*. Hence, the absolute number of changes is 2. Yet, the parameters for the function `distance_matrix` can be set to normalize the length of strings so the penalty of the modification is calculated in relation to the total number of characters by setting `alignment_normalization = TRUE`. In the example above, this means these 2 modifications are divided by the sum of the string length of 4, which equals a relative difference of 0.5.[7]

Of course, these are still only two variants. For a useful distance measurement, a matrix needs to be calculated that compares all the variants of a certain variety with all the variants in every other variety for every variable. Herein lies the value of dialectometry as it is not based on certain single features but it combines all the distance values of all the variables in all the varieties. This step as called aggregation (Nerbonne, 2010).

	All	Are	Dha	E̲r̲r̲	Gir	Jau	Kāl	Mān	Mer	Nan	Rām	Śāh	Ṭop
All	0,0	2,8	3,6	3,9	10,4	3,3	6,1	10,9	1,6	2,9	3,1	13,9	2,5
Are	2,8	0,0	5,3	5,7	13,6	4,4	8,2	12,7	2,7	0,6	0,6	16,4	4,0
Dha	3,6	5,3	0,0	4,8	11,9	2,6	6,2	11,2	3,6	5,1	5,2	14,9	3,8
E̲r̲r̲	3,9	5,7	4,8	0,0	12,9	4,2	5,9	12,0	4,7	5,6	5,5	15,5	4,8
Gir	10,4	13,6	11,9	12,9	0,0	10,9	14,4	13,2	12,3	13,6	13,8	12,3	13,0
Jau	3,3	4,4	2,6	4,2	10,9	0,0	5,2	10,0	3,6	4,3	4,6	13,4	4,1
Kāl	6,1	8,2	6,2	5,9	14,4	5,2	0,0	12,1	5,3	8,0	8,6	15,7	6,2
Mān	10,9	12,7	11,2	12,0	13,2	10,0	12,1	0,0	9,8	12,3	12,7	6,8	12,5
Mer	1,6	2,7	3,6	4,7	12,3	3,6	5,3	9,8	0,0	2,4	2,6	12,8	2,4
Nan	2,9	0,6	5,1	5,6	13,6	4,3	8,0	12,3	2,4	0,0	0,5	16,4	3,7
Rām	3,1	0,6	5,2	5,5	13,8	4,6	8,6	12,7	2,6	0,5	0,0	16,6	4,1
Śāh	13,9	16,4	14,9	15,5	12,3	13,4	15,7	6,8	12,8	16,4	16,6	0,0	16,5
Ṭop	2,5	4,0	3,8	4,8	13,0	4,1	6,2	12,5	2,4	3,7	4,1	16,5	0,0

Table 5: distance matrix of 13 MaREs and MaPEs

[7]The dental voiceless aspirate is represented by the digraph *th* in this illustration but in the calculation it is considered one element to reflect its phonological status. As one of the anonymous reviewers pointed out, due to the use of IAST aspirates like *th* are treated by the Levenshtein algorithm as two string elements not reflecting the phonological status of aspirates. Even though the examples in this paper are presented in IAST, for the calculation I have resorted represent aspirates with capital letters and non-aspirates with lowercase letters.

In this manner, the Levenshtein distance was calculated for the Āśokī data set containing 66 tokens from 13 locations with MaREs and MaPEs.[8] As a result, a distance matrix is created by calculating the Levenshtein distance between all the variants at a certain location with all the variants at another location for every variable. Table 5 represents the resulting distance matrix but for a more concise display the values were rounded to one decimal.

It is rather difficult to make sense of a plain distance matrix. Hence, further processing of the distance measurements is necessary. Certain methods of illustration have proven useful, first and foremost creating dendrograms on the basis of hierarchical clustering (section 2.3) as well as multidimensional scaling (section 2.4). The results from both approaches can be used to create maps (section 2.5).

2.3 Hierarchical Clustering

To project the linguistic distances onto a map, it is necessary to reduce the distance matrix to a value matrix (Scherrer and Stoeckle, 2016, 101). One means to accomplish that is clustering, whereby one of the most frequently used approaches is agglomerative hierarchical clustering.

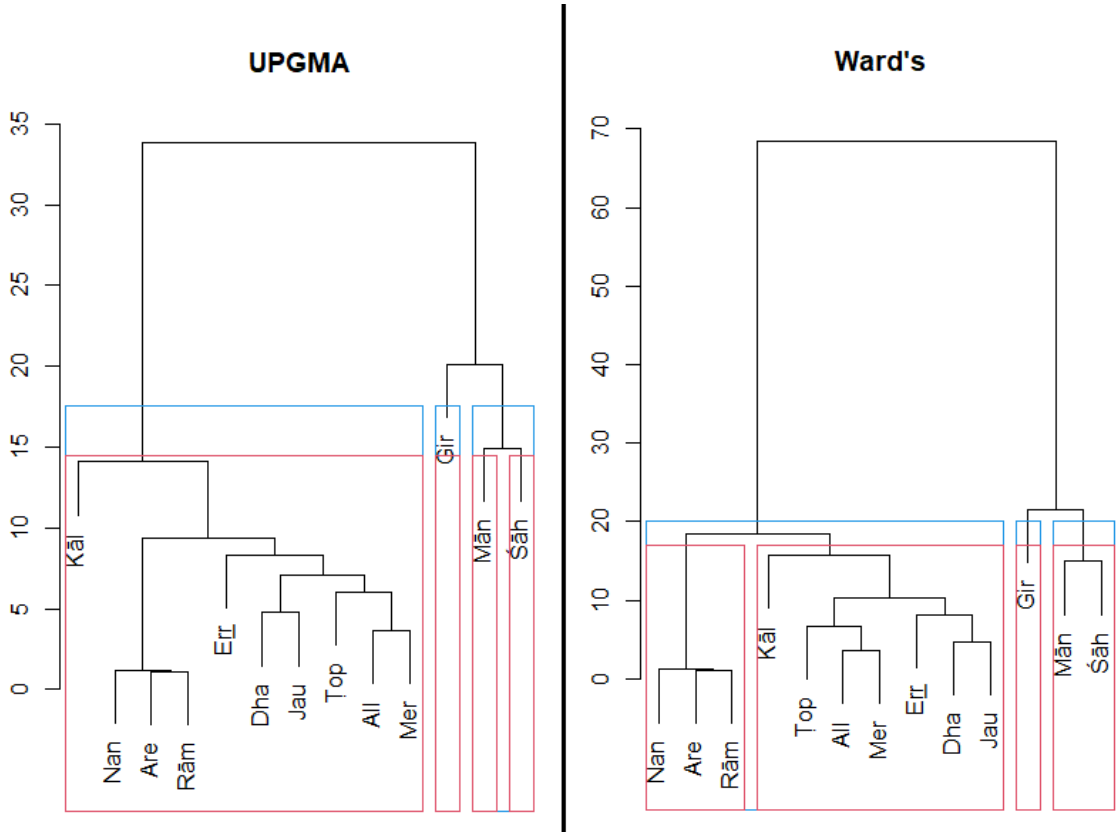


Figure 2: hierarchical clustering with 3 (blue) and 4 (red) clusters

`R` provides the built-in function `hclust` for that. For this paper, the agglomeration methods of UPGMA (= unweighted pair group method with arithmetic mean; called `"average"` in `R`) and Ward's minimum variance method (called `"ward.D2"` in `R`) were chosen in order to be able to compare the validity of the results. Most frequently, the results obtained by hierarchical clustering are plotted as a dendrogram.

From a linguistic point of view, a dendrogram allows for a grouping of dialects. The branches show which varieties are linguistically closer to each other. The distances between the branches

[8] The parameters in `R` were set like this: `distance_matrix(dataset, funname = "leven", alignment_normalization = TRUE, delim = "|")`.

give valuable clues about the most suitable dialect categorization.

Figure 2 shows a dendrogram based on UPGMA to the left and Ward's method to the right. What can be clearly seen from both plots is that there are two major linguistic clusters. One is constituted by Gir, Mān, and Śāh, the other by Kāl, Nan, Are, Rām, Ṭop, All, Mer, Err, Dha, and Jau. The blue lines indicate which locations are grouped together when the number of clusters is set to be three. Even then, the ten locations on the left from Kāl to Mer form one cluster, Gir alone a second, and Mān with Śāh a third. These clusterings hold true with UPGMA as well as Ward's.

Strikingly, the agglomeration into four clusters, illustrated by the red lines, yields different results depending on the selected method. With UPGMA, it does not lead to a subdivision of the location from Kāl to Mer. It rather assigns Gir, Mān, and Śāh to separate clusters each.

When selecting four clusters with Ward's method, however, Gir constitutes one of its own, while Mān and Śāh remain in one cluster together. Nan, Are, and Rām are grouped together and separated from another cluster comprising Kāl, Ṭop, All, Mer, Err, Dha, and Jau.

Consequently, the grouping of the language varieties of these locations into three clusters seems valid as both agglomeration methods agree in this respect. The subdivision into four clusters remains questionable, however. Shim & Nerbonne (2022, 23) state that hierarchical clustering methods are rather unstable and need to be validated with other methods, e.g. multidimensional scaling. In the following section, two different forms of multidimensional scaling will be applied.

2.4 Multidimensional Scaling

Nerbonne (2010, 487) describes multidimensional scaling (MDS) as "a statistical technique aimed at representing very high dimensional data in a smaller number of dimensions." This is accomplished by assigning the calculated distance values points in a coordinate system, usually either in two or three dimensions. These coordinates yield a plot that can be read as a linguistic map that depicts the linguistic distances each and every point has from the other (Embleton et al., 2013, 14). To put it simply, MDS is one form of graphical representation of a distance matrix.

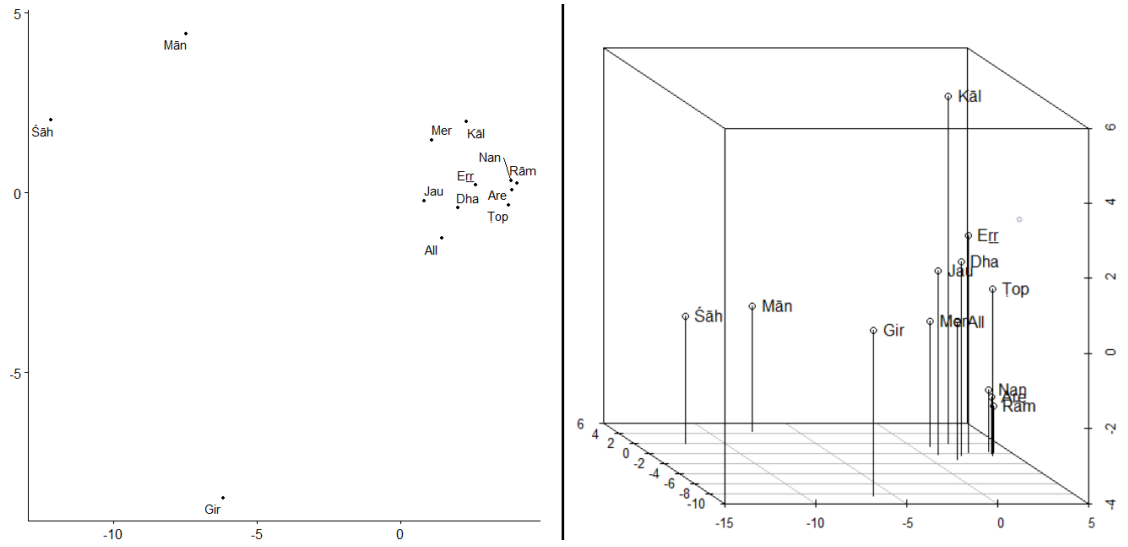


Figure 3: 2D and 3D multidimensional scaling

The left plot in Figure 3 shows a projection of MDS onto two-dimensional space. As with hierarchical clustering, Kāl, Nan, Are, Rām, Err, Dha, Jau, Ṭop, All, and Mer are very close to each other. Gir as very far down. Even though Śāh and Mān are nearer to each other than to any other variety, they show nevertheless some considerable linguistic differences.

The 3D illustration on the right side of Figure 3 depicts the same distances on the horizontal

axis but it gives more details about the distances between points that are very close to each other. This is in accordance with the dendrogram in Figure 2, in which Kāl is the highest point on this branch and Nan, Are, and Rām the lowest. The distances between the cluster to the right still remain the same to Gir, Śāh and Mān.

Compared with the dendrogram in Figure 2, it can be claimed with certainty that Kāl, Nan, Are, Rām, Err, Dha, Jau, Ṭop, All, and Mer form a cluster and, therefore, constitute one dialect group. Gir is set so far apart from any other language variety, hence, it must be assumed that it forms a dialect of its own.

Not as straightforward is the classification of Śāh and Mān. The notion of these two as individual dialect groups would be supported by hierarchical clustering with UPGMA, not by Ward's method though.

In both two- and three-dimensional scaling, Nan, Are, and Rām appear rather close to Kāl, Ṭop, All, Mer, Err, Dha, and Jau. Consequently, it does not seem advisable to divide these varieties into separate clusters as suggested by Ward's method in Figure 2.

2.5 Mapping Linguistic Distances

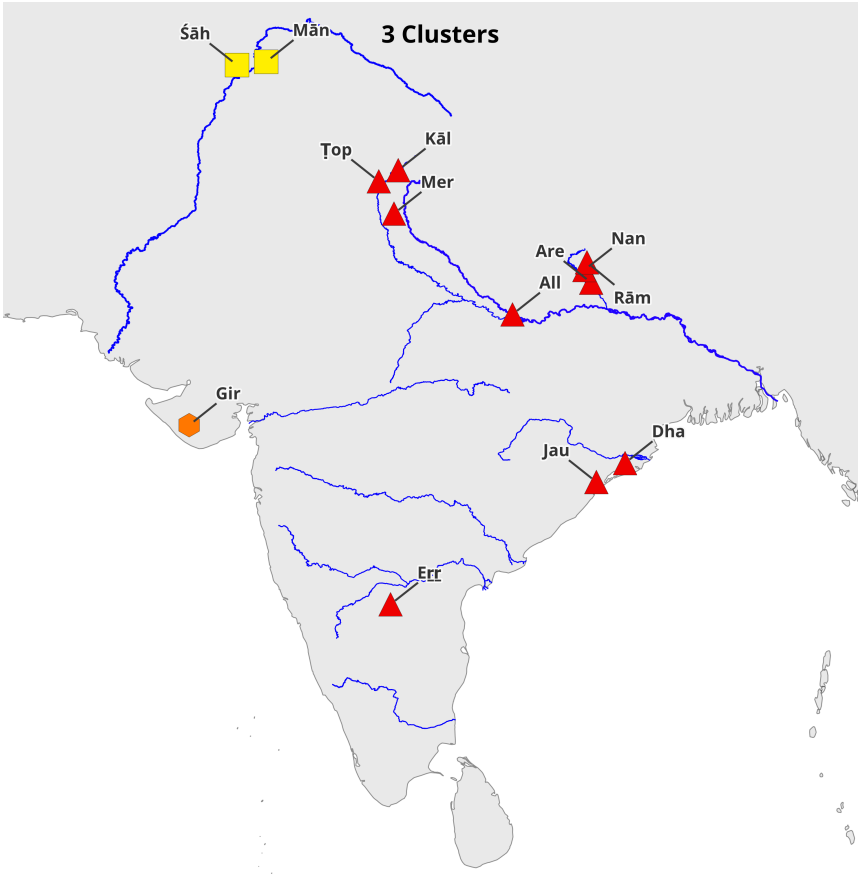


Figure 4: map with 3 clusters of Āśokī

Based on the results of hierarchical clustering and MDS, it is sensible to divide the language varieties of the Major Rock and Major Pillar Edicts into three clusters. These dialect groups can be projected onto a geographical map in order to illustrate the geographical dimensions. For these purposes, the coordinates of the Aśokan sites and the clusters obtained by the above-described methods were imported into `GQIS`, which allowed for the creation of this map of linguistic clusters (Figure 4).

Seeing the three dialect clusters in geographical space, it becomes clear that there is an obvious relation between geographical and linguistic distance. I will, consequently, follow Salomon and Oberlies in referring to the dialects according to their geographical provenance. Northwestern Āśokī (yellow squares) is constituted by Śāh and Mān although there is some variation between these two varieties. Gir is clearly set apart and is the only representative of Western Āśokī (orange hexagon). The varieties of Kāl, Nan, Are, Rām, Ṭop, All, Mer, E̲r̲r̲, Dha, and Jau form Eastern Āśokī (red triangles), which is the best and most widely attested dialect.

Circling back to Figure 3, the coordinates assigned by MDS to the Aśokan sites based on the distance matrix mirror the geographical distribution of the dialect groups as a whole on the map, but the distribution of individual varieties is different. The distances of the Northwestern, Western and Eastern dialect in the MDS plots more or less reflects their geographical distance. Even though this is mere coincidence, it is a striking one indeed.

3 Discussion

With regard to Table 1, the dialect classification of Salomon (1998) and Oberlies (2003) can be affirmed. The grouping in Table 2, however, does not seem to be valid compared with the dataset for which the measurements in this paper were made. Even with the two different options of four clusters in Figure 2, there is no reason for separating Kāl from Dha and Jau as Sen (1960) and Misra & Misra (1982) suggested, nor E̲r̲r̲ from Kāl, Dha and Jau as proposed by Bubeník (1996). There may be arguments for this division with a focus on single linguistic characteristics. Based on the dataset for this paper, there is no evidence for this differentiation from the point of view of dialectometric aggregation. It is possible, however, that these results might change with an expanded data set containing more wordforms and linguistic phenomena.

The wide prevalence of Eastern Āśokī creates an epistemological issue. As Salomon (1998, 75) pointed out:

> But it must also be understood that they [i.e. the Aśokan inscriptions] do not provide anything like a real dialect map of the time. For the geographical distribution of the dialects—especially of the eastern dialect—can hardly correspond with linguistic reality; the eastern dialect was obviously not the mother tongue of residents of the far north and the central south, though it was used for inscriptions (Kālsī, E̲r̲r̲aguḍi, etc.) in those regions.

Hence, I want to emphasize that the aim of this paper is not to present a dialect map of the 3rd century BCE. Figure 4 is supposed to be a map of a linguistic clustering of the language varieties used in the Aśokan inscriptions regardless of whether or not they are an authentic reproduction of speech habits of speakers of that time.

Still, this study has some limitations. The data set with 66 tokens is rather small. Due to the fact that the Levenshtein distance is applied to whole strings, it was necessary to include only those wordforms that are attested on all or most of the sites. To base the analyses on a broader linguistic foundation, it will be necessary to utilize some other kind of comparison—perhaps plain word stems (which comes with its own challenges).

Another option would be to chose an approach like Goebl (2010) and taxate the data according to linguistic phenomena. Methods not relying on distance matrices like the ones Pickl & Pröll (2019) use might be a viable endeavour. In the future, other methods of mapping will be explored like multidimensional scaling maps (Nerbonne, 2010). Furthermore, it is my desideratum to expand the dialectometric approach to include all the MIA Aśokan edict sites.

Acknowledgements

I want to thank Dr. Philipp Stöckle for providing his expertise and patiently answering all my methodological and technical questions. I am also grateful to Prof. Alexandra N. Lenz and the Austrian Academy of Sciences for giving me the time to work on this paper. And a big thank you goes to Assoc. Prof. Hannes A. Fellner for his on-going support.

References

Stefan Baums and Andrew Glass. 2002-ongoing. A dictionary of Gāndhārī (online). `https://gandhari.org/dictionary`.

Vít Bubeník. 1996. *The structure and development of Middle Indo-Aryan dialects*. Motilal Banarsidass, Delhi.

Sheila Embleton, Dorin Uritescu, and Eric S. Wheeler. 2013. Defining dialect regions with interpretations: Advancing the multidimensional scaling approach. *Literary and Linguistic Computing*, 28(1):13–22.

Harry Falk. 2006. *Aśokan sites and artefacts. A source-book with bibliography*. Number 18 in Monographien zur indischen Archäologie, Kunst und Philologie. Philipp von Zabern, Mainz am Rhein.

Harry Falk. 2013. Remarks on the Minor Rock Edict of Aśoka at Ratanpurwa. *Jñāna-Pravāha Research Journal*, 16:29–48.

Hans Goebl. 2010. Dialectometry and quantitative mapping. In Alfred Lameli, Roland Kehrein, and Stefan Rabanus, editors, *Language and Space. An International Handbook of Linguistic Variation. Volume 2: Language Mapping*, number 30.2 in Handbücher der Sprach- und Kommunikationswissenschaft, pages 433–457, 2201–2212. De Gruyter Mouton, Berlin, New York.

Wilbert Jan Heeringa. 2004. *Measuring dialect pronunciation differences using Levenshtein distance*. Ph.D. thesis, University of Groningen.

Roland Kehrein. 2012. *Regionalsprachliche Spektren im Raum. Zur linguistischen Struktur der Vertikale*. Number 152 in Zeitschrift für Dialektologie und Linguistik. Beihefte. Franz Steiner Verlag, Stuttgart.

Joseph B. Kruskal. 1983. An overview of sequence comparison: time warps, string edits, and macromolecules. *SIAM Review*, 25(2):201–237.

Satya Swarup Misra and Haripriya Misra. 1982. *A historical grammar of Ardhamāgadhī*. Ashutosh Prakashan Sansthan, Varanasi.

John Nerbonne. 2009. Data-driven dialectology. *Language and Linguistics Compass*, 3(1):175–198.

John Nerbonne. 2010. Mapping aggregate variation. In Alfred Lameli, Roland Kehrein, and Stefan Rabanus, editors, *Language and Space. An International Handbook of Linguistic Variation. Volume 2: Language Mapping*, number 30.2 in Handbücher der Sprach- und Kommunikationswissenschaft, pages 476–501. De Gruyter Mouton, Berlin, New York.

Thomas Oberlies. 2003. Aśokan Prakrit and Pāli. In Danesh Jain and George Cardona, editors, *The Indo-Aryan languages*, Routledge Language Family Series, pages 161–203. Routledge, London et al.

Simon Pickl and Simon Pröll. 2019. Ergebnisse geostatistischer Analysen arealsprachlicher Variation im Deutschen. In Joachim Herrgen and Jürgen Erich Schmidt, editors, *Sprache und Raum. Ein internationales Handbuch der Sprachvariation. Volume 4: Deutsch*, number 30.4 in Handbücher der Sprach- und Kommunikationswissenschaft, pages 861–879. De Gruyter Mouton, Berlin, Boston.

Richard Salomon. 1998. *Indian epigraphy: a guide to the study of inscriptions in Sanskrit, Prakrit and the other Indo-Aryan languages*. South Asia Research. Oxford University Press, New York et al.

Yves Scherrer and Philipp Stoeckle. 2016. A quantitative approach to Swiss German. Dialectometric analyses and comparisons of linguistic levels. *Dialectologia et Geolinguistica*, 24(1):92–125.

Ulrich Schneider. 1978. *Die großen Felsen-Edikte Aśokas. Kritische Ausgabe, Übersetzung und Analyse der Texte*. Number 11 in Freiburger Beiträge zur Indologie. Otto Harassowitz, Wiesbaden.

Sukumar Sen. 1960. *A comparative grammar of Middle Indo-Aryan*. Number 1 in Special Publications of the Linguistic Society of India. Linguistic Society of India, Poona.

Ryan Soh-Eun Shim and John Nerbonne. 2022. dialectR: Doing dialectometry in R. In *Proceedings of the Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 20–27, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Oskar von Hinüber. 2001. *Das ältere Mittelindisch im Überblick. 2., erweiterte Auflage*. Number 20 in Veröffentlichungen der Kommission für Sprachen und Kulturen Südasiens. Verlag der Österreichischen Akademie der Wissenschaften, Wien.

Martijn Wieling and John Nerbonne. 2011. Bipartite spectral graph partitioning for clustering dialect varieties and detecting their linguistic features. *Computer Speech and Language*, 25:700–715.

A Abbreviations of Aśokan Sites

Ahr	Ahraurā	All	Allāhābād
Are	Arerāj	Bah	Bahāpur
Bai	Bairāṭ	Bar	Barābār
Bhā	Bhābhrū Stone Inscription	Bra	Brahmagiri
Dha	Dhaulī	E$\underline{\underline{\text{r}}}$	E$\underline{\underline{\text{r}}}$raguḍi
Gav	Gavīmaṭh	Gir	Girnār
Guj	Gujarrā	Jaṭ	Jaṭiṅga-Rāmeśvara
Jau	Jaugaḍa	Kāl	Kālsī
Kan	Kandahār	Laġ	Laġman
Lum	Lumbinī	Mān	Mānsehrā
Mas	Maski	Mer	Merāṭh
Nāg	Nāgārjuni	Nan	Nandangaṛh
Nig	Niglīvā	Niṭ	Niṭṭūr
Pāḷ	Pāḷkīguṇḍu	Pāṅ	Pāṅgurāriā
Rat	Ratanpurvā	Rāj	Rājula-Maṇḍagiri
Rām	Rāmpūrvā	Rūp	Rūpnāth
Sah	Sahasrām	Sāñ	Sāñcī
San	Sannati	Sār	Sārnāth
Śāh	Śāhbāzgaṛhī	Sid	Siddapur
Sop	Sopārā	Tax	Taxilā
Ṭop	Ṭoprā	Uḍe	Uḍegoḷam

Hoisting the colors of Sanskrit

Gérard Huet
Inria Paris Center

Abstract

The Sanskrit Heritage Reader is a tool that transforms a Sanskrit text into a representation of all its possible segmentations as a word-by-word enunciation (*padapāṭha*), undoing the sandhi euphonic rules. It allows a human annotator to select relevant segmentations with the use of a man-machine graphical interface using a notion of colored segment. We discuss in this paper this notion of color, and argue about its linguistic status, similar to a notion of parts of speech, but also allowing to segment complex compounds into a linear representation of so-called pre-compounds. This permits to present the *padapāṭha* with segmented compounds, without having to decide prematurely of their exact internal morphology. Colors can be seen as the coarsest classification of morphemes allowing the regular representation of Sanskrit. The study reveals subtle difficulties in Sanskrit analysis not usually discussed in grammars, which deal extensively with generativity (*vṛtti*), but rarely discuss text analysis (*śābdabodha*).

1 Introduction

The Sanskrit Heritage Reader[1] is a segmenting service for Classical Sanskrit. It takes as input a piece of Sanskrit text and proposes the various ways the text may be segmented into as a sequence of word forms (*pada*) by undoing phonetic glueing (*sandhi*). Such a sequence is called a word-by-word recitation (*padapāṭha*) of the piece of text (*śabda*). Actually, it does a bit more: it breaks compounds into their constituents, profiting of the fact that the intra-compound sandhi obeys the same rules as sentential sandhi. Thus the tool may be used for resolving compounds into constituents as well as providing word decomposition of sentences, in view of further analysis by various parsers to recognize its meaning.

This segmenter was designed as a finite-state relational process or Eilenberg machine (Huet, 2005). It assumes the prior generation of word forms and necessary phonemes from a given vocabulary defined by a generative lexicon, in our case the Sanskrit Heritage dictionary. The dictionary defines lexical items given with generation parameters, such as the present class (*gaṇa*) of a verb, or the gender of a nominal. Thus the various data-banks used by the segmenter store the corresponding signifier not just as a list of phonemes (*varṇa*) but as a tagged entity exhibiting its generation parameters. Its tags are used for informing the user with the generation parameters of the various pieces of the *padapāṭha*. Furthermore, the various kinds of segments are displayed with a characteristic colour, making explicit its combinatorial power. Thus finite verb forms such as *gacchati* are represented as red segments, nominal forms such as *mudrā* are represented as blue segments, and nominal stems like *hasta* are represented as yellow segments, usable as first component of compound form *hastamudrā*, itself represented as two consecutive segments, yellow *hasta* followed by blue *mudrā*.

This color-coding turned out to be very effective to select the right segmentation of a text from its possibly many potential segmentations. It is specially so when using the graphical interface

[1] https://sanskrit.inria.fr/DICO/reader.html

of the Reader (Goyal and Huet, 2016). The goal of this note is to give a systematic exposition of this notion of color, and to investigate its proper linguistic status.

2 Basic colors

The main partition of Sanskrit padas is between nominals, constructed with the *sup* declension suffixes, and verbal finite forms, constructed with the *tiṅ* conjugation suffixes, according to Pāṇini's sūtra (I,4,14): *suptiṅantam padam*. We use the color red for *tiṅanta* padas, this hot color being appropriate for the dynamic action that they denote, while the cool blue color is used for *subanta* nominal padas.

Thus on input *vavarṣarudhiram* we get the colored segments indicated in Figure 1.

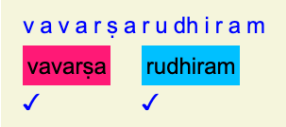


Figure 1: vavarṣarudhiram

When clicking in the interface on red *vavarṣa*, one gets its morphological tag as: `[vṛṣ]{pft. ac. sg. 3 | pft. ac. sg. 1}` and similarly on blue *rudhiram*, one gets `[rudhira]{m. sg. acc. | n. sg. acc. | n. sg. nom.}`. This shows that this segmentation may be interpreted in 6 different ways according to their inflexion *vibhakti*, and consequently as possibly various meanings. Here our colors are essentially marking part-of-speech of the words, no more.

Thus on sentence *bhīmorudhirampibati* we get the same blue color for the agent Bhīma and for the object of its drinking, blood, as shown in Figure 2.

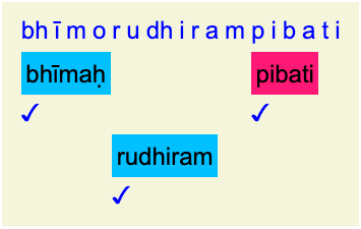


Figure 2: bhīmorudhirampibati

Actually, we set aside the vocative forms, colored green to make clear that they do not contribute to the meaning of the text, but rather serve are indications pertaining to the enunciation conditions. They belong to the discourse structure of the text, rather than to the sentence structure proper. Also, grammatical content-less words such as conjunctions and other adverbials, which are indeclinable stand-alone items, are colored mauve, to distinguish them from blue nominals, which denote participants (*kārakas*) having thematic role in the situation described by the input sentence. Finally, (inflected) pronouns are colored a lighter shade of blue than actual nominals.

Thus on input *deva adyamamabhāryāpacati* "Majesty, today my wife is cooking", we get the multicolor rendering shown in Figure 3.

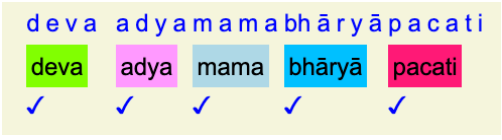


Figure 3: deva adyamamabhāryāpacati

3 Nominal Compounding

In Sanskrit, compounding is productive, without limitation on the number of its components. The standard compounding operation applies to two nominal forms, glueing by sandhi the stem of the first to the second one. Thus, a yellow cloth, *pītam ambaram*, may be contracted in one pada by compounding, yielding form *pītāmbaram*. We represent such a *karmadhāraya* compound with two segments, one yellow for the left stem *pīta*, preceding the blue right form *ambaram*, like in Figure 4 below.

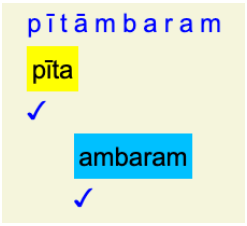


Figure 4: pītāmbaram

The notation generalizes in a straightforward manner to *dvandva* compounds, with possibly many yellow segments, like for *aśvāvyuṣṭrāḥ* (horses, sheep and camels), represented in Figure 5 below.

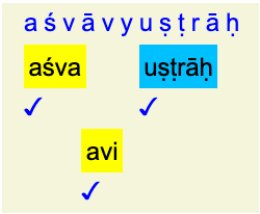


Figure 5: aśvāvyuṣṭrāḥ

But sometimes there may be an ambiguity, in case of an embedded compound such as *daśarathaputraḥ* which could be analysed theoretically as *(daśa-ratha)-putraḥ* or *daśa-(ratha-putraḥ)*. This ambiguity is often avoided by not segmenting proper names, when they are properly lexicalized, like here shown in Figure 6.

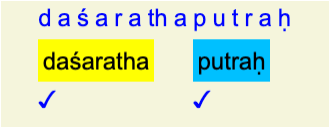


Figure 6: daśarathaputraḥ

Also, substantives obtained by compounding with a root form like *nṛpa*, king "who leads men" should be lexicalized as a frozen form (*rūḍha*). This also applies to lexical items like *malamūka* (deaf and dumb). This way most complex compounds may be understood un-ambiguously as left-associating, like in Pañcatantra's famous 10-components *samāsa* shown in Figure 7.

Now that we understand the representation of determinative (*tatpuruṣa*) compounds, let us move to the possessive class (*bahuvrīhi*). Here we have a problem, because such exocentric compounds, turning into adjectives, may transform their right hand component in order to assign the compound some gender that is not allowed for the original right-hand side nominal. That is, their gender is not a synthesized attribute, but it is inherited from the surrounding noun phrase head. Thus we may get *pītāmbaraḥ* "who wears a yellow garment", typically as an

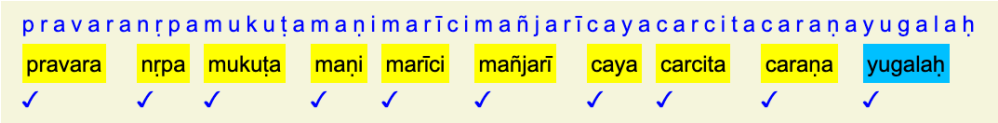


Figure 7: pravaranṛpamukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ

epithet of Viṣṇu, where the masculine form *ambaraḥ* of its right component is not a valid form of the neuter substantive *ambaram*. We solve this problem by generating extra forms for such substantives, usable only in *ifc* (*in fine composi*) position. We assign the color cyan to such components, yielding the representation shown in Figure 8.

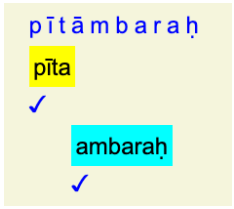


Figure 8: pītāmbaraḥ

This same mechanism is used to enter forms of lexical entries which are restricted to the ifc role, like *-kāra*, or root ifc forms. Thus for *kumbhakāraḥ* (pot-maker) we get Figure 9.

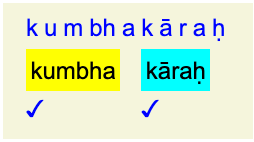


Figure 9: kumbhakāraḥ

An additional difficulty concerns feminine substantives like *daṃṣṭrā*. First, they must generate an iic (*in initio composi*) ended in the feminine suffix *-ā*, in order to allow recognition of a compound such as *daṃṣṭrākarālaḥ* "exhibiting horrific fangs". But we must also license a segment in the masculine stem *-daṃṣṭra-* to allow constructing a *bahuvrīhi* compound, itself usable as left component of further compounds. We use the khaki color for this (rather rare) occurrence. Here is an example, for *bhagnanakhadaṃṣṭravyālam* "wild beasts deprived of their claws and fangs", shown in Figure 10.

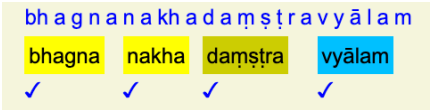


Figure 10: bhagnanakhadaṃṣṭravyālam

Please note that such khaki segments signal their use both as an ifc on their left, to build a stem now used as iic to their right. Thus it is a stem that is not an iic, but rather the last component of the *prātipadika* of a compound ended in an ifc component. Thus in *kumbha-kāra-putraḥ* segment *kumbha* is yellow, but segment *kāra* is khaki, as right component of the *prātipadika* of compound pada *kumbhakārasya*. It thus partly disambiguates the grouping of stems used to build a complex compound.

Most regular compounds are left-associative, like the example in Figure 7. The successive compositions of meanings allows understanding such long compounds in real time, without

taxing short-term memory. Many of the exceptions to this rule concern proper names, and frozen technical terms which may not be compositional in meaning. Such items should be lexicalised, and the compounds containing them will be linear, understandable by left to right composing of the meanings of its individual segments, like in the above example of Figure 7.

At this point we note that, even if proper names are lexicalized, we must allow for the possibility of using a descriptive compound in the vocative. For instance, on input *namaste'stu mahāmāye śrīpīṭhe surapūjite* (I honor Thou, Great Illusion, Enthroned by Fortune, Blessed by the Gods), we get Figure 11.

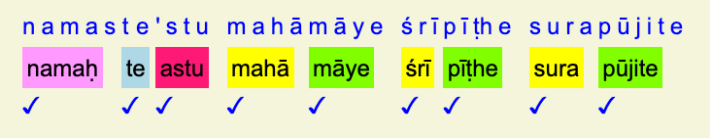


Figure 11: namaste'stu mahāmāye śrīpīṭhe surapūjite

This concludes our discussion of regular nominal compounds. There exist also exceptions to the two rules that are implicit in those: firstly, the first component does not carry the *vibhakti* suffixes, and is reduced by *luk* to its stem form; secondly, retroflexion does not cross over the compound frontier. When one of these two conditions is not met, the compound must be lexicalized. For instance, consider *agrevaṇam* "border of the wood", where the first component is in the locative (*aluk*), and the second component *vanam* incurs retroflexion because of its left context. Such compounds are in small number, mostly to form proper names like *Janamejaya* or *Rāmāyaṇa*.

4 Adverbial compounds

An important, although diverse, family of compounds is referred to by the name *avyayībhāva* "turned into an indeclinable". A typical representative is *yathāvṛddhi* "according to the phase of the moon". Its left-hand side is the invariable *yathā*, and its right side is the neuter form *vṛddhi*. Here this right component cannot be used stand-alone, since it is not a form of feminine nominal *vṛddhi*. We chose to represent such a compound with 2 segments, a pink iic and a mauve ifc, as shown in Figure 12.

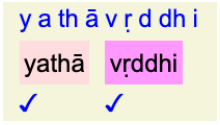


Figure 12: yathāvṛddhi

The *avyayībhāva* compound family comprises many unusual constructions, and sometimes the indeclinable is in its right hand part rather than its left part, like *sūpaprati* "with some sauce". Such exceptional items must be lexicalized.

Absolutives are colored mauve, in keeping with their adverbial nature, but a specific color could have been specified. Absolutives in *-tvā* are provided for roots without preverbs, absolutives in *-ya* attach to verbs provided by preverbs, which are represented glued to their root, like finite verbal forms. They are thus all mono-segment. Occasionally they appear negated, in which case we get two segments, the privative prefix being a stand-alone segment, like in the following example, *yato vāco nivartante aprāpya manasā saha* "*(brahman* is) where speech returns from, unable to grasp it with the mind", shown in Figure 13.

Also in mauve are occasional absolutives in *-am* (the so-called *ṇamul* construction), and infinitive forms in *-tum*. At this point it should be mentioned that the various participial forms are colored blue, in keeping with their nominal declension.

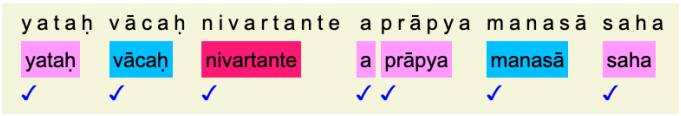


Figure 13: yato vāco nivartante aprāpya manasāsaha

Other generative adverbs are the so-called *tasil*, adverbs of manner like *ubhayatas* (from both sides), colored mauve.

5 Verbal compounds

Finite forms of verbs are usually represented as a single segment. That is, potential preverbs are glued to the root form, contrarily to the original design of our Reader, where preverbs were represented as separate segments. Similarly, prefixes such as prepositions, but also particles such as *sa-*, *su-*, *dus-*, *ku-*, etc are considered morphemes of the inner morphology of padas, and not segments to be compounded. This also applies to the privative prefixes *a-/an-*. This induces extra lexicalisation, but greatly simplifies the user interface. This is also the case for secondary *taddhita* suffixes such as *-tva*, *-tā*, *-vat*, etc.

There is an exception for the so-called periphrastic perfect, obtained by glueing a special morpheme in *-ām* to a form in the perfect of one of the auxiliary verbs *as*, *bhū* and *kṛ*. We represent such forms as two segments, the form in *-ām* in orange, while the auxiliary perfect form is a usual verbal form in red; Thus, for sentence *āsāṃcakre hvayāmāsa ca* (he sat and called) we get Figure 14. By contrast, so-called periphrastic future forms are monosegmental, treated as a specific conjugation.

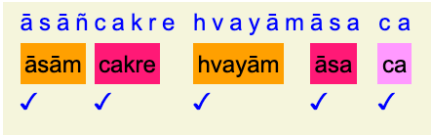


Figure 14: āsāṃcakre hvayāmāsa ca

Another kind of verbal compound is the so-called *cvi* inchoative construction. It allows root forms of the 3 auxiliaries to be prefixed with nominal stems in *-ī* (or sometimes *-ū*). Some other initial segments like *sākṣāt* may also be used, the class of such segments is called *gati*. By analogy with periphrastic perfect, we use the orange color to represent such initial segments. Finally, the construction is extended to nominal compounds, formed with a *gati* followed by a declined *kṛdanta* (first level nominal derivative) of the auxiliaries. Figure 15 shows a few typical such forms.

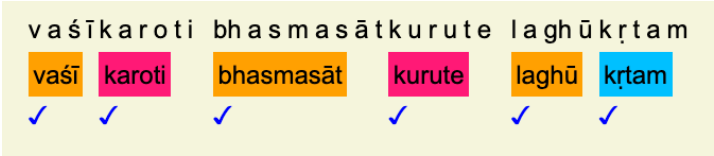


Figure 15: Various inchoative compounds

It should be noted that these nominal inchoative compounds may themselves be subject to compounding, like in *vivādāspadībhūtam* (that has become litigious) in Figure 16 below. Please note that we have here one pada represented as three segments.

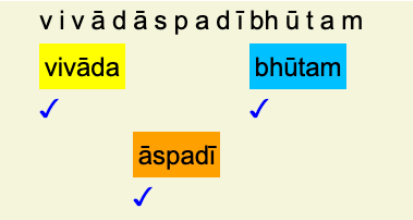


Figure 16: vivādāspadībhūtam

6 Special constructions

One last productive morphological construction is exemplified by the bahuvrīhi compound *vaktukāmaḥ* "desirous of speaking". Its left component is an infinitival form deprived of its final *-m*, and its right component is a form of *kāma*, *manas* or *śakya*. This construction is not very frequent, but it is productive, used by the best authors, and stated in a *vārttika* to *sūtra* (VI,1,144). It must therefore be accommodated, as a separate construct. We use again the orange color for the infinitive segment *vaktu*, in analogy with the verbal compounds above, while we use the cyan color for the ifc segment *kāmaḥ*. We show in Figure 17 the example *punarapi vaktukāma ivāryo lakṣyate* "Your honor appears desirous of speaking again".

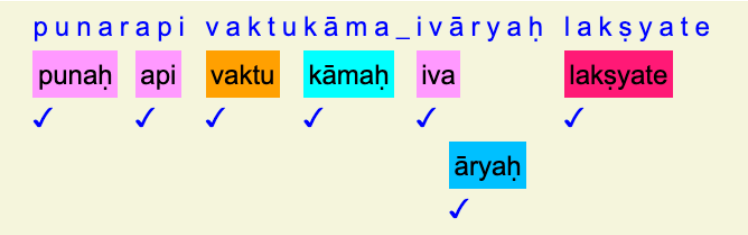


Figure 17: punarapi vaktukāma ivāryo lakṣyate

It is not always obvious to decide whereas an unusual formation ought to be productive or treated as a lexicalized exception. For instance, *sūtra* (II,1,72) alludes to irregular compounds in the group (*gaṇa*) headed with *mayūravyaṃsaka*. This group is indeed a motley crew, with bizarre isolated items like *jahijoḍaḥ* (in the habit of hitting one's chin) which may be simply lexicalized. Others mentioned in *vārttika*s allude to generic schemas like *aśnītapibatā* built on two imperative forms compounded as a feminine substantive, meaning a festive occasion where the two actions are repeatedly performed, like in this case eating and drinking. Indeed the *gaṇapāṭha* lists 12 such attested substantives, raising the question of whether this construction is productive. We decided against accepting this scheme as generic, because this would oblige us to introduce two specific colors, one for imperative iic segments, the other for imperative ifc segments construed as feminine substantives, and we rather decided to lexicalize the 12 attested occurrences. One may even question whether such forms, completely atypical in making nominal compounds out of verbal forms, deserve the status of grammatically correct Sanskrit. Would such crude colloquial expressions have been considered *śiṣṭa* by Pāṇini? Aren't they just tongue-in-cheek suggestions by Kātyāyana as implicit mockery of grammar's pretension to straight-jacket a living language?

We end this section by mentioning one last color, grey, for input chunks that are not recognized by the lexer. This may be because of an incompleteness of the generative lexicon, or this may be due to an un-grammatical item in the input, like shown in Figure 18, for input *na tathā bādhate śītaṃ yathā bādhati bādhate* "It's not so much the cold as 'bādhati' that bothers me", told by a palanquin bearer to the bogus pandit he is carrying, who had asked: *api śītaṃ bādhatī*?

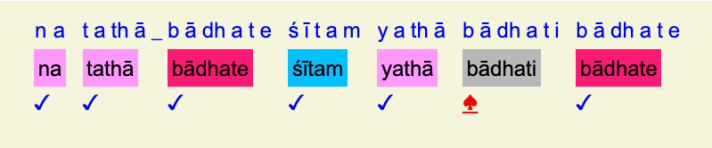


Figure 18: na tathā bādhate śītaṃ yathā bādhati bādhate

7 Making linguistic sense of colors

We have given a fairly extensive treatment of the visualisations of various Sanskrit constructions, as they are recognized by the Sanskrit Heritage segmenter, and as they are displayed in the graphic interface of its Reader. In this visualisation, various segments are displayed in various colors giving some information about morphemes that are necessary to represent Sanskrit padas. Most of the complexity of the treatment is due to the numerous ways of formation of compounding, and specially the changes of gender and number suffixes that occur along compound derivation. However, it is not straightforward to give a precise characterisation of our color encodings in terms of linguistic concepts.

Let us first sum up the various color encodings in the following table:

color	*part of speech*
blue	inflected nominal
red	finite verb form
yellow	nominal stem as iic
green	vocative
mauve	indeclinable, particle
light blue	pronoun
cyan	ifc
khaki	ifc stem
pink	indeclinable iic
orange	perfect iic, gati, infinitive stem
grey	unrecognized

For one thing, the colors presented above are ambiguous. For instance, the orange color is assigned to three kinds of morphemes: periphrastic perfect stems in *-ām*, *gati* verbal prefixes, and infinitives deprived of their final *-m*. It is obvious that these three constructions are independent, and that the common color is just merging these three constructions into a general category of "verbal compounds". We could indeed easily distinguish the three by visualising them differently, and each shade of orange would then designate unambiguously a precise grammatical operation.

Then, the status of "segment" itself is not entirely obvious. Actually, each segment generally represents a choice among several possible interpretations, explicit only if we click on the segment to uncover its possibly many tags. Consider for instance *pītāmbaraḥ* in Figure 9 above. The cyan component *ambaraḥ* is tagged `{m. sg. nom.}`. This does not mean that *ambaraḥ* is a valid masculine substantival form, which it is not, just that the particular segmentation *pīta-ambaraḥ* may be interpreted as the masculine form of an adjective obtained by the bahuvrīhi interpretation of the compound: "he who has a yellow garment", in the nominative case. Thus the tag is not a tag of the segment, but of the whole compound pada. Now consider *pītāmbaram*. Here *ambaram* may have two colors. As a cyan segment, it would be tagged `{m. sg. acc.}`, meaning similarly that this particular segmentation *pīta-ambaram* may be interpreted as the same masculine adjective, but now in the accusative case. Whereas if it appears as a blue segment, it bears the multitag `{n. sg. acc. | n. sg. nom.}`, meaning that here *pīta-ambaram* could be interpreted either as the *karmadhāraya* compound obtained by contraction of *pītam ambaram* "yellow cloth" (in the nominative or accusative case), or possibly the bahuvrīhi adjec-

tive, but now fit with the neuter gender, in order to serve as determinant to some neuter head noun. This shows that in general, even if we restrict segments to single tags, a given compound segmentation does not characterize its mode of formation.

We must explain the reason for this apparent confusion. It stems from the morphological generation process, that compiles the lexicon entries by feeding the various databanks corresponding to our colors. On processing the entry *ambara*, which is marked as a neuter substantive, it generates ifc items (cyan) only for the missing genders, in order to be able to recognize masculine and feminine instances of the bahuvrīhi adjective. Indeed, on entry *pītāmbarā*, our segmenter generates only a cyan ifc *ambarā*, with tag `{f. sg. nom.}`. The rationale of not generating a cyan *ambara* of gender neuter when analysing *pītāmbaram* was that it would just be redundant with the *karmadhāraya* segmentation, from the point of view of possible tags. Thus it is left to the next stage of interpretation after segmentation, i.e parsing, to make sense of sequences of padas decorated with *vibhakti* parameters. And it is the parser that must guess, in case of the neuter gender, whether the segment fits with the surrounding context, in order to build the agreeing nominal phrases. Thus in the case of a neuter interpretation, it is the burden of the parser to choose between the two interpretations of the compound pada as a substantive (*viśeṣya*) or as an adjective (*viśeṣaṇa*). Whereas in the case of cyan segments, there is no such choice, they must be adjectives. Unfortunately, the parser does not have the information, if we transmit to it just morphological tags, but not the color that in the cyan case bears information.

This discussion shows that there is not a good transmission of information if the result of the segmenter is just a list of morphological tagged pada. It suggests that we could do better, if the *viśeṣya/viśeṣaṇa* status of compounds is transmitted when known. And this is where colors may help. But in order to deal correctly with the neuter case, we must distribute it in the two colors. Thus on *pītam ambaram* we could generate a blue *ambaram* bearing tags `{n. sg. acc. | n. sg. nom.}`, as presently, but also a cyan *ambaram* bearing now tags `{n. sg. acc. | n. sg. nom. | m. sg. acc.}`. So we would have more singly tagged segmentations, but now we can transmit their substantive/adjective character with the color, that would have now a clearer linguistic status. Thus we could generate more precise information by transmitting the color along with the tagging, but at the price of overgeneration — more solutions would be allowed.

Unfortunately, there are many more compound constructions which are ambiguous with regard to their *viśeṣya/viśeṣaṇa* status, specially when compounding is iterated, typically in poetic style (*kāvya*). Thus when we recognize a series of embedded compounds $X_1X_2...X_nY$ we only propose a list of n yellow stems (*prātipādika*) followed by a blue (or cyan) pada, and this ignores not only the exponentially many ways in which the X sequence is the frontier of binary compounds, but also the *viśeṣya/viśeṣaṇa* status of internal binary compounds, according to their *bahuvrīhi/tatpuruṣa* actual construction. Not to speak of the fact that a consecutive sequence of X's and possibly Y might represent one dvanda multi-component compound.

Two remarks are in order. The notation of syntactic constituents used by Gillon to represent phrase-structure in Sanskrit has a special mark "-B" to indicate *bahuvrīhi* raising (Gillon, 1995; Gillon, 2009). Unfortunately, this mark has a null phonetic realization ("morphological zero") and this is one of the main causes of Sanskrit ambiguity. Sometimes, Navyanyāya terms, using compounding for relational expressions, occasionally use suffixing by a taddhita pratyaya *-ka* (technically called *kaP*) to make explicit *bahuvrīhi* raising. In this case, the ambiguity is lifted by explicit phonetic marking — the extra *ka* syllable is the information that decides its *viśeṣaṇa* status. Another device is the accent. Accent on the first component marks its *bahuvrīhi* character. Unfortunately, accent is not marked in Classical Sanskrit, so the parser must decide the matter. And actually, in poetry, the ambiguity may be used to have two different interpretations in the two branches of a double-entendre (*śleṣa*).

This discussion shows that it would be wrong to interpret the two colors blue and cyan as part-of-speech markers, identifying respectively substantives and adjectives. They both pertain to nominals, i.e. subantas, the *viśeṣya/viśeṣaṇa* status of which is not intrinsic, but depends

on the context. Furthermore, the distinction between nouns and adjectives in Sanskrit is not completely clear. Actually, even an authority such as S. D. Joshi says: 'it is very difficult to provide a satisfactory definition of the concepts *viśeṣaṇa* and *viśeṣya*, because we don't have adequate criteria for the differentiation of adjectives and substantives"(Joshi, 1966). Additional discussion on this question is provided in (Dash, 1987).

Actually, the design rationale of the segmenter is just to build the *padapāṭha*, and to transmit to the next layer of interpretation the extra information it may have gathered on the way; thus, the Heritage segmenter, being lexicon-directed, is aware of morphological tags, which are parameters of their synthesised forms, generated from the lexicon, and thus transmits padas with their *vibhakti*, but should not try to guess thematic roles (*kāraka*) or even recognize noun phrases. It is up to the next stage of parsing to understand the situation semantics. Thus, the Heritage shallow parser (Huet, 2007), just by grouping phrases in the first three cases (nominative, accusative, instrumental), guesses the thematic roles of the situation, without having to understand the structure of compounds. In comparison, Amba Kulkarni's dependency parser (Kulkarni, 2019) goes further in analysing sentences, and uses semantic principles such as compatibility (*yogyatā*). This way, her parser is able to go as far as translation to Hindi.

8 Geometry of colors

Our colors actually indicate states of the segmenting automaton, which correspond to a mixture of morphological categories and their mutual combinatorics. Each color corresponds to a databank of morphemes. These morphemes are assembled by external sandhi in order to form padas, according to the finite automaton graph shown in Figure 19 below.

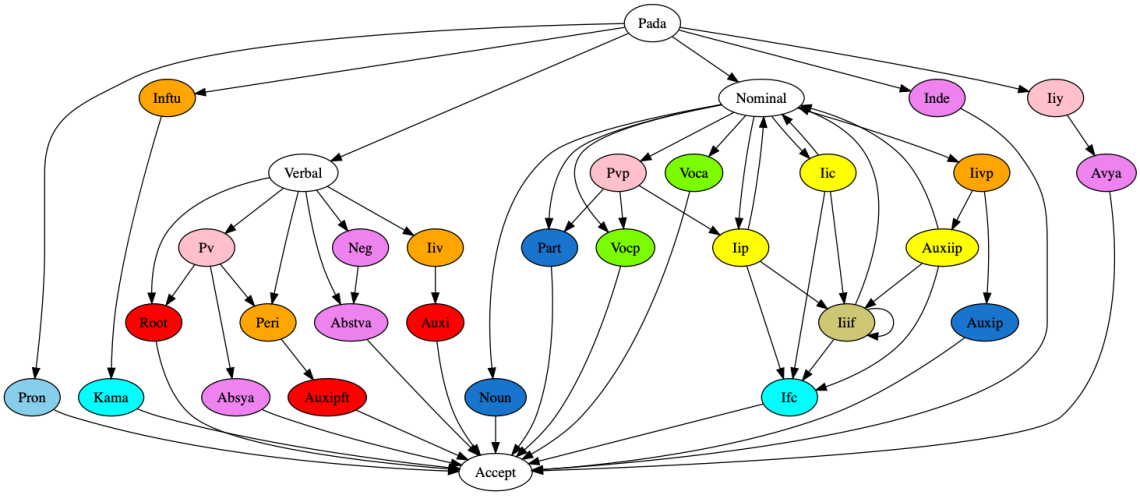


Figure 19: Colors geometry of padas

Our design of the segmenter evolved over time. Initially, preverbs were recognized as independent segments. But this feature was deemed confusing by users, with non-connected preverbs floating around. Thus we decided to glue them on the fly, either to finite verb forms of the roots, or to their participial forms (*kṛdanta*). At glueing time, we recognize that the particular sequence of preverbs is consistent with the roots' requirements. This allows us to dispense with huge databanks storing the glued preverbs, and to satisfactorily deal with problematic sandhi operations like in the recognition of *ihehi*, with a mechanism of *phantom phonemes* (Huet, 2006). It is to be noted that assuming preverbs as prefixes of verbal forms may prevent recognition of some Vedic sentences, where prepositions may float around more freely.

Thus preverbs still appear in the finite automaton state transition diagram, but are silently glued to verbal forms when visualised by the Reader. This is important for instance for prop-

erly recognizing absolutive forms, ending in *-tvā* for roots, and in *-ya* for verbs prefixed with preverbs (using *kṛt* suffixes respectively *KtvāandLyaP*). This induced us to generate *kṛdanta* forms, notably participial forms, in a separate databank from other nominals (see Part state in Figure 19). Such forms are generated with compound tags indicating their internal morphology. For instance, *pragatam* is tagged as `[pra-gata { pp. }[pra-gam]]{n. sg. acc. | n. sg. nom. | m. sg. acc.}`. This information is useful for further stages of parsing, since they are useful to determine their combinatorial power (*ākāṅkṣā*). However, we kept the blue colors for such *kṛdanta* forms, in an effort to avoid confusion of the annotator. We could of course have distinguished them with some different shade of blue, but the information is there anyway in the tag when needed.

Another improvement was to treat privative prefixes *a-/an-* as part of internal morphology, as opposed to independent iic. segments. This was crucial for helping annotators to notice situations of ambiguity where a segment ending in *-ā* precedes a possibly negated notion, leading to two opposite analyses. On the other hand, this obliged us to lexicalize such privative compounds, and this is problematic for long-scope negation, where the privative prefix operates on a compound (Gillon, 1987). This treatment of negation is homogeneous with the treatment of nominal prefixes like *su-*, *dus-*, *sa-*, *vi-*, etc. This relieves the segmenter from overgenerating with such mono-syllabic particles if treated as genuine stems, but at the cost of having to lexicalize the corresponding compounds. Symmetrically, we treat taddhita suffixes similarly. Again, this is problematic when these particles have a long scope, affixed to possibly long compounds. This also prevents recognition of long compounds used in Navyanyāya relational terms, where taddhita suffixes may cascade arbitrarily, often in un-Pāṇinian ways.

9 Psychology of colors

From the point of view of programming, our colors are implemented as a discrete set of atoms, that does not have any structure. However, from the point of the human-machine interface, our colors are mapped into RGB color representations, which are carefully chosen tints with features like hotness/coldness along a scale of activity associated with the corresponding segment of speech. Thus verbal forms, denoting actions, are bright red, while substantives are painted of a deep sky blue cold color since they have a more static role as participants to a situation. Pronouns are like budding nouns, of a ligher shade of blue. Exocentric [*bahuvrīhi*] compounds, of cyan color, as adjectives show some eagerness to associate with a deeper blue substantive they will qualify. Indeclinables are mauve, showing some intermediate status between nouns and verbs. Finally vocatives are of a flashy acid green, urging the user to either discard them, or to admit them when appropriate, but as discourse operators independent of the phrasal structure. Compounds have their iic marked as yellow, orange, or pink, according to the nature of their ifc, and with a slightly paler tint.

Thus these color codings have some rationale as psychological conceptual triggers, and this is very important from the point of view of helping the user to select the appropriate segmentation solution. Some care was taken also to have a consistent palette of colours, not too aggressive, and combining smoothly in order to offer a satisfactory esthetic experience. It is to be remarked that Amba Kulkarni's Saṃsādhanī platform uses also colors in its interface, and these colors are reminiscent of our own encodings, but with more shades of colors, because the parser process needs to distinguish clearly the cases of nominals, in order to group them easily as noun phrases by agreement of their respective colors. In our segmenter, we made the choice of having few colors (10 in total) in order not to have the annotator lose time in a finer grain selection that is not relevant at segmentation time.

10 Conclusion

We have given a complete description of a notion of color which characterizes sorts of Sanskrit morphemes necessary to account for the notion of pada. All productive constructions of surface

Sanskrit morphology are covered. We may consider this notion of color as a notion of parts of speech appropriate for Sanskrit. Segment colors are not independent, since the color transitions are governed by the Sanskrit segmenting finite-state automaton when recognizing padas. Thus, our colors indicate parts of speech categories, but now obeying algebraico/geometric laws corresponding to their mutual interactions as morphemes in pada production, and obtained by quotienting the state space of the segmenting automaton.

In this study, we have extended the initial dichotomy between subantas and tiṅantas in order to accommodate the various notions of compound word. The notion of nominal stem (*prātipādika*) came prominent, a sequential list of them being the key to collapse long compounds in the linear frontier of their parse trees, gaining an exponential factor by avoiding premature settlement of non-deterministic choices that are best solvable in the subsequent workflow of the global parser. This puts in evidence the need to introduce a variety of morpheme categories and their mutual connectivity requirements. It is satisfactory that most of these categories correspond to Pāṇinian notions, such as *cvi*, *gati*, *tasil*. Even less expected notions, such as the (khaki) stems of ifc-only items used as pseudo-iics, are known in the tradition as *bhāṣitapuṁska*. This conversion to the masculine stem of iics is the basis for our generation of yellow iic segments, but we need it also for conversion to masculine stem of inner compounds built on an item usable only in ifc, a subtle consideration.

One construct stands out as exceptional, allowing mixture of tiṅanta material (the infinitive *tumun* with *lopa* elision of final *-m*) with subanta padas (forms of *kāma*, *manas* and rarely *śakya*). This construct is often passed over in Sanskrit grammars, or stated as belonging to the late language (Whitney, 1924)§968g, (Apte, 1885)§181, (Renou, 1956) p72, but it is productive in classical Sanskrit. Indeed it is not discussed in Pāṇini's Aṣṭādhyāyī, and only appears in Kātyāyana's vārttikas to sūtra (VI,1,144); The same goes true for perfect periphrastic forms, which are exceptional in Vedic, restricted to auxiliary *kṛ* in sūtra (II,1,40), and only described fully in Patañjali's Mahābhāṣya (Renou, 1956) p72. We can thus witness diachronic evolution of the language by the necessary successive adjunctions of items to Figure 19. Refer also to the discussion on *aśnītapibatā* above.

The notion of pre-compound, allowing the collapse of an arbitrary number of embedded compounds, has put in evidence the necessity of providing segments which are stems of forms which normally occur only as right-hand sides of compounds, exemplified by the khaki *daṣṭra* in Figure 10. This problem is not discussed in Western Sanskrit grammars, to our knowledge. Such rare segments serve to partly disambiguate the compounds in which they appear, since they mark a boundary between two different compounds.

We have also departed from the standard presentations by separating vocative forms, deemed to belong to discourse analysis rather than sentential recognition. Such vocatives (as well as interjective particles like *bhoḥ*) are colored as acid green, and stand out in the interface in order for the annotator to decide between them and iic stems, since this is a frequent ambiguity, notably for stems in *-a*. The rationale is that the choice of the vocative must be consistent with the global context, known to the annotator from the particular text she/he is studying, or from semiotics considerations.

In summary, we have proposed a regular description formalism for *padapāṭha* sequences based on a notion of colored segment, deemed to be complete for Classical Sanskrit, and appropriate for quick selective annotation of corpus by human users with minimum training.

References

Vāman Shivarām Apte. 1885. *The Student's Guide to Sanskrit Composition. A Treatise on Sanskrit Syntax for Use of Schools and Colleges*. Lokasamgraha Press, Poona, India.

G. Cardona. 1988. *Pāṇini: his work and its traditions*. Motilal Barnasidass.

Siniruddha Dash. 1987. Adjectives and substantives as separate categories in Sanskrit. *Lokaprajñā*, 1,1:90–96.

Pierre-Sylvain Filliozat. 1988. *Grammaire sanskrite Pâninéenne*. Picard, Paris.

Brendan S. Gillon. 1987. Two forms of negation in Sanskrit: *prasajyapratisedha* and *paryudās-apratiṣedha*. *Lokaprajñā*, 1,1:81–89.

Brendan S. Gillon. 1995. Autonomy of word formation: evidence from Classical Sanskrit. *Indian Linguistics, 56 (1-4)*, pages 15–52.

Brendan S. Gillon. 2007. Exocentric (bahuvrīhi) compounds in classical Sanskrit. In Gérard Huet and Amba Kulkarni, editors, *Proceedings, First International Symposium on Sanskrit Computational Linguistics*, pages 1–12.

Brendan S. Gillon. 2009. Tagging classical Sanskrit compounds. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Linguistic Modeling*, 4(2):117–126.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614.

Gérard Huet, 2006. *Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich*, chapter Lexicon-directed Segmentation and Tagging of Sanskrit, pages 307–325. Motilal Banarsidass, Delhi.

Gérard Huet. 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.

S. D. Joshi. 1966. Adjectives and substantives as a single class in the parts of speech. *Publications of the Centre of advanced study in Sanskrit, University of Poona*, Class A, no 9.

Amba Kulkarni. 2019. *Sanskrit parsing based on the theories of Śābdabodha*. Indian Institute of Advanced Study (DK Printworld distr.).

K V Ramakrishnamacharyulu Pavankumar Satuluri and Amba Kulkarni. 2016-17. Order of operations in the formation of Sanskrit compounds with special reference to introduction of samāsānta element and deletion of case endings. *Journal of Oriental Institute, Vadodara*, 66(4):77–86.

Louis Renou. 1942. *Terminologie grammaticale du sanskrit*. Honoré Champion, Paris.

Louis Renou. 1956. *Histoire de la langue sanskrite*. Editions IAC, Lyon.

Louis Renou. 1966. *La Grammaire de Pāṇini*. Ecole Française d'Extrême-Orient, Paris.

Pavankumar Satuluri and Amba Kulkarni. 2013. Generation of Sanskrit compounds. In *Proceedings of ICON 2013, the 10th International Conference on NLP*, pages 77–86, Noida, India.

Rama Nath Sharma. 1987-2003. *The Aṣṭādhyāyī of Pāṇini (6 vols)*. Munshiram Manoharlal Publishers.

William Dwight Whitney. 1924. *Sanskrit Grammar*. Leipzig. 5th edition.

Using TEI for digital Sanskrit editions containing commentaries A study of Kālidas's *Raghuvaṁśa* with Mallinātha's *Sañjīvani*

Tanuja P. Ajotikar
Assistant Professor, The Sanskrit Library
Co-director Sanskrit Department India Programs,
Maharishi Vedic University, Vlodrop, The Netherlands
tanuja@sanskritlibrary.org

Ketaki Kaduskar
Research Assistant, The Sanskrit Library
ketaki.kaduskar@gmail.com

Peter M. Scharf
President, The Sanskrit Library
Adjunct Professor, IIIT Hyderabad
scharf@sanskritlibrary.org

Abstract

In the present project of making a TEI digital edition of Mallinātha's commentary *Sañjīvanī* on Kālidāsa's *Raghuvaṁśa*, we encountered material that called for expansion of the procedures for creating XML editions of Sanskrit texts containing commentaries in accordance with the Text Encoding Initiative (TEI) guidelines. Ajotikar and Scharf (2023) previously described standardized procedures for digitizing the Sanskrit texts containing commentaries. To accommodate the additional material encountered in the present project we created several additional values of the `seg` element's `type` attribute: `alternate`, `def`, `paraphrase`, `syntax`, and `constituent`. These expand the categorization of glosses, refine categorization of derivational material, and deal more precisely with comments on compound constituents. We also verify, correct and make additional citations thereby making substantial contributions to textual research in the process of undertaking the higher-level encoding of the commentary.

Keywords: XML, TEI, Sanskrit, commentary, text-encoding

1 Introduction

Ajotikar and Scharf (2023) described standardized procedures for creating XML editions of Sanskrit texts containing commentaries in accordance with the Text Encoding Initiative (TEI) guidelines and discussed issues related to it that arose in their mark-up of Bhaṭṭi's *Rāvaṇavadha* with Mallinātha's commentary. In this paper, we present additional issues that arose in the TEI mark-up of Kālidas's *Raghuvaṁśa* with Mallinātha's commentary *Sañjīvanī*. We completed mark-up of the first five contos of the *Raghuvaṁśa* and the first three cantos of the *Sañjīvanī*.

2 File preparation procedure

The file preparation procedure includes three phases. In the first phase, the base text and the commentary are transcoded into the Sanskrit Library Phonetic ASCII encoding (SLP1), each in its own file. In the second phase, sandhi-analysis is conducted on the base text in a separate file. During this process, each word in the base text file is assigned an `xml:id` which serves as the key to co-ordinate the two files. The commentary is tagged in a third file which references words in the base text by referring to the `xml:id` of the word as the value of the `corresp` attribute.

3 Details of data preparation

Considerable effort is required to obtain quality character data of the base text and commentary. First of all, when creating a digital edition, it is necessary to consider the copyright issue. To

simply reproduce a printed edition in the digital medium without adding significant knowledge without permission of the holder of a valid copyright would violate intellectual property rights. However, no intellectual property rights would be violated by making a digital copy of an edition in the public domain. Nor would intellectual property rights be violated by producing a new edition with significant new knowledge added. The editions we considered are all in the public domain. We chose to tag Kale's (1922) edition. Although character data of the *Raghuvaṁśa* is easily found on the Web, character data of Mallinātha's commentary is not. To type the commentary would be time-consuming and expensive compared with deriving character data by optical character recognition (OCR). However, the OCR output available at `archive.org` produced several years ago was junk. Since then OCR software that produces relatively good output of a Devanāgarī printed page has become available on line. Initially we processed the scanned images of Kale (1922) at `https://ocr.sanskritdictionary.com`. Yet proof-reading this output revealed that it needed extensive correction which significantly affected the pace of the project. We then discovered a few digital editions of the *Sañjīvanī*, one of which with Devanāgarī character data is available at `https://www.ebharatisampat.in`. We transcoded into SLP1 the first five sargas of this edition, the source of which is not identified, and edited it to conform with Kale's (1922) edition. Thereby we obtained creditable digital character data of the *Sañjīvanī* efficiently.

3.1 The saṁhitā file

We refer to the file containing the continuous base text without sandhi analysis as the saṁhitā file. In this file, each verse of the base text is analysed metrically. Although the generic metrical pattern of each canto is primarily the same, identifying the meter-type in cantos composed in the Upajāti meter also recognized the specific subtype of each verse. This information is not documented in any previous digital or printed edition of the *Raghuvaṁśa*.

Figure 1 shows a sample of the mark-up of a verse, and Table 1 shows the metrical patterns employed in the first five cantos.

Figure 1

TEI encoding of metrical patterns in Kālidāsa's Raghuvaṁśa

```
<lg type='fdDiH' xml:id='s2.v1'>
 <l>
  <seg n='a'>aTa prajAnAmaDipaH praBAte</seg>
  <seg n='b'>jAyApratigrAhitaganDamAlyAm</seg>
  <pc>.</pc>
 </l>
 <l>
  <seg n='c'>vanAya pItapratibadDavatsAM</seg>
  <seg n='d'>yaSoDano Denumfzermumoca</seg>
  <pc>..2.1..</pc>
 </l>
</lg>
```

Table 1
Meter identification

sarga	verses	meters	subtype
1	95	अनुष्टुभ्, प्रहर्षिणी	None
2	75	उपजाति, मालिनी	ऋद्धि, भद्रा, उपेन्द्रवज्रा, इन्द्रवज्रा, जाया, माला, हंसी, कीर्ति, माया, सिद्धि, बाला, रामा, वाणी, शाला, बुद्धि, हैमी, आर्द्रा
3	70	वंशस्थ, हरिणी	None
4	88	अनुष्टुभ्, प्रहर्षिणी	None
5	76	उपजाति, वसन्ततिलका	मालिनी, पुष्पिताग्रा, हैमी, उपेन्द्रवज्रा, माया, इन्द्रवज्रा, जाया, बाला, कीर्ति, वाणी, भद्रा, शाला, आर्द्रा, ऋद्धि, बुद्धि, सिद्धि, रामा

3.2 The padapāṭha file

In the padapāṭha file, sandhi is analysed between words in the saṁhitā file, and each pada is set in a `w` element and provided with a unique `xml:id` attribute. Figure 2 shows an example of the markup in the padapāṭha file. Table 2 shows the word count in each of the first five cantos.

Figure 2
Word count in the first five sargas of Kālidāsa's Raghuvaṁśa

```
<lg n='1' xml:id='s1.v1'>
 <l n='1' xml:id='s1.v1.s1'>
  <w n='1' xml:id='s1.v1.w1'>vAgarTAviva</w>
  <w n='2' xml:id='s1.v1.w2'>saMpfktO</w>
  <w n='3' xml:id='s1.v1.w3'>vAgarTapratipattaye</w>
  <pc>.</pc>
  <w n='4' xml:id='s1.v1.w4'>jagataH</w>
  <w n='5' xml:id='s1.v1.w5'>pitarO</w>
  <w n='6' xml:id='s1.v1.w6'>vande</w>
  <w n='7' xml:id='s1.v1.w7'>pArvatIparameSvarO
   <m n='1' xml:id='s1.v1.w7.m1'>pArvatI</m>
   <m n='2' xml:id='s1.v1.w7.m2'>parameSvara</m>
  </w>
   <pc>..</pc>
 </l>
</lg>
```

Generally we do not need to analyse any compound into its constituents in the `padapāṭha` file. However, Mallinātha often dissolves compounds, and provides synonyms, derivations, references and other comments, not only on the compound as a whole but also on its constituents. In order to facilitate precise reference to the constituents commented upon in the commentary file, we split compounds into their constituents where the commentary analyzes the compound and provides a synonym, lexical reference or derivation for any of its constituents. Each compound

Table 2
Pada count

sarga	verses	padas
1	95	969
2	75	1039
3	70	1029
4	88	857
5	76	1060

constituent is set in an `m` element and provided with an `xml:id` attribute. Table 3 shows the count of the compound padas that are so analyzed into their constituents in the first three cantos, the percentage of padas analyzed, and number of segmented constituents.

Table 3
Tally of padas, analyzed compounds, and their constituents in cantos 1 and 2

sarga	verses	padas	analyzed padas	percentage of padas analyzed	segmented constitutents
1	95	969	88	9.08%	215
2	75	1039	99	9.53%	250
3	70	1029	71	7.00%	178

3.3 Coordinating analysis in the commentary file with the padapāṭha file

The commentary file consists of the digital character data of the *Sañjīvanī* in SLP1 encoding analyzed in accordance with the TEI guidelines in the same manner as Ajotikar and Scharf (2023) analyzed Bhaṭṭi's *Rāvaṇavadha*. Analytic elements `ab`, `seg`, `w`, etc. are coordinated with specific words, compound constituents, and morphemes in the padapāṭha file by referring to the `xml:id` of the `w` element containing the word, or the `m` element containing the compound constituent or morpheme in the padapāṭha file as the value of the `corresp` attribute of the analytic element in the commentary file. In the following subsections, we illustrate our method of tagging the constituents of compounds analyzed by Mallinātha, and how we supply information assumed as obvious by Mallinātha in order to make its relationship to the text commented upon explicit.

3.3.1 Compound constituent analysis

When Mallinātha supplies any information regarding a constituent of a compound, technically there is no word in the base text to which it corresponds. As mentioned above, in such cases, we split the compound pada in the padapāṭha file into its constituents and set each constituent in an `m` element with a unique `xml:id`. This `xml:id` occurs as a value of the `corresp` attribute in the analytic element in the commentary file. Below we provide four examples of the analysis of words into constituents. The first illustrates a simple compound analyzed into its constituents. The second analyzes a taddhita derivate into its constituent base and affix. The third and fourth present the analysis of compounds whose first and second constituent respectively undergo alteration in the compound.

1. In the very first verse, the word *pārvatīparameśvarau* (s1.v1.w7) is a dvandva compound. Mallinātha comments on each of the constituents of the compound as follows:

   पर्वतस्यापत्यं स्त्री पार्वती। 'तस्यापत्यम्' इत्यण्। 'टिड्ढाणञ्' इत्यादिना ङीप्। पार्वती च परमेश्वरश्च पार्वतीपरमेश्वरौ। परमशब्दः सर्वोत्तमत्वद्योतनार्थः।

The word *pārvatī* designates the female offspring of the mountain (*parvata*). The suffix *aṇ* is added (by *A.* 4.1.83 in the meaning of 'his offspring') by *A.* 4.1.92, after the word *parvata*. The feminine suffix *ṅīp* is added by *A.* 4.1.15 *ṭiḍḍhāṇañ* etc. (The dvandva compound) *pārvatīparameśvarau* is dissolved as *pārvatī* and *parameśvara*. The constituent *parama* (of the latter) denotes 'best of all'.

Here Mallināṭha gives the derivation of the word *pārvatī* and supplies the sūtras for the same. Then he shows the paraphrase of the compound (*vigrahavākya*). In order to associate all the information about each constituent in the commentary file with its constituent in the padapāṭha file, we split the compound into two constituents in the `padapAWa` file as shown below:

```
<w n='7' xml:id='s1.v1.w7'>pArvatIparameSvarO
 <m n='1' xml:id='s1.v1.w7.m1'>pArvatI</m>
 <m n='2' xml:id='s1.v1.w7.m2'>parameSvara</m>
</w>
```

2. Mallināṭha analyzes other derivates besides compounds into their constituents; these include *taddhita* derivates. For example, Mallināṭha comments on the word *prasraviṇīm* (s2.v61.w17) as follows:

   प्रस्रवः क्षीरस्रावो ऽस्ति यस्याः सा तां प्रस्रविणीं

   The word *prasraviṇīm* means one who has a flowing forth, i.e. flow of milk.

   Here Mallināṭha does not merely supply the paraphrase but provides a synonym of the first constituent *prasrava*. To the word *prasrava* is added the possessive suffix *in*. Its first component is the word *prasrava*, and the second component is the affix *in*. In the padapāṭha file, we treat such cases similarly to the way we treat compounds, setting each component of the taddhita derivate in an `m` element. For the affix, we use the bare affix without any markers or supplementary sounds added for pronunciation. Thus here the second component is *in*, not *ini* as provided by Pāṇini *A.* 5.2.115 with *i* added for the sake of pronunciation.

```
<w n=`17' xml:id=`s2.v61.w17'>prasraviRIm
 <m n=`1' xml:id=`s2.v61.w17.m1'>prasrava</m>
 <m n=`2' xml:id=`s2.v61.w17.m2'>in</m>
</w>
```

3. In the process of dividing a compound into its constituents, there are a few instances of altered forms where a decision needs to be made regarding how to designate the base of the altered form. For example, for *anāsthā* (s2.v57.w15), which is a nañ-tatpuruṣa compound, Mallināṭha provides the synonymous nañ-tatpuruṣa *anapekṣā*, and adds a lexical reference for the second constituent *āsthā* of the original compound. He states:

   शरीरेष्वनास्था खल्वनपेक्षैव। 'आस्था त्वालम्बनास्थानयत्नापेक्षासु कथ्यते' इति विश्वः।

   Indifference towards bodies is indeed detachment. Viśva provides four synonyms of the word *āsthā*: *ālambana*, *āsthāna*, *yatna*, and *apekṣā*.

   In order to associate the lexical reference of the second constituent of *anāsthā*, we dissolve the compound into two constituents. However the first constituent *an* is not a free morpheme. It is an altered form of the negative particle *na*. Rather than using the altered form of the particle *an* before vowel-initial subsequent constituents, and *a* before consonant-initial subsequent constituents, we uniformly use the original unaltered particle *na*. In other words, we do not use the constituent of a compound as it occurs in it; rather we use the base word as lexicalized in the padapāṭha file, as shown below.

```
<w n=`15' xml:id=`s2.v57.w15'>anAsTA
 <m n=`1' xml:id=`s2.v57.w15.m1'>na</m>
 <m n=`2' xml:id=`s2.v57.w15.m2'>AsTA</m>
</w>
```

4. The previous example was of a compound in which the first constituent was altered. Here we provide an example of a compound in which it appears that the final constituent is altered. The word *kuṇḍodhnī* (s1.v84.w3) has two constituents: *kuṇḍa* and *udhas*. Mallinātha comments on the compound as follows:

   कुण्डमिवोध आपीनं यस्याः सा कुण्डोध्नी। 'ऊधस्तु क्लीबमापीनम्' इत्यमरः। 'ऊधसो ऽनङ्' इत्यनङादेशः। 'बहुव्रीहेरूधसो ङीष्' इति ङीष्॥

   The one whose breast, i.e. udder, is like a pitcher. Amara states that the word *ūdhas* is neuter and its synonym is *āpīna*. The final sound of the word *ūdhas* is replaced by *anaṅ*, by *A.* 5.4.131 *ūdhaso 'naṅ*, in a *bahuvrīhi* compound of which the final constituent is *ūdhas*. The feminine suffix *ṅīṣ* is provided after the bahuvrīhi compound by *A.* 4.1.25 *bahuvrīher ūdhaso ṅīṣ*.

   The provision of the samāsānta suffix *anaṅ* at the end of the compound whose final constituent is *ūdhas* results in the altered morpheme *ūdhan*, which after the addition of the feminine suffix *ṅīṣ* appears as *ūdhnī*. This morpheme is not a free morpheme, never occurs as an independent word, and is not lexicalized. Hence, we restore the second compound constituent to the base form *ūdhas* in the padapāṭha file and set it in an `m` element with an `xml:id` as shown below. In order to associate the passage of the lexical resource in the commentary file with the constituent in the padapāṭha file, this `xml:id` is referred to as the value of the `corresp` attribute of the `seg` element in which Mallinātha makes reference to the passage in Amara's lexicon that provides the synonyms for the constituent.

```
<w n=`3' xml:id=`s1.v84.w3'>kuRqoDnI
 <m n=`1' xml:id=`s1.v84.w3.m1'>kuRqa</m>
 <m n=`2' xml:id=`s1.v84.w3.m2'>UDas</m>
</w>
```

3.3.2 Inferred padas in the commentary file

One would expected that scholastic commentaries like Mallinātha's *Sañjīvanī* comment on almost every word of the base text. However Mallinātha does not always cite the exact pada of the base text. Even without explicitly repeating a compound or other word analyzed, he provides a compound-analysis or derivation, and just thereby refers to the original reading implicitly. In such cases, we add the whole pada analyzed to the commentary file in a `w` element provided with the attribute-value pair `type=`inferred'`. This procedure is an adaptation in XML of the square brackets in which the word is inserted in the printed edition by the editor. Kale (1922) provides the inferred pada in square brackets before or at the end of the compound analysis. We invariably add the inferred pada at the beginning of the compound analysis. Sometimes Mallinātha omits a pada of the base text in his commentary. Usually it is a particle like *ca* or *eva* that is omitted. We deal with such an omission by including the particle at the appropriate place in the commentary file in a `w` element provided with the attribute-value pair `type=`supplied'`. Just one such instance occurs in the first canto; none in the second or third. Table 4 shows the distribution of padas inferred in the first three cantos.

Table 4
Pada count

sarga	verse	padas	inferred padas	percentage
1	95	969	91	9.39%
2	75	1039	89	8.57%
3	70	1029	81	7.87%

4 Issues in tagging the commentary

4.1 Enhancement of attribute-value pairs

Although Ajotikar and Scharf (2023) standardized the procedure of tagging kāvya text with commentary, that sample study was based upon a small extent of text. The project of marking up the first five cantos of the *Raghuvaṁśa* with the *Sañjīvanī* extends the tagging of such texts with commentary significantly. While our procedures remain predominantly the same, the additional data required us to enhance the set of attribute-value pairs in order to record previously unencountered information in the commentary. The discussion hereafter focuses on reporting these enhancements with representative examples.

4.1.1 Synonymity

Ajotikar and Scharf (2023, p. 130) state

> Where Mallinātha supplies a word with a synonym, the synonym is put in a `seg` element as a sister to the `w` element containing the word. The `seg` element is supplied with the attribute-value pair `type=`synonym'` and a `corresp` attribute with the value of the `xml:id` of the `w` element.

While observing his procedure in the current project, we noticed that Mallinātha treats compounds differently from simple words. When he provides a synonym of a simple word, he almost always supplies a reference to a thesaurus. For example, when he comments on the word *uḍupena* (s1.v2.w11) he supplies a reference to the *Amarakośa*.

> उडुपेन प्लवेन। 'उडुपं तु प्लवः कोलः' इत्यमरः।
> The word *uḍupena* 'by raft' means *plavena* 'by boat'. Amara records (two synonyms) *plava* and *kola* for *uḍupa*.

However, when Mallinātha provides a synonym of a compound of only occasional occurence, he makes no reference to a lexical resource because no lexical treatise includes the compound. In this case Mallinātha creates a synonym by replacing each constituent in the compound with a synonym. For example, he composes the synonym *īśvarakiṁkarasya* for the word *devānucarasya* (s2.v52.w2). It is observable that he formulated the synonym by replacing the constituents *deva* and *anucara* with the synonyms *īśvara* and *kiṁkara* respectively. In both kinds of cases, whether he provides a lexical reference or not, we set the synonym in a `seg` element with the attribute-value pair `type=`synonym'`.

Different from the above cases of synonymity, we discovered three types of cases where Mallinātha glosses a word with a word that is not simply a synonym. In one case the gloss is a morphological alternate. In many cases, the gloss is a semantic explanation. These are of two types: an explanatory word that implicitly explicates grammatically known syntactico-semantic relations, and an explanatory word that merely explicates contextual semantics. We illustrate these types of glosses with an example of each below.

1. There is only one instance where Mallinātha provides an alternate morphological form in a gloss:

   जरसा जरया विना (s1.v23.w9).

   The word *jarā* has the optional form *jaras* when a vowel-initial nominal termination follows. Here Kālidāsa uses the optional form *jarasā*. Mallinātha comments on it by supplying the other alternate form *jarayā*. This cannot be treated as a synonym since an optional morphological form of the same nominal base is not a synonym. We cover this instance by formulating a new attribute-value pair for the `seg` element `type=`alternate'`.
2. In the many instances where Mallinātha provides a gloss that is a semantic explanation, we assign the value `def` (definition) to the attribute `type` in the `seg` element. Our first example illustrates the type of gloss by which he provides an explanatory word that implicitly

explicates grammatically known syntactico-semantic relations. For the word *upahāsyatām* (s1.v3.w4) 'the state of being to be ridiculed', Mallinātha provides the gloss *upahāsaviṣayatām* 'being an object of ridicule'. Here the explanatory word implicitly refers to grammatical rules that explicate the semantic structure of the glossed word. By stating *upahāsa-viṣaya*, Mallinātha implicitly refers to one of the senses in which the krtya affixes are provided after a root, namely, to denote the direct object of the action of the root, as stated by *A.* 3.4.70 *tayor eva krtyaktakhalārthāḥ*.

Our second example illustrates the type of gloss by which Mallinātha provides an explanatory word that merely explicates contextual semantics. He usually adds the phrase *ityarthaḥ* after such contextually explanatory glosses. The compound *ānākarathavartmanām* (s1.v5.w6) 'the track of whose chariots reach heaven' is explained by providing the compound *indrasahacāriṇām* 'companions of Indra' concluded with the phrase *ityarthaḥ*. The statement that those whose chariots reach heaven are companions of Indra clearly does not provide a synonym because the gloss is contextually specific. Rather it states simply in ordinary terms what Kālidāsa poetically phrases in a term that could have different meanings in different contexts. The gloss does not denote the primary meaning of the glossed word.

4.1.2 Syntax

As is commonly done in scholastic commentaries, Mallinātha provides derivations of most of the words in the base text while commenting on a verse. Often these derivations are explicit grammatical comments relevant to construing the syntactic relation of the word to others in the sentence. In such cases, the `seg` element in which the statement is set is provided with the attribute-value pair `type=`syntax'`. Such notes may concern the syntax of the main verb of a clause, the participatory role a denoted object plays in an action (*kāraka*) denoted by a nominal, or any other relation. The following examples illustrate these points.

1. Regarding the verb *mucye* 'I am released' (s1.v72.w2), Mallinātha comments मुक्तो भवामि। कर्मणि लट्। 'I am freed. The *laṭ* affix is provided to denote the direct object (*karman*).' The first sentence provides a paraphrase of the finite verb form of the root *muc* 'free, liberate' with the past passive participle of the same root plus a finite verb of the root *bhū* 'be'. The second sentence makes an explicit grammatical statement concerning syntax, namely, that the *l*-affix *laṭ* denotes the direct object (*karman*). This explicit statement clarifies that the word *mucye* occurs in the present tense and is passive. The point is significant for the syntactic structure of the sentence.
2. Mallinātha glosses the word *locanābhyām* (s2.v19.w11) with the word *karaṇaiḥ*. Thereby he indicates that the word *locana* is the instrument of the action of drinking denoted by the root *pā* 'drink' in the finite verb form *papau* in the verse.
3. While explaining the word *ikṣvākūṇām* (s1.v72.w8), Mallinātha provides the reference to the grammatical rule *A.* 2.3.50 *ṣaṣṭhī śeṣe* which explains the use of its vibhakti stating, इक्ष्वाकूणामिति शेषे षष्ठी। 'In the word *ikṣvākūṇām*, the sixth-triplet termination occurs in a remaining sense.'

4.1.3 Constituent

1. Generally Mallinātha reveals the constituents of a compound when he gives the paraphrase (*vigrahavākya*) but does not repeat those constituents outside the paraphrase. Occasionally, however, he mentions the constituents separately and comments on them in detail. For example, while commenting on the word *gūḍhākāreṅgitasya* (s1.v20.w3), Mallinātha describes the technical meaning of the words *ākāra* and *iṅgita*. His comments are as follows:

   शोकहर्षादिसूचको भ्रुकुटीमुखरागादिराकारः। इङ्गितं चेष्टितं हृदयगतविकारो वा। 'इङ्गितं हृद्गतो भावो बहिराकार आकृतिः' इति सञ्जनः।

   The word *ākāra* 'shape' means facial expression by knitting the eyebrows etc.

which indicates (the emotions) sorrow, joy etc. The word *iṅgita* 'indication' means a gesture or change of feeling. Sajjana says *iṅgita* means a feeling in the heart, and *ākāra* means an external expression.

Here Mallinātha not only provides definitions of both constituents *ākāra* and *iṅgita* but also supplies a lexical reference. In order to coordinate this information with the base text we mention the constituents of the compound in separate `m` elements in the padapāṭha file. In the commentary file, we introduce the new value `constituent` of the `type` attribute in the `seg` element that has the `corresp` attribute with the value of the `xml:id` of the constituent in the padapāṭha file. Figure 3 shows this segment marked up in the padapāṭha file.

Figure 3
TEI encoding of Mallinātha's commentary on the compound constituents ākāra *and* iṅgita

```
     <w n='2' type='inferred' xml:id='s1.v20.w3'>gUQAkAreNgitasya</w>
     <seg type='def' corresp='s1.v20.w3.m2'>SokaharzAdisUcako BrukuwImuKarAgAdiH
      <seg type='constituent' corresp='s1.v20.w3.m3'>AkAraH.</seg></seg>
     <seg type='constituent' corresp='s1.v20.w3.m3'>iNgitaM</seg>
     <seg type='synonym' corresp='s1.v20.w3.m3'>cezwitam</seg>
     <seg type='synonym' corresp='s1.v20.w3.m3'>hfdayagatavikAraH</seg> vA.
     <seg type='lexicon'>
      <quote>iNgitaM hfdgato BAvo bahirAkAra AkftiH</quote>
       <note>
        <bibl corresp='xml/common/listbiblfile.xml#aufrecht.cc'>Aufrecht
<biblScope>1.687</biblScope></bibl>.
        <bibl corresp='xml/common/listbiblfile.xml#ncc'>NCC
<biblScope>37.160.a.1</biblScope></bibl>.
       </note>
       iti sajjanaH.</seg>
```

4.1.4 Paraphrase

There are many instances where Mallinātha elucidates a word in the base text with a paraphrase. Such paraphrases are different from synonyms, which are single words, from vigrahavākyas, which are paraphrases of compounds, and from definitions. In order to deal with this kind of paraphrase we introduce a new value `paraphrase` of the `type` attribute of the `seg` element as shown in the following examples:

1. Mallinātha explains the word *vanāya* (s2.v1.w6) (dative of *vana* 'forest') by paraphrasing it *vanaṁ gantum* 'in order to go to the forest'. He supplies the infinitive of the verb *gam* 'go' and changes the dative to an accusative to show that the forest is the direct object (*karman*) of the action of going. This paraphrase precisely reformulates the original expression in accordance with the ellipsis which Pāṇini deals with in *A.* 2.3.14 *kriyārthopapadasya ca karmaṇi sthāninaḥ.*

   ```
   <w n=`10' xml:id=`s2.v1.w6'>vanAya</w>
   <seg type=`paraphrase' corresp=`s2.v1.w6'>vanaM gantum</seg>
   ```

   Exactly similar is the paraphrase *yajñaṁ kartuṁ* 'in order to perform a yajña' for *yajñāya* 'for a yajña' (s1.v26.w4).
2. In the compound *phalānumeyā* (s1.v20.w5) 'to be inferred by results', Mallinātha explains the constituent *anumeya* 'to be inferred' by paraphrasing it *anumātuṁ yogyā* 'fit to be inferred' which clarifies that the sense of the krtya affix *yat* occurs in the sense of being fit or suitable (*arha*) in accordance with *A.* 3.3.169 *arhe krtyatrcaś ca.* We mark this in the commentary file as follows:

```
<w n=`10' xml:id=`s1.v20.w5'>PalAnumeyA</w>
…
<seg type=`paraphrase' corresp=`s1.v20.w5.m2'>anumAtuM yogyA</seg>
```

3. Similarly, Mallinātha explains the word *vivakṣuḥ* 'desirous to speak' (s2.v43.w9) by paraphrasing it *vaktum icchuḥ* thereby showing that the word is formed from the desiderative root *vivakṣa* derived from the primary root *vac*. The commentary file marks this paraphrase as follows:

```
<w n=`9' xml:id=`s2.v43.w9'>vivakzuH</w>
<seg type=`paraphrase' corresp=`s2.v43.w9'>vaktum icCuH</seg>
```

4.1.5 Words supplied by the commentator

Occasionally Mallinātha supplies a word to fill out the syntax of the verse upon which he comments, usually but not always by putting *iti śeṣaḥ* after the word. In such cases, we set the word in a `w` element with the `type` attribute value `added`, and an `n` attribute given the value 101 or above. Several instances occur in the first three sargas (1.37, 1.68, 1.69, 1.83; 2.56; 3.6, 3.68).

4.2 Quotations

Ajotikar and Scharf (2023) included the procedure of tagging a quotation in a commentary: add the reference in a `note` element which has the text identifier in a `bibl` element which in turn has the location within the text in a `biblScope` element. However they did not identify the quotations in their sample data. In this project, we tried to identify the source of every quotation Mallinātha cites. We can categorise these quotations as follows:

1. quotations from lexical resources and metrical literature,
2. quotations from the grammatical resources which are mostly the sūtras quoted from the *Aṣṭādhyāyī*,
3. other quotations which are not from lexical and grammatical literature.

We refer to two editions of the *Raghuvaṁśa*, one by Nandargikar (1982) and one by Pandit (1874). Nandargikar provides several appendices including a list of the works and authors quoted by Mallinātha, and a list of unidentified quotations. Pandit includes among his appendices a list of works and authors referred to by Mallinātha, and a list of unidentified quotations referred to by Mallinātha. However, neither editor made any effort to verify the original source to which Mallinātha attributes a quotation. We did. In our attempt to verify the original source of the quotation, we found that there are many instances where the source to which the quotation is attributed does not in fact contain the passage. Closer examination revealed that some of the quotations occur in a different text from the one reported by Mallinātha. We traced the correct source of some of these, and in addition some of the unidentified quotes. We provide the correct source in our XML commentary file. The works which are not extant today but were available to Mallinātha are provided with the reference to their entry in the *New Catalogus Catalogorum* (*NCC*). Yet many quotations which we could not locate remain unidentified.

4.2.1 Quotations from lexical resources

Mallinātha refers heavily to lexical resources. He refers not only to the base text of such resources but also occasionally to commentaries on them. As mentioned, we carefully attempted to verify every quotation from a lexicon against the original source; doing so revealed numerous errors in the printed editions. We categorize the quotations under three headings: *verified*, *corrected*, and *unidentified*.

Verified: These quotations are found in the original texts cited. The following is a list of the lexical resources (*kośas*) to which Mallinātha refers in the first three cantos:

1. *Anekārthasaṅgraha* of Hemacandra
2. *Anekārthasamucchaya* of Śāśvata
3. *Abhidhānaratnamālā* of Halāyudha
4. *Amarakośa* of Amara
5. *Kṣirasṭāmin*'s commentary on the Amarakośa
6. *Ekākṣararatnamālā* of Mādhava
7. *Nānārthārṇavasaṅkṣepa* of Keśava
8. *Vaijayantikosha* of Yādavaprakāśa
9. *Viśvaprakāśa* of Maheśvara
10. A lexicographical work of Sajjana no longer extant

For the last item, the lexical work composed by Sajjana referred to by Mallinātha (s1.v2.w11, s1.v20.w3), we provide the reference to the *NCC* (Dash 2015, 160a, first entry).

Corrected: When we discover the correct source of a quote which the printed edition of the *Sañjīvanī* incorrectly attributes to a different text, we provide the correct source. We do so by setting the incorrect source in the `sic` element, and the correct source in the `corr` element. In the first three cantos, there are eleven instances where we corrected the source. For example, while commenting on the word *dākṣiṇyarūḍhena*, for the constituent *dākṣiṇya*, Mallinātha provides the synonym *paracchandānuvartanam* 'following another's will'. Subsequently he states,

> 'दक्षिणः सरलोदारपरच्छन्दानुवर्तिषु' इति शाश्वतः।
> According to Śāśvata, the word *dakṣiṇa* means 'straight-forward,' 'generous' and 'behavior as per another's will'.

Despite his claim that this quotation is in Śāśvata's *Anekārthasamucchaya*, it is not found in it. It is actually found in the *Viśvaprakāśakośa*. We correct it as follows:

```
<seg type=`lexicon' corresp=`s1.v31.w2.m1'>dakziRaH
 saralodAraparacCandAnuvartizu
 <quote>dakziRaH saralodAraparacCandAnuvartizu</quote>
 …iti <sic>SASvataH</sic><corr>viSvaH</corr>
</seg>
```

The following is the list of all corrected instances in the first three cantos:

1. शाल (s1.v13.w3.m1): यादव corrected to शाश्वत
2. नेमीनाम् (s1.v17.13.m1): यादव corrected to हलायुध
3. वेलाः (s1.v30.w2.m1): विश्व corrected to शाश्वत
4. दाक्षिण्यं (s1.v31.w2.m1): शाश्वत corrected to विश्व
5. सुरभिः (s2.v3.w7): विश्व corrected to शाश्वत
6. दावम् (s2.v8.w6): यादव corrected to शाश्वत
7. रागः (s2.v15.w8.m1): शाश्वत corrected to विश्व
8. रत्नाम् (s2.v30.w7.m1): यादव corrected to विश्व
9. कङ्क (s2.v31.w5.m3): विश्व corrected to एकाक्षररत्नमाला
10. अदह्यत (s2.v32.w7): यादव corrected to धनञ्जय (दशरूपक)
11. गुणाग्र्यवर्तिना (s3.v27.w10): विश्व corrected to शब्दरत्नसमन्वयः

Concerning 10 (s2.v32.w7), the reading in the edition, namely, 'अधिक्षेपाद्यसहनं तेजः प्राणात्ययेष्वपि ऽ इति यादवः, suggests that the author of the cited passage is Yādava, the author of the *Vaijayantīkośa*. However, the passage occurs in the *Daśarūpaka* (Parab 1941, p. 42), not in any lexical resource.

Concerning 9 (s2.v31.w5.m3), Mallinātha mistakenly claims that the verse line कङ्कः पक्षिविशेषे स्याद्वृक्षाकारे युधिष्ठिरे occurs in Viśva's *Viśvaprakāśakośa*. However, after careful examination, the quote is found in the *Ekākṣararatnamālā*. The first edition of the text by

Ramnikvijay (2019) identifies its title as *Ekākṣaraśabdamālā* and its author as Haritālarājāmātyamādhava. These are not found as such in the *NCC*. Instead, the *NCC*, vol. 3, p.59b (the 17th entry in the column), edited by Raghavan (1967), reads

एकाक्षररत्नमाला by Mādhavācārya, son of Māyana, minister of Harihara.

Thus the *NCC* provides the title *Ekākṣararatnamālā* and identifies the author as Mādhavācārya, son of Māyana, minister of Harihara. What is given as the name of the author in the printed edition, Haritālarājāmātyamādhava, means 'Mādhava, a minister of Haritāla', which agrees to some extent with the information found in *NCC*. In this case, we provide references to both the printed edition and the *NCC* in the comentary file.

Unidentified: There are just two instances in which the quotation is unidentified:

1. At the end of verse 2.35, Mallināṭha quotes the verse,
   पृथिवी सलिलं तेजो वायुराकाशमेव च।
   सूर्याचन्द्रमसौ सोमयाजी चेत्यष्टमूर्तयः॥
   and claims that it is quoted from the *Vaijayantīkośa*. However the quote is not found in the *Vaijayantīkośa*.
2. भरणो पोषणो भर्म इति हैमः। (s3.v12.w6): When Mallināṭha comments on the word *garbhabharmaṇi*, he provides a lexical reference for the constituent *bharman* claiming that it is from Hemacandra's *Anekārthasaṅgraha*, but it is not found.

There is one interesting case where Mallināṭha refers to a variant reading in the *Amarakośa* which is noted by Liṅgayasūrin in his *Amarapadavivrti* (Ramanathan 1978, p. 37). While commenting on the word *atrasta* (s1.v21.w3), he quotes the *Amarakośa* regarding the constituent *trasta* as त्रस्तो भीरुभीरुकभीलुकाः. In his citation, the quote contains the word *trasta* instead of *trasnu* which is found in the passage in most of the printed editions of the *Amarakośa* (अधीरे कातरत्रस्नौ भीरुभीरुकभीलुकाः). Liṅgayasūrin in his *Amarapadavivrti* registers a variant on this verse saying, त्रस्तो इति वा पाठः. Apparently Mallināṭha knew this line in the *Amarakośa* as अधीरे कातरत्रस्तौ भीरुभीरुकभीलुकाः. This is very significant from the point of view of tracing Mallināṭha's sources.[1]

4.2.2 Quotations from treatises on meter

The verses of each canto are predominantly composed in a single metrical pattern; however, the pattern changes towards the end of the canto. Whenever the metrical pattern changes, Mallināṭha cites a characterization of the metrical pattern (*lakṣaṇa*) but never tells the source of the citation. In the first three cantos there are five occurrences where he identifies the meter and then cites its *lakṣaṇa* (s1.v95, s2.v1, s2.v75, s3.v1, s3.v70). The sources of these citations are not identified in the printed editions. We identify them. Four of them occur in Kedārabhaṭṭa's *Vrttaratnākara*, and one occurs in Gaṅgādāsa's *Chandomañjarī*.

4.2.3 Quotations from grammatical texts

Mallināṭha refers to the sūtras of the *Aṣṭādhyāyī* in most of the derivations and syntactic comments he provides. Occasionally he quotes from commentaries like the *Vyākaraṇamahābhāṣya* and *Kāśikāvrtti*. We have verified all of these grammatical quotations except one. Mallināṭha attributes the quotation, न केवलं श्रूयमाणैव क्रिया निमित्तं करणभावस्य। अपि तर्हि गम्यमानापि (s2.v34), to a work called the *Nyāsoddyota* which is not extant. For this quote we provide the reference to the entry for the text in the *NCC*.

One derivation is interesting because it reveals a discrepancy among grammatical texts, particularly regarding lists (*gaṇa*s). The derivation concerns the word *vārdhaka* (s1.v8.w5). Mallināṭha derives it by adding the affix *vuñ* after the base *vrddha* by *A.* 5.1.133 *dvandvamanojñādibhyaś*

[1]Mallināṭha was from Andhra. Hence it is not surprising that he knew the variant noted by a southern commentator.

ca, stating, द्वन्द्वमनोज्ञादिभ्यश्च। इति वुञ्प्रत्ययः। Interestingly the word *vrddha* is not included in the *gaṇa* that begins with the word *manojña* as per the list in the *Kāśikāvrtti* on *A.* 5.1.133. If the reading in the *Kāśikāvrtti* is followed, then the word *vārdhaka* cannot be derived. However the *Gaṇaratnamahodadhi* does include the word *vrddha* in the gaṇa *manojñādi* (verse 409). We must suppose that either Mallinātha knew a reading in the *Kāśikāvrtti* which included the word *vrddha* in the gaṇa *manojñādi*, or he refered to the *Gaṇaratnamahodadhi*.

4.2.4 Micellaneous quotations

Apart from lexical and grammatical texts, Mallinātha quotes verses or passages from many other texts. As with the lexical quotations, we attempted to verify the source of these and categorize them according to whether they have been verified, corrected, or remain unidentified. We provide the reference in the critical edition of the text if available. The number of unidentified micellaneous quotations is greater than the unidentified lexical quotations. The list of these other texts quoted in the first three cantos is as follows:

1. *Agnipurāṇa*
2. *Aṣṭāṅgasaṅgraha*
3. *Āśvalāyanagrhyasūtra*
4. *Kīrātarjunīya*
5. *Gautamdharmasūtra*
6. *Cāṇakyanīti*
7. *Taittirīyasaṁhitā*
8. *Nītisāra*
9. *Parāśarasmrti*
10. *Brhajjātaka*
11. *Manusmrti*
12. *Mandāramaranda*
13. *Mahābhārata*
14. *Mānavagrhyasūtra*
15. *Meghadūta*
16. *Yājñavalkyasmrti*
17. *Rāmāyaṇa*
18. *Śaṅkhasmrti*
19. *Saṅgītamakaranda*
20. *Skandapurāṇa*
21. *Harivaṁśa*

4.2.5 Corrected Quotations

Below is the list of the quotations for which we provide the corrected source.

1. Commenting on the word *maunam* in s1.v22, Mallinātha states, यथाह कामन्दकः 'नान्योपतापि वचनं मौनं व्रतचरिष्णुता' इति। The cited verse line does not occur in the *Kāmandakanītisāra* but is found in the *Agnipurāṇa* (239.22).
2. At the end of his commentary on s1.v85, Mallinātha quotes the verse,

   आग्नेयं भस्मना स्नानमवगाह्यं तु वारुणम्।
   आपोहिष्ठेति च ब्राह्मं वायव्यं गोरजः स्मृतम्॥

   and claims that this verse occurs in the *Manusmrti* (उक्तं च मनुना). The verse does not occur in the *Manusmrti* but is found in the *Skandapurāṇa*.
3. Similarly, commenting on s2.v75 Mallinātha states,

   'यथेयं पृथिवी मह्युत्ताना गर्भमादधे।
   एवं त्वं गर्भमाधेहि दशमे मासि सूतवे॥'
   इत्याश्वलायनानां सीमन्तमन्त्रे स्त्रीव्यापारधारण आधानशब्दप्रयोगदर्शनादिति।

The verse does not occur in the *Āśvalāyanagrhyasūtra* but is actually found in the *Mānavagrhyasūtra*.

4.2.6 Unidentified Quotations

Below is the list of the many unidentified quotations in the first three cantos.

1. आयुक्तकेभ्यश्चौरेभ्यः परेभ्यो राजवल्लभात्। पृथिवीपतिलोभाच्च नराणां पञ्चधा मतम्॥ (s1.v60)
2. ऋणं देवस्य यागेन ऋषीणां दानकर्मणा। संतत्या पितृलोकानां शोधयित्वा परिव्रजेत्॥ (s1.v71)
3. कामं पितरं प्रोषितवन्तं पुत्राः प्रत्याधावन्ति एवमेत (एवम् ह वैत-) मग्नयः प्रत्याधावन्ति सशकलान्दारूनिवाहरन् (s1.v49)
4. चतुर्थऽनवलोभनम् इत्याश्वलायनः। (s3.v10.w6)
5. छिन्द्याद्बाहुमपि दुष्टमात्मनः (s1.v28)
6. त्रिंशद्भागात्मकं लग्नम् (s3.v13)
7. नासाकण्ठमुरस्तालुजिह्वादन्तांश्च संस्पृशन्। षड्भ्यः संजायते यस्मात्तस्मात्षड्ज इति स्मृतः (s1.v39)
8. निर्वाणोत्थानशयनानि त्रीणि गजकर्माणि (s1.v71)
9. पृथिवी सलिलं तेजो वायुराकाशमेव च। सूर्याचन्द्रमसौ सोमयाजी चेत्यष्टमूर्तयः॥ (s2.v35)
10. प्रतिपाद्यमहिम्ना च प्रबन्धो हि महत्तरः (s1.v2)
11. प्रोष्यागच्छतामाहिताग्नीनामग्नयः प्रत्युद्यान्ति (s1.v49)
12. मांसलश्च (s3.v34.w3)
13. मृदं गां दैवतं विप्रं घृतं मधु चतुष्पथम्। प्रदक्षिणानि कुर्वीत विज्ञातांश्च वनस्पतीन्॥ (s1.v76.w6)
14. रविणास्तमयो योगो वियोगस्तूदयो भवेत् (s3.v13)
15. राजा त्वर्थान्समाहृत्य कुर्यादिन्द्रमहोत्सवम्। प्रीणितो मेघवाहस्तु महतीं वृष्टिमावहेत्॥ (s1.v26)
16. विषादश्चेतसो भङ्ग उपायाभावनाशयोः (s.v40.w1)
17. शक्तानां भूषणं क्षमा (s1.v22.w3)
18. शुभदो मो भूमिमयः (s1.v1)
19. स खलु पुत्रार्थिभिरुपास्यते (s1.v35.w2)
20. समिद्धेऽग्नावाहुतीर्जुहोति (s1.v53.w1)

Usually Mallinātha cites these passages with a vague reference to their source such as *iti vacanāt* or *iti smrteḥ*. Yet in a couple of instances, he attributes the quotation to a particular author. However these do not occur in the texts by those authors. On s1.v22, Mallinātha claims that the verse शक्तानां भूषणं क्षमा is by *Cāṇakya*. However we did not find it in the *Cāṇakyanīti*. Similarly the quote निर्वाणोत्थानशयनानि त्रीणि गजकर्माणि (s1.v71) is claimed to be by Pālakāpya, but we did not find it in his *Gajacikitsā*. Lastly मांसलश्च (s3.v34.w3) is claimed to be by a vrttikāra. We did not find it in the *Kāśikāvrtti* and do not know to which vrttikāra he refers. Since these have not been located in the presumed texts, we label them as well `unidentified`.

5 Conclusion

In the present project of making a TEI digital edition of Mallinātha's commentary *Sañjīvanī* on Kālidāsa's *Raghuvaṁśa*, we encountered material that called for expansion of the procedures for creating XML editions of Sanskrit texts containing commentaries in accordance with the Text Encoding Initiative (TEI) guidelines. To accommodate this material we created the following additional values of the `seg` element's `type` attribute: (1) `alternate`, (2) `def`, (3) `paraphrase`, (4) `syntax`, and (5) `constituent`. The first three expand the categorization of glosses. Where previously all glosses were categorized as synonyms, we now distinguish optional morphological forms, semantic explanations, and paraphrases from synonyms by the following values of the `seg` element's `type` attribute respectively: `alternate`, `def`, and `paraphrase`. Item (4) adds a refinement of the derivation category to distinguish an explicit grammatical comment relevant to construing syntactic relations from derivational material exclusively concerned with the morphology and semantics of the word. Item (5) allows one to relate comments regarding a compound constituent separated from the compound paraphrase to the compound constituent in the padapāṭha file.

In the present project, we also contributed substantially to the constitution and analysis of the text. We verified and corrected the sources of citations, and listed those that remain unidentified, added reference to the *NCC* for non-extant works, and added sources of citations where Mallināth doesn't reveal them. The process of undertaking the higher-level encoding of a text reveals hidden textual problems with Sanskrit texts. The process of analyzing a commentary and precisely categorizing extents of it forces one to notice lacunae in prior work on the text which in turn gives one the opportunity to make significant philological contributions. The contemporary shift of the principal medium of knowledge transmission from the printed word to the digital medium is transforming philology into digital philology which is the future form of textual research.

6 Acknowledgements

We would like to thank the Center of Policy Research and Governance (CPRG) Indian Knowledge Systems (IKS) for providing a Research Grant to Ketaki Kaduskar to participate in this project. We also thank Maithili Kulkarni, Somaiyya Institute of Dharma Studies, Mumbai, who helped to verify lexical resources.

References

Ajotikar, Tanuja P. and Peter M. Scharf (2023). "Development of a TEI standard for digital Sanskrit texts containing commentaries: A pilot study of bhaṭṭi's Rāvaṇavadha with Mallināth a's commentary on the first canto". In: *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*. Ed. by Amba Kulkarni and Oliver Hellwig. Canberra, Australia (Online mode): Association for Computational Linguistics, pp. 128–145. URL: https://aclanthology.org/2023.wsc-csdh.9.

Dash, Siniruddha, ed. (2015). *New Catalogus Catalogorum. An alphabetical register of Sanskrit and allied works and authors saṁyamanśreṇivicāra–samasyāhomaprayoga*. Madras: University of Madras.

Kale, Moreshvara Ramachandra (1922). *The Raghuvaṁśa of Kālidāsa. with the commentary the Sañjīvanī of Mallināth a Canto 1–10 edited with a literal translation into English, copious notes in Sanskrit and English and various readings*. Bombay: Gopal Narayan and co. Third revised edition.

Nandargikar, Gopal Raghunath (1982). *The Raghuvaṁśa of Kālidāsa*. Delhi: Motilal Banarasidas. Fifth edition.

Pandit, Shankar P. (1874). *The Raghuvaṁśa of Kālidāsa. with the commentary the Sañjīvanī of Mallināth a edited with notes, part III, cantos 14–19*. Bombay: Government central book depot.

Parab, Kashinath Pandurang, ed. (1941). *Śrīdhanañjayaviracitaṁ Daśarūpakam. Dhanikakrtayāvalokayā vyākhyayā bhāratīyanāṭyaśāstragatadaśanirūpaṇena ca sametam*. Mumbai: Ninrnaysagar Press.

Raghavan, V. (1967). *New Catalogus Catalogorum. An alphabetical register of Sanskrit and allied works and authors*. Ed. by Raja C. Kuhnan. Madras University Sanskrit Series 18. Madras: University of Madras.

Ramanathan, A. A. (1978). *Amarakośa. with the commentary the unpublished south indian commentaries Amaapadavivrti of Liṅgayasūrin Amaapadapārijāta of Mallināth a Amaapadavivaraṇa of Appayārya critically edited with introduction*. Madras: The Adyar library and research centre.

Ramnikvijay, Pannyas, ed. (2019). *Ekākṣaraśabdamālā*. Ahmedabad: Śrutajñāna saṁskārapīṭha.

Inter Sentential Discourse Relations

Saee Vaze
Department of Sanskrit Studies,
University of Hyderabad
20hsph03@uohyd.ac.in

Amba Kulkarni
Department of Sanskrit Studies,
University of Hyderabad
ambakulkarni@uohyd.ac.in

Abstract

In this paper, we present a tagging scheme for inter-sentential discourse relations that is developed, based on the insights from Indian Grammatical Tradition. We rely on the three factors - *ākāṅkṣā*, *yogyatā* and *sannidhi* to decide the connectivity between two consecutive sentences. Various clues are identified that bind the two consecutive sentences. A tag set is presented based on the explicit discourse markers. Implementation of discourse level analysis based on the explicit discourse markers is tested on the *Śrimadbhagvadgītā* corpus. It is observed that some discourse markers are ambiguous and it is not trivial to develop a disambiguation module for such markers.

1 Introduction

The term discourse analysis has gained a lot of attention in the recent past. It typically refers to a linguistic unit that goes beyond a sentence.[1] Thus, the discourse analysis goes beyond the scope of sentence boundaries and looks at the text as a unit of language. In understanding the meaning of a discourse, both the linguistic and non-linguistic factors contribute. The linguistic factors include coherence markers while non-linguistic background includes, speaker-listener dynamics, situationality etc. In Natural Language Processing, the core interest is in producing computer-processable models of discourse at different levels such as sentence, paragraph, text, etc. Varied work has been done on the topic already on different levels and languages.

From the theoretical point of view the work by Indologists and Sanskrit scholars in the field of discourse analysis is very rich and valuable. Scharf and Hock (2015) provides an exhaustive bibliography of works in the field of general discourse and formal syntax. However, there is very little work from the perspective of computational linguistics with regards to Sanskrit language. There are several efforts in the West especially in the field of computational linguistics. Treatment of cohesion by Halliday and Hasan (1976) attempts to look at the text as a linguistic phenomenon. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) was the first effort towards establishing the discourse structure in the form of a graph, by connecting two adjacent units by a discourse relation. Another seminal effort was made by the team lead by Arvind Joshi in the project Penn Discourse Tree-Bank (Prasad et al., 2006) which focuses on the structure of arguments and how a connective enables a certain discourse relation, implicit or explicit. The discourse tree-banks were created from a huge data from Wall Street Journal (Mann and Thompson, 1988) following the RST framework. The other discourse databanks include Linguistic Discourse Model (Polanyi, 2008), and the Discourse Graphbank (Wolf and Gibson, 2005). The Discourse-Lexicalized Tree Adjoining Grammar (Webber and Joshi, 1998) was developed following the Penn Discourse Treebank guidelines. With the emergence of such computational guidelines and resources for several languages discourse tagged datasets were developed for languages other than English such as Czech (Mladová et al., 2008), Chinese (Jiang et al., 2018) and Turkish (Zeyrek et al., 2010) to name a few. Similar efforts were made

[1] https://www.merriam-webster.com/dictionary/discourse

for some Indian Languages resulting in Hindi Discourse Relation Bank (Umangi et al., 2009), Bangla RST Discourse Treebank (Das and Stede, 2018), Annotated Tamil Corpus (Rachakonda and Sharma, 2011), Annotations of Connectives and Arguments in Malayalam (Kumari and Devi, 2016) etc.

Regarding Sanskrit, in the recent past Kulkarni and Das (2012) presented a brief summary of the various sets of discourse relations found in the Indian grammatical tradition(IGT). They have also shown the usefulness of these relations by developing a Finite State Automaton to tag the texts in *Mahābhāṣya* following the cues available. Recently Terdalkar and Bhattacharya (2019) developed a Question Answering system for special domains. Apart from these there is not much work in the area of discourse analysis in Sanskrit.

In what follows we brief our approach to discourse analysis in Sanskrit following IGT followed by a review of earlier work on Discourse analysis in Sanskrit. In Section 3 we present various clues that mark the *ākāṅkṣā* between the two consecutive sentences. We identify the explicit discourse markers in Sanskrit that connect two consecutive sentences. This is followed by a discussion on the implementation, challenges and evaluation.

2 Discourse Analysis in Sanskrit

The computational models such as RST and Penn Discourse may be tried for Sanskrit as well. However there are three considerations why we decided to follow the IGT. The first and foremost concerns with the rich linguistic tradition of India. The theories of *śābdabodha* that deal with the process of understanding texts are almost as old or albeit a little older than *Pāṇini*'s grammar. *Jaimini* in his composition of *Mīmāṁsāsūtra* not only provided his interpretations of the vedas, but also provided a glimpse of what principles he followed in interpreting the texts. Further *Śabara* elaborates these principles including the ones which *Jaimini* merely indicated. The seeds sown by these *Māmāṁsaka*s further grew into various guidelines to decide the coherence between the textual segments. The *Naiyāyika*s and the *Vaiyākaraṇa*s also followed the *Mīmāṁsaka*s resulting into various sets of coherence relations proposed by them for describing the coherence between the various segments of the texts. These relations cover a wide range of units starting from the sentences to paragraphs to chapters to texts to discipline. Depending on the style of the texts, and the type of unit the text belongs to, different annotation schemes were proposed by different schools. A detailed description of this is available in Kulkarni and Das (2012).

The second consideration is that these discourse relations are also used by the commentators while commenting upon important texts, or editors who used them as subtitles providing some hints towards understanding the cohesion, and sources for coherence markers. For example the Nirṇaya-sāgara edition of the *Mahābhāṣya* has subtitles which show the logical structure of the discourse.

The third consideration is the following. *Mīmāṁsaka*s discuss three factors viz. *ākāṅkṣā*, *yogyatā* and *sannidhi* as important factors for the verbal cognition. These factors play an important role throughout the process of verbal cognition - not limited just to the sentential analysis - but extending to the understanding of the complete text. Thus these three factors can be considered to be guidelines for identifying the clues and connecting the segments of the texts accordingly. Having developed a sentential parser (Kulkarni, 2019) based on the theories of *śābdabodha*, where all these factors were used for sentential analysis, it gave us a confidence that these factors can be further extended for discourse analysis as well.

Hence we decided to base our approach to discourse analysis following the IGT.

2.1 Earlier work and its limitations

The relations in the tag-set proposed by Krishnamacharyulu (2009) contain inter-sentential relations as well. These inter-sentential relations are marked by some connectives which are indeclinables. Some of these connectives occur in pairs. Kulkarni and Das (2012) had proposed a tagging scheme for them. Each of these connectives takes two arguments. Following logicians convention, these arguments are named by the general terms *anuyogika*[2] (combining) and *pratiyogī* (having a counter part). So, if C is the connective connecting two sentences S1 and S2 then the general structure is represented as in Figure 1.

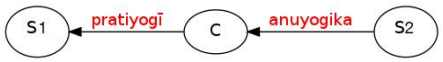


Figure 1: Discourse structure with single connective

When there are two parallel connectives C1 and C2 connecting S1 and S2 then the relation between them is represented as in Figure 2.

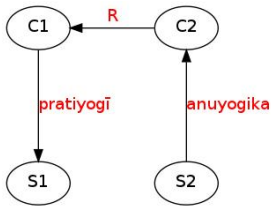


Figure 2: Discourse structure with paired connectives

Here R binds C1 and C2. The relation of the connectives with the sentence is through the main verbs. The sentences are further parsed as dependency trees. In case of paired connectives, the usage allows using either of them or both of them in a sentence. When only one of them is used in a sentence then the structure in Figure 2 collapses to Figure 1.

There were two problems with this work. The first one was related to the names of the relations. The two terms *anuyogika* and *pratiyogī* are very general that they do not convey the semantics of the relation between the two sentences. Secondly, we also noticed that there were several other indeclinables which the authors had missed. So in the next section, we look at the relations between two consecutive sentences, marked by explicit markers, and provide a semantic interpretation of that relationship. We also enlist all possible indeclinables that can mark the relations between the consecutive sentences.

3 Inter-Sentential Discourse Relations

The inter-sentential relations are identified with the help of *ākāṅkṣā*, *yogyatā* and *sannidhi*. Sometimes the fourth factor *tātparya* is also considered to an essential factor in the process of *śābdabodha*. This factor is more relevant from the word sense disambiguation point of view, and also for choosing the level of signification of the word. Hence we focus only on the first three factors. We define the basic elementary units between which we establish the relations. Then we look at the clues that guide us in proposing a relation. Then comes the mutual compatibility between the elementary units which confirms the proposed relation. We assume that the elementary units are consecutive ones, ensuring that the third factor sannidhi is taken care of.

[2]S2 is the anuyogi. So if the arrowhead is pointing towards S2 the name of the relation would have been anuyogi. In this diagram, the arrowhead is pointing towards C, and hence the name of the relation is inverse of anuyogi, i.e. anuyogika.

3.1 Elementary Unit

Elementary Unit refers to the basic elements between which the relation is to marked. We take *vākya* (sentence) as a unit, where *vākya* is defined as '*eka tiṅ vākyam*'. That is a group of related words with one finite verbal form is termed as a sentence. Further, participles, especially those with *kta*, *ktavatu* and the *kṛtya* suffixes such as *anīyar* etc. are typically used as if they are finite verbs (Speijer, 1886) (in section 9). Hence group of related words with such forms, without any finite verbal form, are also considered to be a sentence. (Others not specially mentioned in the list of sentences with non-finite exception, such as *satī-saptamī*, *tumun* etc. would be considered as a single unit and would not fall under the domain of inter-sentential discourse relations.)

With this definition, the following group of words

prātaḥkāle rāmaḥ śālāṁ gacchati. tatra pāṭhaṁ paṭhati. krīḍati ca. sāyaṁkāle gṛhaṁ āgacchati.

consists of four sentences, as delimited with the full stops. Now consider the following sentence:

Sanskrit : *yadi tvam icchasi tarhi ahaṁ tava gṛhaṁ āgacchāmi iti rāmaḥ śāmaṁ vadati.*
Gloss : if you wish{2p,sg,pres}, then I your house{loc} come{1p,sg,fut} so Rama{nom} Shyama{acc} say{3p,sg,pres}
Eng: "If you wish I will come to your house" says Rama to Shyama.

Following the definition of *eka-tiṅ vākyaṁ*, here there are three sentences viz.

1. *tvam icchasi*,
2. *ahaṁ tava gṛhaṁ āgacchāmi*, and
3. *rāmaḥ śyāmaṁ vadati.*

connected by three connectives *yadi*, *tarhi* and *iti*. The words *yadi* and *tarhi* are the pair connectives, and both these connectives have an expectancy of two sentences. The third connector *iti* is a marker for the *karman* (*vākya-karma-dyotakaḥ*) which is in sentential form. Thus now the complex sentence formed by joining the two sentences with the pair of connectives *yadi-tarhi* acts as a *karma* for the verb *vad*.

3.2 *Ākāṁkṣā*

Literally *ākāṅkṣā* is the desire on the part of a listener to know (*jñātum icchā*). In the case of understanding a sentence, the desire is to know how the words in a sentence are connected to each other producing a unified meaning. This *ākāṁkṣā* is expressed in language through various means. As is mentioned in Kulkarni (2019), there are different linguistic clues that mark the expectancies in a sentence, such as

- suffix, as in the case of '*vanaṁ gacchati*', [forest{nom} go{3p,sg,pres}] the suffix '*am*' marks the *karmatva* and thus has an expectancy of a transitive verb to connect with,
- position, as in the case of a sentence starting with the word '*api*', there is an expectancy of a sentence such as '*tvam gacchasi*',[you{nom} go{3p,sg,pres}] so that complete expression expresses a question,
- indeclinables such as '*na*' which have an expectancy of a verbal form to connect to, and finally
- the underlying verbal root in a verbal form has an expectancy for various *kāraka*s.

Similarly the sentence level connections are expressed through various means such as indeclinables, relative position of sentences, semantics associated with the verbal roots, and so on.

- The indeclinables such as *yadi*, *tarhi*, *yataḥ*, *tataḥ*, *atha*, *yathā-tathā*, *yadyapi*, etc. provide a cue that the consecutive sentences (or group of sentences) are related. For example, the two sentences

  Sanskrit: *rāmaḥ paṭhati. tathāpi parīkṣāyām uttīrṇaḥ na bhavati.*
  Gloss: Rama{nom} study{3p,sg,pres} then exam{loc} pass{nom} neg be{3p,sg,pres}
  English: Rama studies. (Even) then he does not pass the exam.

  are connected to each other showing the failure to get the desired results even after performing the necessary task.

- The relative position of the sentences in a conservation also provides us a clue about the temporal sequence between the events associated with the verbal forms.

  Sanskrit: *rāmaḥ prātaḥkāle uttiṣhati. snānam karoti. dugham pītvā śālām gacchati.*
  Gloss: Rama{nom} morning{loc} wake{3p,sg,pres}. Bath{acc} do{3p,sg,pres}. Milk{nom} drink{geund} school{acc} go{3p,sg,pres}
  English: Rama wakes up in the morning. Takes a bath. Goes to school, after drinking milk.

  Here we notice that there is a temporal sequence, and thus the order in which the activities happened is marked in the position of these sentences. There is no lexical unit which marks such relation.

- The use of pronouns connect the sentences when the anaphora resolution is made.

  Sanskrit: *rāmaḥ śālām gacchati. saḥ tatra pāṭham paṭhati.*
  Gloss: Rama{nom} school{acc} go{3p,sg,pres}. He{nom} there lesson{acc} study{3p,sg,pres}
  English: Rama goes to school. There he studies a lesson.

  Here the use of the pronoun '*saḥ*' for Rāma by the speaker, needs to be resolved by the listener. Only then the listener can understand the conversation.

- The semantics associated with verbs also raise certain expectancies. For example look at the two *śloka*s the first one and the seventh one from *Saṅkṣepa-rāmāyaṇam*. The first one viz.

  tapassvādhyāyaniratam tapasvī vāgvidām varam
  nāradam paripapraccha vālmīkirmunipuṅgavam

  has the verbal form *paripapraccha*(asked) which has an expectancy of an answer. This expectancy is fulfilled by the verbal form *abravīt* (said) from the seventh *śloka*, viz.

  ... śrūyatām iti āmantrya prahṛṣṭaḥ vākyam abravīt.

In this paper we focus only on the lexical units that express the sentential expectancies.

3.3 *Yogyatā*

Some indeclinables such as '*atha*', can be used to denote conjunction as well as succeeding action. Similarly words such as *yasmāt-tasmāt* or *yena-tena* can represent the *kāraka* relations such as *apādāna* or instrument, alternately these words may also denote a *hetuḥ* - a cause-effect relation. In order to decide the appropriate role in the context, we need to look at the context - both linguistic as well as non-linguistic, identify the linguistic and ontological factors that can help in the disambiguation, and so on. We look at one particle '*hi*' (See Sec 5.1) which is ambiguous between a causal marker and a definiteness marker, and show how difficult it is to address the problem of ambiguity.

4 Discourse Relation tagging and clues

Below we present the list of discourse markers we have come across so far (and also implemented), along with the relation(s) they express and the tagging with an example sentence.

Relation	**Markers**
Succeeding (*anantarakālaḥ*)	*tataḥ, anantara, atha*
Simultaneity (*samānakālaḥ*)	*yadā-tadā*
Co-location (*samānādhikaraṇaḥ*)	*yatra-tatra*
Similarity (*sādṛśyam*)	*yathā-tathā*
Cause-Effect (*kārya-kāraṇam*)	*yataḥ-tataḥ, ataḥ, yasmāt, tasṃāt, yena, tena, hi*
Possiblity (*āvaśyakatā-pariṇāmaḥ*)	*yadi-tarhi, iti, cet*
Hindrance in cause-effect (*vyabhicāraḥ*)	*yadyapi-tathapi, cedapi, athāpi, tarhyapi*
Antithesis (*virodhaḥ*)	*parantu, kintu*
Conjunction (*samucccayaḥ*)	*ca, api, cāpi, athaca, athāpi, evañca*
Disjunction (*anyataraḥ*)	*vā, uta, yadvā, athavā, utāpi, utasvit*

Table 1: List of discourse relations and markers

1. **Succeeding (*anantarakālaḥ*) :**
   Here the relation of succeding activity to the preceeding is marked. The presence of indeclinables such as *atha*, *tataḥ*, etc. trigger the relation of the current sentence with the previous one. The activity denoted by the current sentence is marked as the succeding activity for the activity denoted by the previous sentence. See Figure 3.

   Sanskrit : *ahaṃ śṛṇomi atha likhāmi.*
   Gloss : I{nom} listen {3p,sg,pres} then write{3p,sg,pres}
   English : I listen, then I write.

   Before moving to the next relation, we highlight the salient features of the discourse graph representation.

   - The relations between two sentences is through a link between the head (*mukhya viśeṣya*) of the two sentences.
   - The arrow head is with the node that satisfies the property named by the edge label.
   - The direction of the arrow, unlike in dependency trees, does not denote the dependency, or the head and the sub-ordinate.
   - In order to distinguish the discourse relations from the intra-sentential relations, discourse relations are marked with double line.

   Thus in Figure 3 the verbs from the two sentences viz. *śṛṇomi* and *likhāmi* are related by the relation of *anantarakālaḥ*, and the marker for this relation is the word *atha*.

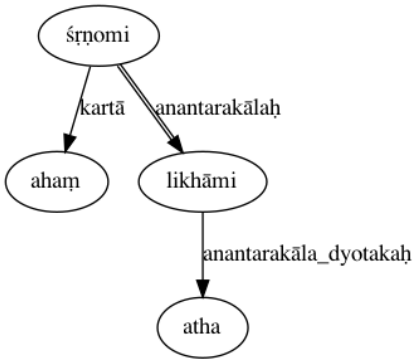


Figure 3: anantarakālaḥ relation

2. **Simultaneity (*samānakālaḥ*) :**
In Sanskrit, there are two ways of expressing the simultaneity. One is the use of present participles (*kṛt* suffixes - *śatṛ* and *śānac*) which are part of intra-sentential relations. The second is the use of indeclinable pair *yadā-tadā*. In this case, the verbs in finite form from both the sentences are connected with a relation *samānakālaḥ*. The words *yadā* and *tadā* mark a relation of *kālādhikaraṇam* (time locative) (See Figure 4).
Here is an example:

Sanskrit : *yadā bharataḥ mārge gacchati tadā saḥ devālayam paśyati.*
Gloss: when Bharata{nom} path{loc} go{3p,sg,pres} then he{nom} temple{acc} see{3p,sg,pres}
English : On his way Bharata sees a temple.

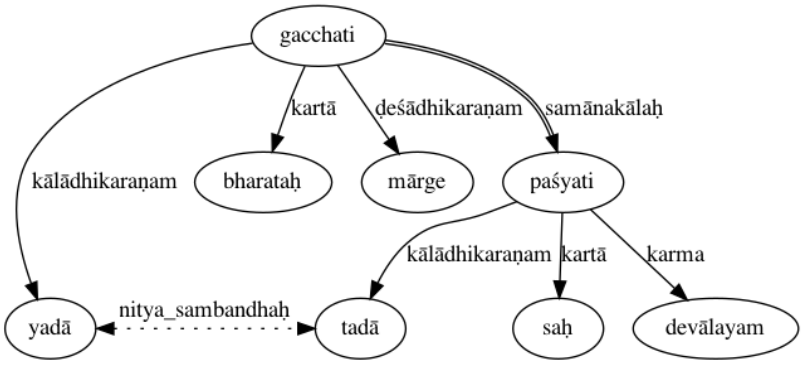


Figure 4: samānakālaḥ relation

3. **Co-location (*samānādhikaraṇaḥ*) :**
This relation indicates that the activities indicated by the two consecutive sentences are performed at the same location. This relation is marked by the pair of indeclinables *yatra-tatra*. The consecutive sentences use these two words denoting the *deśādhikaraṇam* (place locative), or only one of them is used in one sentence (See Figure 5).

Sanskrit : *yatra nāryaḥ tu pūjyante tatra devatāḥ ramante.*
Gloss : where women{nom} emph_marker worship{3p,pl,pres} there Gods{nom} reside{3p,pl,pres}

English : Where women are worshiped there reside the Gods.

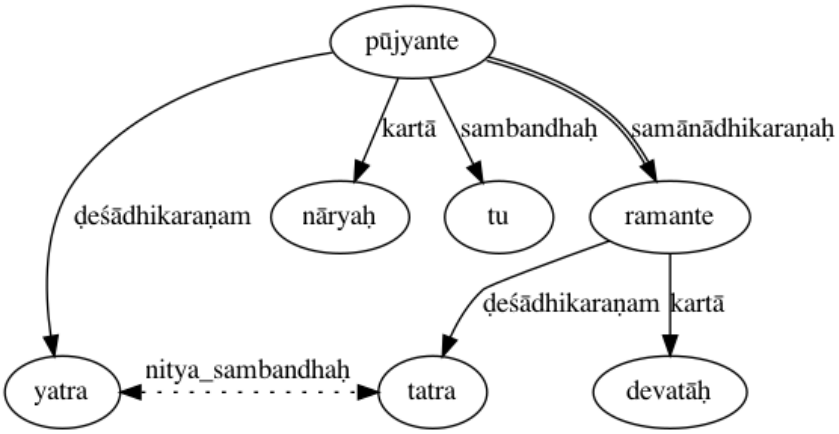


Figure 5: samānādhikaraṇaḥ relation

4. **Similarity (*sādṛśyam*) :**
This relation is of similarity. The similarity is between the two activities expressed through two consecutive sentences (See Figure 6). The example is :

Sanskrit : *janāni karmāṇi yathā kurvanti tathā te phalam prāpnuvanti.*
Gloss : people{nom} deeds{acc} as do{3p,pl,pres} so they{nom} fruit{acc} reap{3p,pl,pres}
English: People reap the fruits as per their deeds.

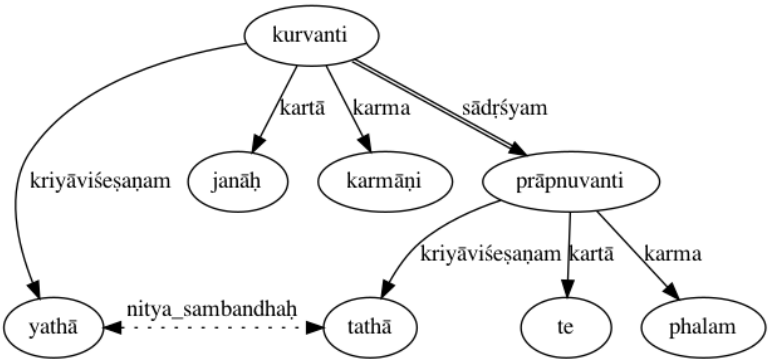


Figure 6: sādṛśyam relation

5. **Cause-Effect (*kārya-kāraṇa*) :**
When there is certainty about the cause, or the event expressing the cause has already taken place or there is a certainty that a certain event expressing the cause is going to happen, to express the certainty of the result following the cause, such constructions are used. This is a dichotomous relation where the sentence expressing the cause is marked with *yataḥ* indicating the reason/cause (*kāraṇa-dyotakaḥ*) and the sentence expressing the result is marked with the connective *tataḥ* which is an indicator of the result (*kārya-dyotakaḥ*). It is possible that only one connective among the two is used. Still it gives the same

meaning viz. cause-effect relation (See Figure 7). An example of this type of construction is:

Sanskrit : *yataḥ avarṣat tataḥ mayūraḥ nṛtyati.*
Gloss : because rain{3p,sg,pp} therefore peacock{nom} dance{3p,sg,pres}
English : Because it has rained, (therefore) peacock is dancing.

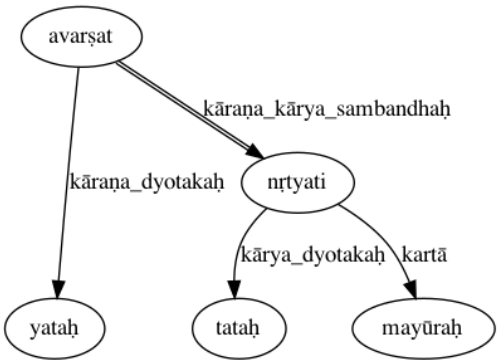


Figure 7: kārya-kāraṇa relation

6. **Conditional (*āvaśyakatā-pariṇāma*)** :
In slight contrast with the previous one, there are conditional sentences where there is no certainty of the event indicating the cause. To indicate the possibility of the resulting event provided the event corresponding to the cause takes place, such constructions are used. These sentences are marked with *āvaśyakatā-pariṇāma-sambandhaḥ*, which is a dichotomous relation where, the marker *yadi* indicates the necessity (*āvaśyakatā-dyotakaḥ*) and the marker *tarhi* indicates the result (*pariṇāma-dyotakaḥ*). The markers are used either in pair or individually as well (See Figure 8). An example of this type is:

Sanksrit : *yadi paṭhasi tarhi uttīrṇaḥ bhaviṣyasi.*
Gloss : if read{2p,sg,pres} then pass{nom} be{2p,sg,fut}
English : If you study (then) you will pass.

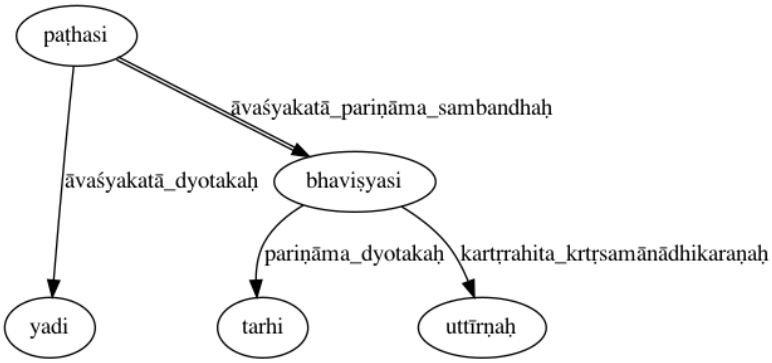


Figure 8: āvaśyakatā-pariṇāma relation

7. **Anomaly (*vyabhicāra*) :**
This is an exception or violation in naturally occurring cause-effect relationship. The pair of words *yadyapi* (even though) and *tathāpi* (even then) are the markers that trigger such

relations. We mark *vyabhicāra-sambandhaḥ* between the two finite verbs indicating the actions. The marker *yadyapi* is tagged as *kāraṇa dyotakaḥ* and *tathāpi* as *kārya dyotakaḥ*. The exceptions may be of two types. In one case even though the cause is present, the expected result is absent, and in the second case the result is present even though the desired cause is missing, thus violating the concomitance between the cause and effect (See Figures 10 and 9). The two types of examples are:

Sanskrit : *yadyapi varṣā bhavati tathāpi mayūraḥ na nṛtyati.*
Gloss: even-if rain{3p,sg,pres} happen even-then peacock{nom} neg dance{3p,sg,pres}
English: Even if it rains, even-then peacock does not dance.

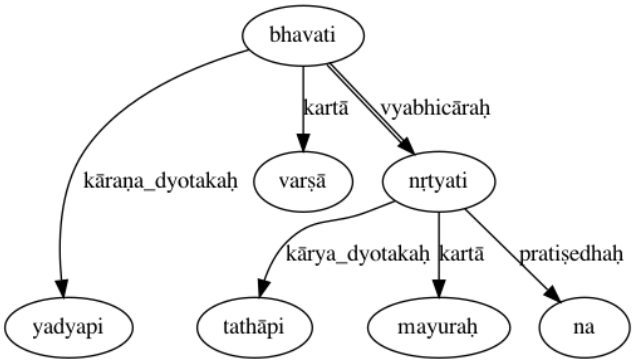


Figure 9: vyabhicāra relation type 1

Sanskrit: *yadyapi saḥ vaidyaḥ na asti tathāpi saḥ cikītsāṃ jānāti.*
Gloss: Even-if he{nom} doctor{nom} neg be{3p,sg,pres} even-then he{nom} cure{acc} know{3p,sg,pres}
Eng: Even if he is not the doctor, even-then he knows the cure.

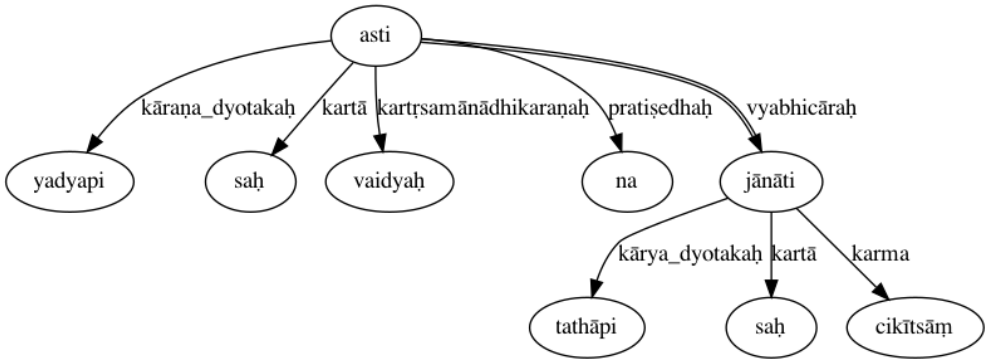


Figure 10: vyabhicāra relation type 2

8. **Antithesis (*virodhaḥ*) :**
Antithesis shows contradiction or opposition. It is typically marked by particles such as *parantu* and *kintu* (See Figure : 11). For example :

Sanskrit : *gajendraḥ tīvram prayatnam akarot parantu nakra-grahāt na muktaḥ.*
Gloss : Elephant{nom} hard effort{acc} do{3p,sg,past} but crocodile-grip{abl} no free{1p,sg,ppp}

English : Elephant tried hard but couldn't escape from the crocodile-grip.

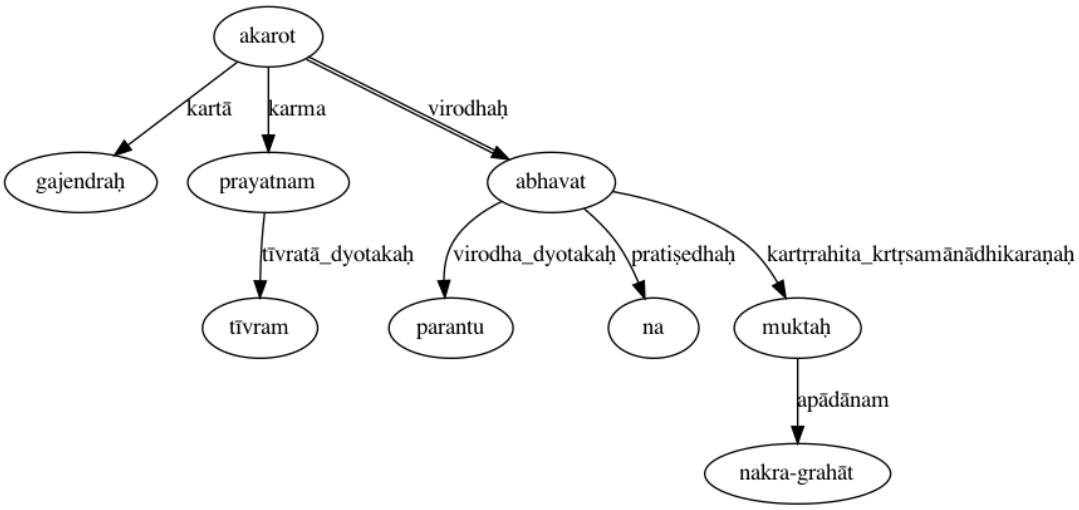


Figure 11: virodhaḥ relation

9. **Conjunction (*samuccayaḥ*):**
The conjuncts conjoined by the conjunctions are marked by this relation (See Fig 12). The example is :

Sanskrit : *Bhikṣām aṭa api ca gāṃ ānaya.*
Gloss : alms{dat} roam{2p,sg,imp} also and cow{acc} bring{2p,sg,imp}
English : Roam around for alms and bring the cow.

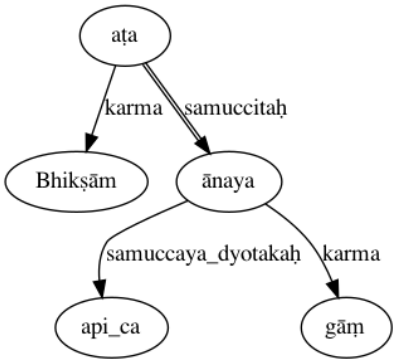


Figure 12: samuccayaḥ relation

The detailed discussion on the representatoon of conjuncts with various categories from computational point of view is presented in (Kulkarni and Panchal, 2019).

10. **Disjunction (*anyataraḥ*) :**
The disjuncts conjoined with disjunctive markers are marked by this relation (See Fig 13). The Example is :

Sanskrit : *sītā śvaḥ kāryakrame gāsyati athavā nartsyati.*
Gloss : Sita{nom} tomorrow program{loc} sing{3p,sg,fut} or dance{3p,sg,fut}

English : Sita will sing in tomorrow's program or will dance.

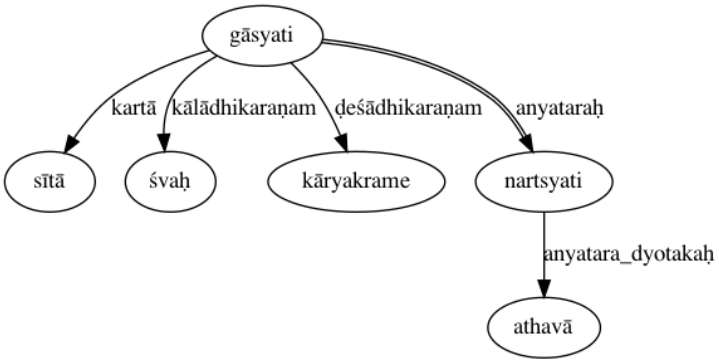


Figure 13: anyataraḥ relation

5 Implementation, Challenges and Evaluation

We selected *ŚrimadBhagvadGītā*(SBG) as a corpus for testing. There are 700 verses where each verse may consist of more than one sentence or more than one verse may consitute one sentence. The criterion for deciding the boundary of a sentence is '*eka ṭiṇ vākyam*'[3] and '*sākāṇkṣam cet vibhāge syāt*'.[4] Thus any group of words having one and only one finite verb and where every word is related to at least one other word from the group is termed as a sentence. Detecting sentence boundary has been earlier addressed by Hellwig (2016). It has been observed that a considerable number of errors are produced by the sentence boundary algorithm when sentences are smaller in length and especially without the use of copula. Since we were interested in the performance of the inter-sentential discourse analyser, to avoid cascading effect on the errors, we decided to manually annotate the sentence boundaries.

All the sentences having the inter-sentential markers were extracted from SBG. The distribution of various markers and the relations they mark is shown in Table 2. In the case of unambiguous markers, it was trivial to mark the relations automatically. The ambiguous markers fall under two categories. The first one has pronouns that are ambiguous due to the ambiguity of the case markers such as ablative and instrumental case suffixes which can mark a *kāraka* relation such as *apādānam* or *karaṇam* and a non-*kāraka* relation like *hetuḥ*. In this case unless the pronoun reference is identified, it is difficult to decide what relation is marks. The second category has ambiguous indeclinables such as *hi* and *atha*. In order to understand the problems in disambiguation, as a case study we looked at all the instances of *hi* in SBG. We describe below our observations.

5.1 Disambiguation of '*hi*'

The Sankrit-Hindi Apte's dictionary has the following four different senses of the word *hi*.

1. isaliye ki, kyoṁki
2. nissandeha, niścaya hi
3. udāharaṇasvarupa

[3]a group of words with one finite verb is a sentence.

[4]When a group of words is split into two parts and a word from one group has an expectancy for the word from the other group, all the words together from one sentence.

Markers	**Relation**	Frequency
tasmāt	*kārya-kāraṇa-sambandhaḥ*	21
tasmāt	*apādānam*	2
yasmāt	*kārya-kāraṇa-sambandhaḥ*	1
yasmāt	*apādānam*	1
tataḥ	*kārya-kāraṇa-sambandhaḥ*	8
tataḥ	*apādānam*	8
tataḥ	*anantarakālaḥ*	7
yataḥ	*kārya-kāraṇa-sambandhaḥ*	2
yataḥ	*apādānam*	2
ataḥ	*kārya-kāraṇa-sambandhaḥ*	2
ataḥ	*apādānam*	2
tena	*karaṇam*	7
yena	*karaṇam*	8
hi	*kārya-kāraṇa-sambandhaḥ*	49
hi	*sambandhaḥ*	17
tadā	*samānakālaḥ*	12
yadā	*samānakālaḥ*	11
tatra	*deśādhikaraṇam*	13
yatra	*samānādhikaraṇaḥ*	5
tathā	*sādṛśyam*	13
tathā	*kriyāviśeṣaṇām*	8
yathā	*sādṛśyam*	13
yathā	*kriyāviśeṣaṇām*	4
yadi	*āvaśyakatā-pariṇāmaḥ*	4
cet	*āvaśyakatā-pariṇāmaḥ*	6
ṭathāpi	*vyabhicāraḥ*	1
yadyapi	*vyabhicāraḥ*	1
anantaram	*anantarakālaḥ*	2
atha	*anantarakālaḥ*	1
atha	*samuccayaḥ*	4
atha	*praśnārthaḥ*	5
ca	*samuccayaḥ*	49
api	*praśnārthaḥ*	1
api	*sambandhaḥ*	52
vā	*anyataraḥ*	6
vā	*praśnārthaḥ*	1
athavā	*anyataraḥ*	2

Table 2: List of discourse relations and frequency occured in Śrimadbhagvadgītā

4. kevala, akelā

The Sanskrit-English Monnier William's dictionary has the following 3 different senses.

1. for, because, on account of

2. just, pray, do

3. indeed, assuredly, surely, of course, certainly

Speijer (1886) (§429) while commenting on it observes "*hi* was at the outset an emphatic, a weak 'indeed', but generally it is a causal particle, at least in prose.". Further in §443 Speijer states "... it has rather a general employment when annexing sentences which contain some motive, reason, cause or even an illustration of that which preceeds."

For the purpose of annotation we do not distinguish between the two usages marking emphasis (sense 2 of Sanskrit-Hindi) and marking exclusiveness (sense 4 of the Sanskrit-Hindi). We treat them under a generic term *sambandaḥ*, but we distinguish these usages from the usage of one marking the cause. The reason for not distinguishing between the emphasis and exclusiveness is that for their disambiguation just a sentence level information is not sufficient. One has to look at the context that may involve extra-linguistic information. There were total 66 shlokas that have '*hi*'. *Śaṅkarācārya* has commented on all the *ślokas* from 10th verse of the second chapter. There were 5 instances of '*hi*' till the 9th verse of the second chapter. Excluding these 5, among the remaining 61, *Śaṅkara* has marked 46 instance of '*hi*' as causal indicator, and 15 fall under the second category.

Both the authors classified '*hi*' in two categories independently without reffering to the *Śaṅkarabhāśya*. The classification is represented in Table 3.

	kārya-kāraṇam	**sambandaḥ**	total
Annotator 1	33	33	66
Annotator 2	49	17	66

Table 3: Inter-annotator agreement

The inter-annotator confusion matrix is shown in the Table 4. The comparison of the annotations of both the authors with that of *Śaṅkara* is shown in the Tables 5 and Table 6.

Thus we notice that, if we consider the *Śaṅkarabhāśya* as the gold data, the performance of the annotators measured against the gold data is not very satisfactory. Annotator 1 could

Annotator1↓	Annotator2 →		
	kārya-kāraṇam	**sambandaḥ**	Total
kārya-kāraṇam	27	6	33
sambandaḥ	22	11	33
Total	49	17	66

Table 4: confusion matrix: Annotator 1 and 2

Annotator1↓	Śaṅkarbhāṣya →		
	kārya-kāraṇam	**sambandaḥ**	Total
kārya-kāraṇam	25	4	29
sambandaḥ	21	11	32
Total	46	15	61

Table 5: confusion matrix : Annotator 1 and Śaṅkarabhāśya

Annotator2↓	Śaṅkarbhāṣya →		
	kārya-kāraṇam	**sambandaḥ**	Total
kārya-kāraṇam	39	6	45
sambandaḥ	7	9	16
Total	46	15	61

Table 6: confusion matrix : Annotator 2 and Śaṅkarabhāśya

mark only 60% of the cases correctly and the annotator 2 marked around 79% of the cases correctly. The disagreement between two annotators is also high, around 42%. If we look at the reason behind these differences we notice that the use of '*hi*' as an emphatic marker or exclusiveness marker sometimes also imply *kārya-kāraṇam*. This is observed in the commentary on '*vyākhyānato viśeṣapratipattiḥ na hi sandehāt alakṣaṇam*'. Here, we notice that almost all commentators before *Nāgeṣa* consider the use of '*hi*' as an emphatic marker, but *Nāgeṣa* categorically calls it *kāraṇa-dyotakaḥ*. The point we would like to drive here is that it is not trivial to disambiguate and many-a-times the whole context, the purpose etc. need to be taken into account.

Based on the data available we came up with some heuristics that uses the information of position of '*hi*' and the presence of pronouns or negative particle before it to classify '*hi*' into two categories. The results of the heuristics are shown in table 7. We note that the machine has provided correct results in 83% of the cases, outperforming both the annotators! It, of course, remains doubtful, if the heuristics developed for SBG will hold good across various genre of texts. The simple heuristic used is: if the word '*hi*' is at the second or the third position in a sentence, then it is marked as a *kāry-kāraṇa-bhāva*, with some special rules when the pronouns and indeclinables occur before the word *hi*. In all other cases it is marked as a *sambandhaḥ*.

Gold Data↓	Machine Results →		
	kārya-kāraṇam	**sambandaḥ**	Total
kārya-kāraṇam	41	5	46
sambandaḥ	5	10	15
Total	46	15	61

Table 7: machine produced results with comparison to gold data

6 Conclusion

The task of identifying analysing and implementing inter-sentential discourse relations with IGT perspective is still at an initial stage. We have identified various inter-sentential relations based on the explicit markers. Our next task is to identify the pair of verbs showing the expectancies such as to ask and to answer, to buy and to sell, etc. Another important task is to develop a module for anaphora resolution. We also noticed that the disambiguation is not an easy task. While some heuristics helped us in disambiguating the word *hi* it is not yet clear how much domain dependenct are these heuristic rules. The most important task ahead is therefore modeling *yogyatā*.

References

Debopam Das and Manfred Stede. 2018. Developing the Bangla RST Discourse Treebank. In *International Conference on Language Resources and Evaluation*.

G V Devasthali. 1959. *Mīmāṃsā: The vākya śāstra of Ancient India*. Booksellers' Publishing Co., Bombay.

M. A. K. Halliday and R. Hasan. 1976. *Cohesion in English*. English Language Series, London.

Dhanurdhar Jha. 2002. *Vākyārtha vivecanam*. Naag Publishers, Jaipur.

Feng Jiang, Sheng Xu, Xiaomin Chu, Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2018. MCDTB: A macro-level Chinese discourse TreeBank. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3493–3504, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

K V R Krishnamacharyulu. 2009. Annotating the Sanskrit texts based on the śābdabodha systems. In *3rd International Sanskrit Computational Symposium*. LNAI Springer Verlag.

Amba Kulkarni and Monali Das. 2012. Discourse analysis of Sanskrit texts. In *Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects*, pages 1–16, Mumbai, India, dec. The COLING 2012 Organizing Committee.

Amba Kulkarni and Sanjiv Panchal. 2019. Co-ordination in Sanskrit. In *Indian Linguistics, 80(1-2)*, pages 59–76.

Amba Kulkarni. 2019. *Sanskrit parsing based on the theories of Śābdabodha*. Indian Institute of Advanced Study, Shimla and D K Publishers (P) Ltd.

Amba Kulkarni. 2021. Sanskrit parsing following Indian theories of verbal cognition. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(2), Apr.

S Sheeja Kumari and Sobha Lalitha Devi. 2016. Annotations of connectives and arguments in malayalam language. volume 25, pages 280–285. 1st Global Colloquium on Recent Advancements and Effectual Researches in Engineering, Science and Technology - RAEREST 2016 on April 22nd & 23rd April 2016.

William C. Mann and Sandra A. Thompson. 1988. *Rhetorical structure theory: Towards a functional theory of text organization.*

Lucie Mladová, Šárka Zikánová, and Eva Hajicová. 2008. From sentence to discourse: Building an annotation scheme for discourse based on prague dependency treebank. In *Proc. of LREC*.

Madhusudan Penna. 2021. *Pūrva Mīmāṃsā śāstra*, volume 2. Booksellers' Publishing Co., Bombay.

Livia Polanyi. 2008. The linguistic structure of discourse. In *The Handbook of Discourse Analysis*, pages 265–281.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2006. *The Penn Discourse TreeBank - Annotation Mannual 1.0*. University of Pennsylvania.

Ravi Teja Rachakonda and Dipti Misra Sharma. 2011. Creating an annotated Tamil corpus as a discourse resource. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 119–123, Portland, Oregon, USA, June. Association for Computational Linguistics.

Kevalanand Saraswati. 1888. *Mīmāṃsā koṣa*, volume 7. Pradnya Pathashala Mandal Granthamala.

P.M. Scharf and H.H. Hock. 2015. *Sanskrit Syntax: Selected Papers Presented at the Seminar on Sanskrit Syntax and Discourse Structures, 13-15 June, 2013, Université Paris Diderot*. Sanskrit Library.

J. S Speijer. 1886. *Sanskrit Syntax*. Leyden : E.J. Brill, University of Cornell.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2019. Framework for question-answering in sanskrit through automated construction of knowledge. In *6th International Sanskrit Computational Linguistics Symposium (ISCLS)*, pages 98–117.

Oza Umangi, Prasad Rashmi, Kolachina Sudheer, Misra Sharma Dipti, and Joshi Aravind. 2009. The Hindi discourse relation bank. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 158–161, Suntec, Singapore, aug. Association for Computational Linguistics.

Bonnie Lynn Webber and Aravind K. Joshi. 1998. Anchoring a Lexicalized Tree-Adjoining Grammar for discourse. In *Discourse Relations and Discourse Markers*.

Florian Wolf and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287.

Deniz Zeyrek, Işin Demirşahin, Ayişiği Sevdik-Çalli, Hale Ögel Balaban, İhsan Yalçinkaya, and Ümit Deniz Turan. 2010. The annotation scheme of the Turkish discourse bank and an evaluation of inconsistent annotations. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 282–289, Uppsala, Sweden, July. Association for Computational Linguistics.

A fast *prakriyā* generator

Arun Prasad
Ambuda
arun@ambuda.org

Abstract

We present `vidyut-prakriya`, a program that generates Sanskrit words along with their Pāṇinian derivations. `vidyut-prakriya` implements more than 2,000 rules of the *Aṣṭādhyāyī* and has strong support for *tiṅanta*s, *kṛdanta*s, *taddhitānta*s, and *subanta*s, with partial support for *samāsa*s and accent. Our program compiles to fast native code and also compiles to WebAssembly for in-browser use. Informal benchmarks indicate that `vidyut-prakriya` is almost three orders of magnitude faster than comparable open-source systems. We end by discussing various applications of a fast *prakriyā* generator and directions for future work.

1 Introduction

Many Sanskrit programs use and rely on a *lemma list* that contains verb roots, nominal stems, and other headwords. For example, an electronic dictionary maps a list of lemmas to a list of definitions. Some programs also rely on a *word list* that contains inflected versions of various lemmas. For example, a more sophisticated electronic dictionary might accept an inflected word then show results for the word's underlying lemma. This distinction between lemma lists and word lists is simple but important. Whereas lemma lists might contain entries like **गम्** and **देव**, word lists might contain entries like **गच्छति**, **जग्मिवांसम्**, **देव्यै**, and **देवानाम्**.

While a word list is useful for dictionaries and other query interfaces, it has other applications as well. For example, programs that analyze Sanskrit sentences have a long history of using word lists internally,[1] which continues into modern approaches like Sandhan et al. (2022). Additionally, popular resources for students of Sanskrit grammar, such as Bodas (2023), display hundreds of thousands of verbal and nominal forms for students, and these forms are regularly consulted by modern-day Sanskrit communities. S. Prasanna (2022) also illustrates the utility of a word list for spellchecking by using both an explicit list of irregular forms and an implicit list that joins base words with a suffix table. We believe that word lists are likewise valuable to any application that cares deeply about correctness.

Given these applications, we believe that an even larger word list might better suit some of these needs, which makes creating such a list a valuable problem to pursue. But since the Sanskrit word list is infinite and grows recursively, we cannot represent it in a straightforward way. So in practice, an infinite word "list" is rather a finite program that can generate words as needed. The challenge, then, is to to create such a program so that it solves needs similar to the ones we describe above.

One promising strategy for creating such a program, as demonstrated by Bharati et al. (2006) and others, is to combine an ad-hoc list of attested forms with some method of abstraction, such as a finite state automaton, a statistical model, or a set of manually implemented rules.

[1] Hellwig (2009) and Goyal and Huet (2016) are particularly notable for their longevity and impact. We also have high regard for the *Saṁsādhanī* toolkit from the University of Hyderabad, but we are less familiar with how it works internally.

This approach works well if the ad-hoc list is sufficiently rich to expose all of the edge cases and subtleties of Sanskrit grammar, but we have found that approaches in this vein, as impressive as they are, are prone to over- and under-generating, which means that they might allow invalid words and reject valid ones. This kind of behavior is not always a problem, and it can even be preferable for some use cases; but, it is less suited for applications that care deeply about correctness.

An alternative strategy is to directly implement the underlying rules that generate these words. This is the approach most famously taken by the *Aṣṭādhyāyī*, which condenses the mechanics of Sanskrit grammar to roughly 4,000 short rules. When combined with secondary texts like the *Dhātupāṭha* and *vārttika*s from the commentarial literature, the *Aṣṭādhyāyī* provides a powerful system for generating an infinite number of Sanskrit words.

Accordingly, there are several systems that have implemented parts of the *Aṣṭādhyāyī*, each with different philosophies and goals. Mishra (2009), for example, proposes a formal structure that is highly Pāṇinian in spirit, with each sound in a word having a specific formal representation. Scharf (2015) likewise creates a meticulous formal representation of the *Aṣṭādhyāyī*'s rules in XML that can be converted to executable code. Goyal et al. (2009) use a simpler internal representation but also describe a rich model for conflict resolution between rules. Patel and Katuri (2015), meanwhile, avoid simulating conflict resolution but gain substantial performance benefits in return.

A program that implements the *Aṣṭādhyāyī*'s rules can usually also create a *prakriyā*, a step-by-step derivation that shows which rules of grammar act to create a specific inflected word. For students, a *prakriyā* explains and elucidates the principles of the grammar; for engineers, a *prakriyā* reveals the grammar's operations in case it needs to be debugged; and for downstream applications, a *prakriyā* provides detailed grammatical information about a given word. Thus a *prakriyā* is highly useful in multiple settings.

Rule	Result
3.2.123	भू + लँट्
3.4.78	भू + तिप्
3.1.68	भू + शप् + ति
7.3.84	भो + अ + ति
6.1.78	भव् + अ + ति
(final)	भवति

Table 1: An abbreviated *prakriyā* for the word भवति. Items on the left are references to specific rules in the *Aṣṭādhyāyī*. We have elided various minor rules.

This paper presents our work toward building a comprehensive *prakriyā* generator based on the *Aṣṭādhyāyī*. We have implemented just over 2,000 rules from the *Aṣṭādhyāyī*, and these rules span all major sections of the text. In addition, we have implemented around 500 rules from the *Uṇādipāṭha* and around 50 rules from the *Pāṇinīyaliṅgānuśāsanam*. Our main contributions are the generator itself and our specific engineering decisions, which we believe make a complete implementation of the *Aṣṭādhyāyī* more feasible and useful.

Section 2 describes our approach and the engineering and grammatical principles we follow in this work. Section 3 describes our program's high-level architecture, including which texts we use as the basis of our work. Section 4 describes our implementation, including our data model, pieces of our API, a few example rules, and a description of how we handle optional derivations. Section 5 describes our testing methodology, which draws primarily from the traditional grammatical literature. Section 6 describes the current state of our work and its limitations. The remaining sections describe applications of this work, directions for future work, and our overall conclusions.

2 Approach

We have viewed our program first as an engineering problem and second as a grammatical one. That is, we have prioritized those principles that allow us to build a safe, fast, and consistent program that produces accurate results. Since we are non-grammarians, our approach is to some extent *anti*-theoretical, meaning that we have used traditional grammatical literature primarily as a source of examples but have not deliberately followed a specific system of interpretation or rule ordering.

2.1 Engineering strategy

Broadly, we follow a variety of engineering principles that we have found to lead to maintainable and high-quality software. While these principles are not absolutes, they are generally true in our system:

- *Don't reinvent the wheel.* We acknowledge that *prakriyā* generators are well-trod ground. That said, we have aimed to create a generator that improves on the state well enough to become a community standard. In service of this goal, we have implemented our generator in Rust, a relatively new systems language that combines low-level control with high-level ergonomics. One happy consequence of this decision is that we can easily bind our program to other languages. In other words, we can readily provide the same core implementation in multiple environments and languages. As proofs of concept, we have created Python bindings[2] through Rust's `PyO3` library and JavaScript bindings through Rust's `wasm-pack` ecosystem.[3]

- *Don't repeat yourself (DRY).* We generally implement a rule in exactly one place in our code, which means that each rule we use has exactly one formal specification. Where necessary, we reuse rules sparingly through function calls. Likewise, we maintain a simple, linear, and predictable control flow as opposed to (for example) dynamically selecting and ranking rules within an event loop.

- *If it isn't tested, it's broken.* The *Aṣṭādhyāyī* contains thousands of interconnected rules, and an innocuous change to one rule can easily break dozens of others. Therefore, we generally test each rule with representative examples and counterexamples from the *Kāśikāvṛtti*. We have also an extensive set of additional tests from the *Siddhāntakaumudī*. We explain more about our testing procedure in section 5.

- *Speed is a feature.* A comprehensive test suite is useful only if it can run in a reasonable amount of time. While we avoid over-optimizing (*Premature optimization is the root of all evil*), we have done enough that our full test suite, which creates roughly 1.7 million words, runs in under a minute on our 2019 MacBook Pro. In particular, Rust has continued to pay dividends not only through its native speed but also due to its rich ecosystem of high-performance libraries.

- *Given enough eyeballs, all bugs are shallow.* All of our code is open-source and available online.[4] So far, five people have reported bugs and five have submitted code to our program. As more people use our program and its results, we increase the likelihood of finding mistakes and errors, which we can then correct and add to our test suite.

2.2 Grammatical strategy

As non-grammarians without much expertise in traditional grammar, we have laid aside issues of theory and set ourselves a much humbler task: to generate a list of valid Sanskrit words while adhering to Pāṇinian rules as closely as we can. Specifically:

[2] Available at `https://pypi.org/project/vidyut/`

[3] Demonstrated at `https://ambuda-org.github.io/vidyullekha`.

[4] You may find our code at `https://github.com/ambuda-org/vidyut` under the `vidyut_prakriya` directory.

- In our schema, a word is *valid* if and only if it is attested by a grammatical authority. So far, we have focused most closely on the *Kāśikāvṛtti* and the *Siddhāntakaumudī* (SK) as published at Bodas (2023). We hope to expand our set of examples as time allows.

- We follow Patel and Katuri (2015) by manually ordering rules and avoiding any explicit model of conflict resolution. This policy greatly increases the program's efficiency because the program can run code in a simpler and more predictable way, which helps both the compiler and the CPU process the program more efficiently.

- We interpret a rule in whatever way will let us generate valid words and avoid invalid ones. Generally, we have followed the interpretations of the *Kāśikāvṛtti*, but there is no broader principle we apply when interpreting rules. We do not model *anuvṛtti*.

- For ease of reference, we prefer to group rules by their ordering in the text. For example, our implementation of 7.2.61 is immediately next to our implementation of rule 7.2.62, and likewise these two rules appear in a similar place in our overall control flow. It is often not possible to follow this condition, but we do so where we can.

3 Architecture

This section explains the high-level architecture of our program, with a low-level view of implementation details in section 4.

3.1 Texts used

We focus on the *Aṣṭādhyāyī* and also include *vārttika*s from the *Kāśikāvṛtti* and the *Siddhāntakaumudī*. Our approach to *vārttika*s has been to first find examples from the grammatical literature that the program does not support and then implement the *vārttika*s necessary to support those examples. We have taken this approach because we think that doing so helps us better shape the overall design of the program.

We have consulted and used specific *paribhāṣā*s from the *Paribhāṣenduśekhara* only if doing so was necessary to resolve an explicit error in our program. For example, we have made use of **श्तिपा शपानुबन्धेन** ... for our *yaṅ-luk* derivations.

Our *Dhātupāṭha*, which comes from Bodas (2023), is a superset of *Dhātupāṭha*s from various sources, including the *Siddhāntakaumudī*, the *Mādhavīyadhātuvṛtti*, and the *Bṛhaddhātukusumākara*. The *dhātu*s in this list include *anubandha*s, and we have also modified the list to explicitly include accent. Some example entries in SLP1 transliteration include `RI\Y`, `qukf\Y`, and `qupa\ca~^z`. Our program is not coupled to any specific *Dhātupāṭha* and will accept any *dhātu* as long as the user defines its *gaṇa* and *antargaṇa* and correctly specifies its *anubandha*s and accent.

The other texts we use, including the *Uṇādipāṭha* and the *Gaṇapāṭha*, are as specified in the *Siddhāntakaumudī*.

3.2 Data model

Traditional grammar describes the items in a derivation with various labels, such as *pratyaya*, *āgama*, *dhātu*, *prātipadika*, *abhyāsa*, and so on. In our data model, all of these concepts are aspects of a more general notion that we call a *term*.[5] In our program, we explicitly model and store all of the following for a given term:

- The term's "visible" or surface representation, which will change according to the rules of the grammar.

- The term's instructional (**औपदेशिक**) form, which is how the term is first described in the grammar. This form includes accents and *anubandha*s where applicable.

[5] We do not know if this concept has a traditional name.

- All designations (संज्ञा) that are added by the rules of the grammar, as well as other properties that we describe in 4.1.
- All prior instructional forms, if full substitution applies by 1.1.55 (अनेकाल्शित्सर्वस्य).

For example, our program will represent the verb root डुकृञ् as having the instructional form `qukf\Y` (in SLP1 transliteration), the visible form `kf` (which might change to `kar`, `kAr`, etc. during the derivation), and the designations of *dhātu*, *aṅga*, *ḍvit*, and *ñit*. We also store that the root vowel is *anudātta* since this information is necessary to trigger certain rules.

All other details of our data model are less important, and we refer the reader to section 4.1 for details.

3.3 Argument model

Since the *Aṣṭādhyāyī* is a precise system, we require the user to specify their input conditions in precise detail. For example, suppose that we wish to derive the word `kArayan`. To do so, we must tell the program that we wish to derive a masculine nominative singular *subanta* from the *kṛdanta* formed by adding *śatṛ* to the *sanādi-dhātu* formed by adding the suffix *ṇic* to the *mūla-dhātu ḍukṛñ* that is listed in *tanādigaṇa*. This complex idea becomes clearer when represented as an s-expression:

```
(subanta
    (kṛdanta
        (dhatu
            sanadi
            (dhatu mula qukf\Y tanādi)
            Ric)
        Satf)
    masc nom sg)
```

We have tried to strike a balance between precision and pedantry, and we use Rust's rich type system to guide the user toward a correct request. For example, a Rust program that tries to add a *kṛt* suffix to a *prātipadika* will fail at compile time because a *kṛt* suffix can be added only after a *dhātu*. Section 4.4 contains details on our specific representation in code.

3.4 Rule model

We model a rule as having two parts: a *filter* that matches certain conditions and an *operator* that applies some change to the grammar. In code, this model naturally maps to a simple `if` statement:

```
if filter(prakriya) {
  operator(prakriya);
}
```

The most important aspect of this model is that it does not model *anuvṛtti*: for each rule, the program must explicitly specify all of the conditions necessary for the rule to apply. By avoiding this critical part of the grammar, our program becomes simpler and faster.

That said, an analogue of *anuvṛtti* exists as follows. If multiple rules can apply only when some condition `x` is true, we have found it convenient to write these rules like so:

```
if x {
    if filter_1(prakriya) {
      operator_1(prakriya);
    } else if filter_2(prakriya) {
      operator_2(prakriya);
    }
}
```

3.5 Optional rules

Rules can also apply optionally under various conditions. Our program supports two kinds of option and models them in different ways.

First, we support rules qualified by words indicating an option (*vā*, *vibhāṣā*, *anyatarasyām*, ...), rules qualified by the name of some grammarian, and most rules defined only in a specific semantic sense as follows:

1. Suppose that the program sees optional rules A, B, and C during the derivation. By default, the program accepts each rule. In addition, it also records that rules A, B, and C apply optionally.

2. Once the derivation is complete, the program inspects the output of step (1) and notices that rules A, B, and C are optional. it then creates three new combinations from these rules, namely $(A, B, \neg C)$, $(A, \neg B)$, and $(\neg A)$. Here $\neg A$ means a rejection of A. These three "rule paths" are all added to a queue of unexplored combinations.

3. As long as the queue is non-empty, we pop a path from the queue and run the derivation again using the rule decisions that the path describes. If this process encounters new optional rules, we likewise create new rule paths for them. For example, if we find that we can follow the rule path $(A, \neg B, D, E)$, then we add $(A, \neg B, \neg D)$ and $(A, \neg B, D, \neg E)$ to the path.

To model a choice between three or more options, we split the rule into two binary options. For example, consider rule 3.1.40 (**कृञ्चानुप्रयुज्यते लिटि**), which provides a choice of three different verb roots (**कृ**, **भू**, and **अस्**) when deriving the periphrastic perfect tense. In this scenario, our program first chooses whether or not to use **भू** then chooses whether or not to use **अस्**.

This basic model has been workable for most rules but is too crude for *taddhita* rules, where a specific suffix is available under different rules in different meaning conditions. We instead model such rules as follows:

1. When using the program, the user can optionally request that a *taddhita* should be added only with a specific meaning.

2. When checking if a *taddhita* rule can apply, the program first checks if the user requested a specific meaning that is compatible with the rule. If so, and if the rule requires a different meaning condition, the program skips the rule. If the user did not request a specific meaning, the program simply adds the *taddhita* as long as at least one rule can provide it for the user's input conditions.

3.6 Conflict resolution

We provide no explicit model of conflict resolution. That is, if rule 2 can block rule 1, our program does not inspect the properties of these two rules and has no explicit logic to rank or decide which should apply. Instead, we take the approach similar to Patel and Katuri (2015) and implement these rules as follows:

```
if filter_2(prakriya) {
  operator_2(prakriya);
} else if filter_1(prakriya) {
  operator_1(prakriya);
}
```

We stress that this approach neglects a core issue in interpreting and modeling the *Aṣṭādhyāyī*. It provides no model of a rule's properties, no explicit encoding of common *paribhāṣā*s like **पूर्वपरनित्यान्तरङ्गापवादानामुत्तरोत्तरं बलीयः**, no convenient way to reorder or rerank rules (other than editing code), and no precise way of informing the user that one rule has blocked another.

As compensation, however, this approach provides code that is much simpler, much faster, and much easier to change and understand, and we believe that these advantages are highly compelling.

3.7 Rule ordering

Given the rule model we describe above, a natural question is how we decide whether one rule should apply before another. Simply, our ordering is ad-hoc and chosen in whatever way will correctly generate the test cases we describe in section 5 while adhering to the engineering principles we described in 2.1. While this approach is crude, our use of the "DRY" principle of non-repetition in code means that our program generally commits to a single ordering that applies consistently across *all* examples in our test suite.

We say that this principle applies *generally* because we have occasionally had to violate it by duplicating a rule. For a trivial example, we apply the *it-saṃjñā* rules and most *saṃjñā* rules whenever a term is added to the derivation. For a more interesting example, our system applies rule 6.1.66 (**लोपो व्योर्वलि**) twice: once before *guṇa* so that we can correctly derive **क्रोपयति** (from **क्रूय्** + **पुँक्** + **णिच्**) and once after *guṇa* so that we can correctly derive **अभूत्** (from **भू**).

3.8 Rule organization

Conceptually, we group our rules into two categories: *preparing* rules and *finalizing* rules. *Preparing* rules introduce terms to the derivation but apply few changes, and they include most rules in *adhyāya*s 1-5. *Finalizing* rules apply various phonetic changes to terms in the derivation, and they include most rules in *adhyāya*s 6-8. We have separated rules in this way so that we can support nested derivations, which we have tested in basic cases but not yet in recursive ones.

For example, let us continue with the example of `kArayan` that we mentioned in section 3.3. Given our specification, the program prepares each item from the innermost out then finalizes the derivation to create the requested output:

1. *Prepare* `qukf\Y` by adding it to the derivation and applying *it-saṃjñā-lopa* and other *saṃjñā* rules.

2. *Prepare* `Ric` by adding it to the derivation and applying the same rules as in (1).

3. *Finalize* to create the root `kAri`. Our current implementation always runs this phase after adding *sanādi* suffixes because doing otherwise causes the program to fail on some of our test examples.

4. *Prepare* `Satf~` by first adding `la~w` to the derivation then replacing it with `Satf~`, as specified by 3.1.124 (**लटः शतृशानचावप्रथमासमानाधिकरणे**). We then apply *it-saṃjñā-lopa* and other *saṃjñā* rules. `Satf~` also conditions the addition of `Sap`, so we add it then apply the same rules as in (1).

5. *Prepare* `su~` by adding it to the derivation and applying the same rules as in (1). The derivation at this stage is `kAri + a + at + s`.

6. *Finalize* to apply substitutions and sound change rules. The final word is `kArayan`.

4 Implementation

Here we explain some implementation details from our system, which amounts to around 25,000 source lines of code excluding tests. All code in this section is adapted directly from our implementation, with light edits for clarity and readability.

We store text data internally with SLP1, which simplifies our underlying logic and reduces the computational overhead for core operations. For details on SLP1, see Scharf and Hyman (2012).

4.1 Core data types

We start by presenting the core data types in our program. Almost all of our program's rules involve testing and transforming the data types below.

Our most basic data type is a *Term*, which is a string annotated with additional metadata:

```
struct Term {
    upadesha: Option<String>,
    text: String,
    sthanivat: String,
    tags: EnumSet<Tag>,
    gana: Option<Gana>,
    antargana: Option<Antargana>,
    lakshanas: Vec<String>,
    svara: Option<Svara>
}

enum Tag { .. }
enum Gana { .. }
enum Antargana { .. }
enum Svara { .. }
```

Here are some of the key terms in the example above:

- `struct` indicates a heterogeneous collection of fields.
- `enum` indicates an *enumeration*, i.e. a choice of one item among the members in a list.
- `Vec` indicates a list of elements.
- `Option` is how Rust models a field that can be either present or absent.
- `EnumSet` is a third-party library that lets us model a set of `enum` values.

The most important fields on `Term` are `text`, which is the surface representation of this term; `upadesha`, which is the instructional (**औपदेशिक**) representation as enunciated in works like the *Dhātupāṭha*; and `lakshanas`, which contains a history of substitutions for this term, as per rule 1.1.62 (**प्रत्ययलोपे प्रत्ययलक्षणम्**). In particular, `upadesha` lets our code process a `Term` in predictable ways despite any phonetic changes to the surface form in `text`. `sthanivat`, which we named by reference to rule 1.1.56 (**स्थानिवदादेशोऽनल्विधौ**), is necessary for certain rules in our implementation, particularly those that deal with *dvitva* on a causative root.[6] `svara` specifies the accent type and (if applicable) which vowel in the term the accent should apply to.[7]

The `Tag` enum generalizes the *saṃjñā* concept from traditional grammar. In addition to including traditional *saṃjñā*s, `Tag` also contains various flags that are useful for the derivation. For example, we indicate that a term's last *a* vowel has been deleted through the tag `FlagAtLopa`.

We model a derivation as a list of `Term` structs along with additional metadata:

```
struct Prakriya {
    terms: Vec<Term>,
    tags: EnumSet<Tag>,
    history: Vec<Step>,
    config: Config,
    rule_decisions: Vec<RuleChoice>,
```

[6] We chose the name `sthanivat` for this concept for lack of a better name. We wish to assure the reader that we are aware of how important rule 1.1.56 and have modeled it appropriately throughout the program.

[7] Rust `enum`s are more powerful than simple enumerations and can also contain extra data. This type of model is more properly known as a *sum type*.

```
    lakara: Option<Lakara>,
}

enum RuleChoice {
    Accept(Rule),
    Decline(Rule),
}

enum Rule {
    Ashtadhyayi(String),
    Kashika(String),
    Dhatupatha(String),
    Unadi(String),
    Linganushasana(String),
    Kaumudi(String),
}
```

The syntax here is broadly similar to the `Term` illustration above. For brevity, we have elided the `Config` and `Step` structs, which are straightforward. In the definitions of `RuleChoice` and `Rule`, note that Rust's `enum` type can also associate data with each enum variant.

After `terms`, the most noteworthy fields here are `history` and `rule_decisions`. `history` stores each rule applied in the derivation along with its result. `rule_decisions` stores specific decisions on optional rules that were encountered during the derivation, and we refer the reader to section 3.4 for details.

The strength of this data model is that we can split a word's derivation into separate strings that each have their own metadata, and we have found such a data model to be highly convenient for most rules. That said, its main limitation is that it is not a hierarchical representation, meaning that we cannot easily take either a broader or narrower view:

- *The broader view:* Our representation is most convenient when one term represents one core notion, such as a pratyaya. But during a derivation, we might introduce various *āgama*s that come between a *pratyaya* and the base it follows. In this situation, we most frequently resort to a `TermView`, a new data structure that abstracts over multiple `Term`s. While `TermView` is useful and has reasonable ergonomics, it ultimately highlights a weakness of the underlying data model.

- *The narrower view:* Our representation assumes that each sound in a derivation belongs to exactly one term. However, this assumption does not always hold. For example, certain derivations apply rules that fall in the jurisdiction of rule 6.1.84 (एकः पूर्वपरयोः), which states that a single substitute should be treated as part of both the previous and the following items. Our data schema above cannot model this, and we have worked around it with a small hack: we annotate the first term in the sandhi combination with the `FlagAntyaAcSandhi` tag and block rules like 8.2.39 (झलां जशोऽन्ते) from applying between two terms if the first term has this tag. In the future, we might implement an explicit model of rule 6.1.84 that our API can check.

A more Pāṇinian data structure might be a list of string spans that each have their own metadata. But in our view, it is not obvious how to create a clean and efficient API for working with such spans. Despite its limitations, our list of `Term`s has been successful so far.

4.2 API

On top of these data types, we have created a rich API that lets us easily inspect, test, and transform our derivation state. Below is a representative example of our `Term` API:

```
    impl Term {
        fn antya(&self) -> Option<char> {
            self.text.chars().rev().next()
        }

        fn has_antya(&self, char: c) -> bool {
            self.antya() == Some(c)
        }

        fn set_antya(&mut self, s: &str) {
            let n = self.text.len();
            if n >= 1 {
                self.text.replace_range(n - 1..n, s);
            }
        }
    }
```

Here, `antya` returns the `Term`'s final character, or `None` if the term is empty. Likewise, `has_antya` tests whether the `Term` has a given final sound, and `set_antya` modifies the last sound of the `Term` in-place. (`set_antya` accepts multiple characters to better support certain kinds of substitutions.)

In our production code, `has_antya` uses Rust's support for generic arguments to also accept a *set* of sounds, which lets us test (for example) whether a given `Term` ends in a specific *pratyāhāra*.

`Prakriya` likewise exposes a high-level API for changing internal state. These functions primarily accept *closures*, which are akin to inline functions. We use closures in part for readability and in part because we found it easier to do so while complying with Rust's semantics. Below is a representative example of our `Prakriya` API:

```
impl Prakriya {
    fn run(&mut self, rule: Rule, operator: impl Fn(&mut Prakriya)) -> bool {
        operator(self);
        self.step(code);
        true
    }

    fn run_optional(&mut self, rule: Rule, operator: impl Fn(&mut Prakriya)) -> bool {
        if self.is_allowed(rule) {
            operator(self);
            self.step(rule);
            true
        } else {
            self.decline(rule);
            false
        }
    }
}
```

4.3 Implementing rules

Together, this combination lets us tersely express rules in a human-readable way without sacrificing performance:

```
    let yi_kniti = next.has_adi('y') && next.is_knit();
    if anga.has_upadesha("SIN") && next.is_sarvadhatuka() {
        prakriya.run_at("7.4.21", anga_index, |term| term.set_text("Se"));
```

```
} else if anga.has_upadesha("SIN") && yi_kniti {
    prakriya.run_at("7.4.22", anga_index, |term| term.set_text("Say"));
}
```

Note the explicit ordering of rules, the relatively terse code, and the use of `if-else` chains to implement rule blocking. Our production code is largely the same, but it heavily abbreviates common terms like `prakriya` and `term` and contains some method names that are legacies of earlier stages of development.

4.4 Argument types

Our public API requires users to define their input conditions with precise types. To continue the example from section 3.3, our program models the input conditions for `kArayan` as follows, with some minor syntax elided for clarity:

```
let kr = Dhatu::mula("qukf\\Y", Tanadi).with_sanadi(&[Sanadi::Ric]);
let karayat = Krdanta::new(kr, Krt::Satf);
let karayan = Subanta::new(karayat, Pum, Prathama, Eka);
```

4.5 Performance

Since our full test suite has more than a million examples, we wish to ensure that our program completes in a reasonable amount of time. To that end, we have taken reasonable steps to ensure that the developer experience does not suffer.

Perhaps the most significant decision here is our choice of the Rust programming language. Rust's combination of speed and memory safety makes it an attractive choice for high-performance programming projects, as noted in work like Bugden and Alahmar (2022). In addition to Rust's native capabilities, we wish to highlight three other features that have benefited our program:

- *Strong tooling.* The default Rust installation includes a tool called `cargo`, which manages dependencies, builds code for different environments, runs tests, lints and formats code, and catches common style problems. `cargo` has allowed us to set aside ancillary concerns and focus on implementing rules, and it has also made it easier for us to onboard new contributors to our work.

- *Useful libraries.* Rust maintains a centralized package repository where library versions are guaranteed to be stable and available. For example, the `compact_str` library gives us access to memory-efficient string types that can be stack-allocated if they are sufficiently short. As another example, the `rayon` library provides a lightweight library for parallel execution on iterators, which allows us to more quickly generate a very large word list.

- *Easy reuse.* Rust is easy to bind to other languages and environments, which removes the toil of porting an implementation to another setup. As a proof of concept, we have created the `vidyut` Python library, which is available on Python's standard package index. We have also created a WebAssembly build that can run on a user's device without an internet connection.

5 Testing

The *Aṣṭādhyāyī* generates an infinite number of words, and it is impossible to test them all. Therefore, any implementation of the *Aṣṭādhyāyī* must have a robust testing strategy to justify some level of correctness.

We have tested our program with three kinds of tests: *unit tests* from the *Kāśikāvṛtti*, *regression tests* from the *Siddhāntakaumudī*, and *snapshot tests* that monitor changes over time. Our unit tests and regression tests combine to just under 32,000 lines of source code, and our snapshot test suite contains just under 1.7 million examples.

5.1 Unit tests

Generally, the ancient grammatical literature illustrates the function of a rule with various examples and counterexamples, which establish and limit the rule's scope respectively. We have especially leaned on the *Kāśikāvṛtti* for these examples, since it works through the *Aṣṭādhyāyī* rule by rule and generally limits its commentary to one rule at a time.

A typical unit test appears as follows:

```
#[test]
fn sutra_3_1_68() {
    assert_has_lat(&[], &dhatu("BU", Bhvadi), &["Bavati"]);
    assert_has_tip(&[], &dhatu("qupa\\ca~^z", Bhvadi), &["pacati"]);
}
```

We have created a variety of test functions like `assert_has_lat`, `assert_has_tip`, etc. to verify certain forms. In the example above, the first argument is for *upasarga*s, and `&[]` indicates that we wish to derive the given form without *upasarga*s. Note that `"qupa\\ca~^z"` contains both *anubandha*s and accent marks, both of which are necessary for the program to run correctly. Note also that we use `assert_has_tip` to restrict the output for **पच्** to use only `parasmaipada` endings, if they are available for the root. If we used `assert_has_lat` instead, then the program would produce both **पचति** and **पचते** and the test would fail.

We implement a rule's tests by including all examples mentioned in the *Kāśikāvṛtti*'s commentary on that rule, to the extent that we are able to. We fail to do this either if we are unclear on the intended result or if bugs or other technical limitations prevent us from implementing a given example. For now, we mark these challenging examples with `TODO`s, or in more significant cases, we disable the test entirely. For example, we have disabled the test for rule 3.2.12 (**स्तम्बकर्णयो रमिजपोः**) because we have not implemented the rule that allows non-deletion of the case suffix, which means we cannot produce the expected results **स्तम्बेरम** and **कर्णेजप**.

5.2 Regression tests

Although the *Kāśikāvṛtti* is thorough, its examples are sometimes insufficient to verify that the program is working correctly. In these cases, we have drawn examples from the *Siddhāntakaumudī* (SK), which tends to focus more on the overall *prakriyā* rather than on specific rules. For example, we once noticed a bug in our program where we failed to produce both **ऊर्णुनविथ** and **ऊर्णुनुविथ** (**ऊर्णुञ्** + **सिप्**, **लिँट्**). We found an illustration of this case in Kaumudī 2447 and accordingly implemented it as a test:[8]

```
#[test]
fn sk_2447() {
    let urnu = d("UrRuY", Adadi);
    assert_has_sip(&[], &urnu, Lit, &["UrRunuviTa", "UrRunaviTa"]);
    assert_has_tip(&[], &urnu, Lut, &["UrRuvitA", "UrRavitA"]);
    assert_has_tip(&[], &urnu, Lot, &["UrROtu", "UrRotu", "UrRutAt"]);
    assert_has_mip(&[], &urnu, Lot, &["UrRavAni"]);
    assert_has_iw(&[], &urnu, Lot, &["UrRavE"]);
}
```

Our test suite currently includes almost all examples from *prakaraṇa*s 8-13, which deal with *subanta*s, and 43-58, which deal with *tiṅanta*s. As our program stabilizes, we expect to add more and more test cases from the *Siddhāntakaumudī*.

[8] Some readers might ask how this test passes given that words like `UrRutAd` are also valid forms. Briefly, we have configured our test functions so that they avoid generating nosily duplicative forms. Otherwise, our test logic would be both more tedious to write and more cumbersome to read.

5.3 Snapshot tests

Our largest test suite is a *snapshot test* that deterministically generates around 1.7 million words, including all basic verbs in *kartari-prayoga* and *karmaṇi-prayoga*, all *kartari* and *karmaṇi* forms of *sannanta*, *ṇijanta*, *yaṅanta*, and *yaṅluganta* verbs, and a variety of *kṛdanta*s as well. We store the hashes of these files as part of our test suite, which means that if any result in any file changes, our test suite will raise an error that a human being must manually review.

Snapshot testing demonstrates stability but does not prove correctness. Even so, it is a useful tool for verifying that a rule change does not have unintended consequences.

6 Results

6.1 Implemented rules

	Pada 1	**Pada 2**	**Pada 3**	**Pada 4**
Adhyaya 1	27 / 75	31 / 73	72 / 93	36 / 110
Adhyaya 2	41 / 72	9 / 38	2 / 73	35 / 85
Adhyaya 3	117 / 150	139 / 188	79 / 176	38 / 117
Adhyaya 4	86 / 178	73 / 145	87 / 168	111 / 144
Adhyaya 5	64 / 136	64 / 140	73 / 119	62 / 160
Adhyaya 6	117 / 223	8 / 199	38 / 139	143 / 175
Adhyaya 7	80 / 103	112 / 118	84 / 120	86 / 97
Adhyaya 8	0 / 74	65 / 108	59 / 119	33 / 68

Table 2: Rule implementation of 2071 total rules by *adhyāya* and *pāda*. This is an undercount that includes only those rules that might show up in a *prakriyā*. *Paribhāṣā*s and simpler *saṃjñā* rules like **ह्रस्वं लघु** are not counted here.

Table 2 shows our total count of implemented rules by *adhyāya* and *pāda*. Rules are drawn broadly from all sections of the *Aṣṭādhyāyī*, with the notable exception of *pāda* 8.1.

- *Tiṅantas* have been our primary focus. Our system includes support for all *lakāra*s and *prayoga*s and implements almost all of the *pada* rules in section 1.3, including rules that depend on a specific *upasarga*. Our system also supports *sannanta*, *ṇijanta*, *yaṅanta* and *yaṅluganta* roots, with experimental support for denominative (**नामधातु**) roots. Example words that our program produces include **गच्छति**, **जिगमिषति**, **सञ्चस्कार**, **सिस्रावयिषति/सुस्रावयिष्यति**, **अर्दिधिषति**, and **पापचिषते**.

- *Kṛdantas* have been a secondary focus, and our system here has broad coverage for a variety of common suffixes, including **घञ्**, **ल्युट्**, **ण्वुल्**, **शतृ**, **क्त्वा**, **क्त**, **कसुँ**, and the like. We have also implemented support for around 500 rules from the *Uṇādipāṭha*. Example stems include **गत**, **अभ्यान्त/अभ्यमित**, and **जग्मिवस्/जगन्वस्**.

- *Taddhitāntas* likewise have broad coverage, and they explicitly model a variety of fine-grained meaning conditions with a Rust `enum`. We have met with some challenges here when implementing rules that match against a semantic class, such as "part of the body" as used in 4.3.55 (**शरीरावयवाच्च**). For our coverage here, see *adhyāya*s 4 and 5 in table 2.

- *Subantas* have strong support, but we have not run a formal evaluation against systems like Patel and Katuri (2015). As a rough measure of quality: excluding 8 sutras, we support all examples in *prakaraṇa*s 8-13 of the *Siddhāntakaumudī*.

- *Samāsa*s have basic support. We have implemented many of the rules from *pāda*s 2.1, 2.2, and 6.3 and support a basic model of *upapada-samāsa*s so that we can implement the various

kṛt-pratyaya rules in *pāda* 3.2. Otherwise, initial attempts to model this section have not felt satisfying since so many rules depend on specific semantic conditions and are relatively less mechanical than other sections of the text. One option here is to formally model these semantics in Rust then expose the Rust model through an API.

- *Vedic rules* have been a recent focus, and we currently support around ten of them.
- *Accent rules* have partial support. Currently, our accent model is limited to simple patterns like लित्, चित्, पित्, रित्, and the like. For details on our data model for accent, see 4.1.

6.2 Testing

Test coverage broadly follows the pattern of table 2, but we have disabled around 7 percent of sutra tests. This 7 percent figure is misleadingly high for two reasons. First, we have disabled tests where the system generates results that we don't know how to reconcile with the commentaries due to our own lack of grammatical knowledge. For example, the commentary on rule 7.2.58 (गमेरिट् परस्मैपदेषु) proposes the form सङ्गंसीष्ट as a counter example, but our system also generates सङ्गसीष्ट. We believe that both are correct but have not yet spent the time to vet this form. Second, we have ignored a test for a rule when any one of its examples is generated erroneously, no matter how rare that example might be.

The main comparison we wish to present here is with `SanskritVerb` (Patel (2023)), an extension of the work in Patel and Katuri (2015) with support for *kartari-tiṅanta*s. `SanskritVerb` is a standard implementation in the open-source community and is used in popular projects like Bodas (2023).

We performed an exhaustive comparison against all basic *kartari-tiṅanta*s generated by `SanskritVerb`, which total to around 200,000 forms. After fixing bugs in our setup, `vidyut-prakriya` features various small gains over `SanskritVerb`. A sample:

- Support for optional *karmaṇi-luṅ* forms (अप्यायि, अतायि, अदीपि).
- Support for optional *tanādi-luṅ* forms (अतत, अतथाः).
- Support for periphrastic perfect derivations with अस् and भू (चोरयामास, चोरयाम्बभूव).
- Support for optional forms like स्तभ्नोति, स्तुभ्नोति, and so on.
- Broad support for *ubhayapada* derivations in *curādi-gaṇa*.
- Stronger support for short vowel lengths in certain *curādi-gaṇa* roots, commonly known as *mittva*.
- Support for extra *ātmanepada* forms for various roots, such as स्था per rule 1.2.23.

Most differences are of this kind. They are small, incremental improvements based on supporting or tweaking an existing rule. One notable gain is that we support rule 3.1.31 (आयादय आर्धधातुके वा), which allows a wide variety of optional forms for various roots listed in rules 3.1.28 to 3.1.30. We can support rule 3.1.31 only because our program has strong support for optional derivations.

As time allows, we hope to also run a comparison with the Saṁsādhanī system from the University of Hyderabad.

6.3 Performance

Performance comparisons are an inexact art. This is especially so when comparing two very different systems. With that caveat, we describe an inexact performance comparison between `vidyut-prakriya` and `SanskritVerb`. Specifically, we measure how long it takes each program to generate all basic *kartari-tiṅanta*s for a verb root.

Setup	Time per root (ms)
`SanskritVerb`	1400
`vidyut-prakriya` (re-compile after code change)	7.0
`vidyut-prakriya` (re-compile without code change)	3.5
`vidyut-prakriya` (no re-compile)	3.3
`vidyut-prakriya` (no re-compile, no *prakriyā*s)	2.4

Table 3: Comparison of different setups when generating all basic *kartari-tiṅanta*s for a verb root. All setups were run on the same machine and timed with the `time` command. Results show the mean over a representative sample of roots (100 for `SanskritVerb` and 2200 for `vidyut-prakriya`). Here, "no *prakriyā*s" means that *prakriyā* logging is disabled.

All tests were run on the same machine and timed with Unix's `time` command, and these tests compare both systems as they existed around October 2023. We have not run an updated evaluation due to time constraints, but we believe that this comparison is still informative because the code paths being tested have not substantially changed since then.

Table 3 presents the results of this comparison under different scenarios. If using `vidyut-prakriya` as a release build with no extra compilation, we see that `vidyut-prakriya` is more than 400 times faster than `SanskritVerb`. If we disable *prakriyā* logging for `vidyut-prakriya`, we find that it is almost 600 times faster.

The most obvious explanation for this difference is that `vidyut-prakriya` is written in Rust while `SanskritVerb` is written in PHP, which is generally a much slower language than Rust. In addition, `SanskritVerb` tends to implement rules with regular expressions, which means that both checking whether a rule can apply and applying a rule to the derivation is relatively costly. In comparison, `vidyut-prakriya` tends to perform direct comparisons on stack-allocated strings, which is a much faster procedure.

PHP and other interpreted languages certainly have their own advantages. For example, testing a PHP code change on a single example is much faster than doing the same in Rust, which requires a compile step of several seconds. But if we wish to generate a *large* number of words, then a compiled project like `vidyut-prakriya` is much more effective. This is especially true for the millions of words we generate currently, but the same principle holds even for a few thousand.

6.4 Bugs and errors

`vidyut-prakriya` has a variety of small errors that we explicitly track in our unit and regression tests. These errors are typically limited to rare or unusual forms. Two examples, both from the *Siddhāntakaumudī*:

- **सुषुषुपतुः** — This is the third-person dual perfect of **ञिष्वपँ शये** as combined with the prefix **सु**. Currently, our program does not implement rule 8.3.88 (**सुविनिर्दुर्भ्यः सुपिसूतिसमाः**), so it incorrectly applies **षत्व** to **स्वप्** after **द्वित्व** and derives the wrong form **सुसुषुपतुः**.
- **सखा** – This is the masculine nominative singular form of **सखी** as derived from **सखि** + **क्यच्** + **किँप्**. Our current logic does not apply 7.1.93 (**अनङ् सौ**), so we instead derive the wrong form **सखीः**. **सखी** should not be confused with the much more common **सखि**, whose forms we derive correctly.

Errors like these are eminently fixable, and we can do so safely given our extensive test suite.

7 Applications

Our program's errors are currently limited to rare forms, and these errors are decreasing over time as our test suite expands. Given this work's current standing and future trajectory, we

think it is reasonable to explore some applications of its output.

7.1 As a word generator

We envision using the output of our word generator in one of two ways. Software with limited resources can use our program as-is and generate words on demand as requested by the user. For software with more resources, we have found that a finite state transducer (FST) is highly effective at storing Sanskrit words in a space-efficient way. Briefly, an FST generalizes the prefix tree and suffix tree into a single data structure, such that items that tend to share prefixes and suffixes can be stored with less memory. In one early experiment, we found that we could store tens of millions of Sanskrit words and their basic properties (person, number, etc.) at an amortized cost of roughly one byte per word. For details on this data structure and the specific Rust library we use, we refer the reader to Gallant (2015).

7.2 As a *prakriyā* generator

As a *prakriyā* generator, our program has obvious relevance to anyone who wants to understand how the *Aṣṭādhyāyī* might derive a specific word form. But there are applications beyond this narrow scope as well.

One tool we envision is an electronic grammar interface that is analogous to an electronic dictionary. Suppose a student could enter a word into a search interface and see a rule-by-rule grammatical breakdown of that word. With some translations of key rules, the same interface could offer explanations to students without a grounding in traditional grammar and thus comment on important high-level features in a user-friendly way. In doing so, such a tool could help ameliorate one of the key problems that Sanskrit students face: understanding the structure and function of a given word.

7.3 As a reference implementation of the *Aṣṭādhyāyī*

As our program progresses, we expect that others can start to use it as a reference implementation of the *Aṣṭādhyāyī*. By this, we mean that our program might become a baseline for new implementations or an experimental tool for answering certain kinds of grammatical questions. For example, someone who wishes to model conflict resolution more explicitly might adapt our API and test suite to a new core implementation. Or, a user might examine the impact of tweaking a rule's position or definition by running our program against our full test suite.

Again, we stress that our program takes no perspective on *anuvṛtti* or conflict resolution, and it is unclear what value our program could offer for exploring them, if any.

8 Future work

`vidyut-prakriya` offers several promising directions for future work.

The most promising direction is to continue implementing the rules of the *Aṣṭādhyāyī* and its secondary texts, provided that those rules are relevant for generating words. Our main ambition is to implement all such rules, including Vedic rules (i.e. marked with **छन्दसि**, **मन्त्रे**, **ऋचि**, and so on), the rest of the *Uṇādipāṭha*, and the Phiṭ Sutras.

Another direction is to tweak our *prakriyā*s and rule order to better conform to the conventions of modern grammarians. For example, our system derives **बभूव** by applying rule 6.1.8 (**लिटि धातोरनभ्यासस्य**) before the introduction of **वुँक्-आगम** by rule 6.4.88 (**भुवो वुग्लुङ्लिटोः**), which we have heard is unusual. Having a robust test case helps us make these kinds of changes with confidence.

A third direction is to add support for non-Pāṇinian usages like **उप** + **आस्** + **त्वा** → **उपासित्वा** (Pāṇinian **उपास्य**),[9] which commonly occur in the Itihasas.

A fourth direction is to allow more user control over our program's execution flow. Our program allows this already to a limited extent by allowing the user to disable certain optional

[9]This form is allowed by 7.1.38 (**त्वाडपि छन्दसि**), but our program treats such a derivation as Vedic and does not model that the form is also allowed in the Itihasas.

rules. With tighter code discipline, we might be able to extend this behavior to all other rules as well. Our most ambitious idea in this direction is to use Rust's rich macro system to inspect our existing rules and reorder them according to whatever criteria the user desires. Unfortunately, we expect that supporting this functionality properly would require a near-total rewrite of our code and its rules, which means that this direction is probably best served by an entirely new project.

9 Conclusions

This paper has presented a new Pāṇinian word generator that we developed by focusing first on producing a strong program. Our generator's speed and performance support a large test suite, which in turn permits faster progress on implementing rules. Over time, we expect to continue increasing our program's test coverage and improving the quality and range of its output.

Despite its limitations, we believe that `vidyut-prakriya` represents a major step forward for Sanskrit word generators, and we look forward to creating a program that generates all Pāṇinian words.

Acknowledgements

The authors wish to thank Neelesh Bodas for feedback on specific *prakriyā*s, and for his work on the invaluable `ashtadhyayi.com`; Dhaval Patel for guidance on the design of SanskritVerb; Madhav Deshpande for his comments and guidance; Shreevatsa Rajagopalan for his work on `vidyut-prakriya`'s WebAssembly build and online demo, and for help and advice in preparing this paper; Vikram Bhaskaran, Prasanna Venkatesh T S, Yash Khasbage, and Sourya Kakarla for their other contributions to `vidyut-prakriya`; and Shruthi Raghuraman for her love and support. All errors are my own.

References

Akshar Bharati, Amba Kulkarni, V Sheeba, and Rashtriya Vidyapeetha. 2006. Building a wide coverage Sanskrit morphological analyzer: A practical approach. 01.

Neelesh Bodas. 2023. Ashtadhyayi. `https://ashtadhyayi.com`. Accessed: 2023-09-28.

William Bugden and Ayman Alahmar. 2022. Rust: The programming language for safety and performance.

Andrew Gallant. 2015. Index 1,600,000,000 keys with automata and Rust. `https://blog.burntsushi.net/transducers/`. Accessed: 2023-10-01.

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

Pawan Goyal, Amba Kulkarni, and Laxmidhar Behera. 2009. Computer simulation of Aṣṭādhyāyī: Some insights. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 139–161, Berlin, Heidelberg. Springer Berlin Heidelberg.

Oliver Hellwig. 2009. SanskritTagger: A stochastic lexical and POS tagger for Sanskrit. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 266–277, Berlin, Heidelberg. Springer Berlin Heidelberg.

Anand Mishra. 2009. Simulating the Pāṇinian system of Sanskrit grammar. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 127–138, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dhaval Patel and Shivakumari Katuri. 2015. Prakriyāpradarśinī - an open source subanta generator. 16th World Sanskrit Conference.

Dhaval Patel. 2023. SanskritVerb. `https://github.com/drdhaval2785/SanskritVerb`. Accessed: 2023-10-01.

S. Prasanna. 2022. Spellchecker for Sanskrit: The road less taken. In Md. Shad Akhtar and Tanmoy Chakraborty, editors, *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 290–299, New Delhi, India, December. Association for Computational Linguistics.

Jivnesh Sandhan, Rathin Singha, Narein Rao, Suvendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022. TransLIST: A transformer-based linguistically informed Sanskrit tokenizer.

Peter M. Scharf and Malcolm D. Hyman. 2012. Linguistic issues in encoding Sanskrit.

Peter Scharf. 2015. An XML formalization of the Aṣṭādhyāyī. 16th World Sanskrit Conference.

Anuprāsa Identifier and Classifier: A computational tool to analyze Sanskrit figure of sound

Amruta Barbadikar
Department of Sanskrit Studies,
University of Hyderabad
amruta.barbadikar@gmail.com

Amba Kulkarni
Department of Sanskrit Studies,
University of Hyderabad
ambakulkarni@uohyd.ac.in

Abstract

Anuprāsa is a *śabdālaṅkāra* (figure of sound), in which the poetry is embellished by the repetitive occurrence of letters.[1] The task of processing the decorative language consisting of such figures is a path not explored in the field of Sanskrit computational linguistics. This paper discusses a tool that identifies and classifies *anuprāsa alaṅkāra*. *Anuprāsa*, being a figure of sound, makes the least use of semantics. This tool is essentially developed upon the insights taken from the school of *alaṅkāra*s, especially from the treatise of ĀCĀRYA VIŚVANĀTHA from 14th century AD.

1 Introduction

There is a varied scope for research in the field of Sanskrit computational linguistics. Segmentation(Goyal and Huet, 2013), Morph-analysis and Generation (Kulkarni and Shukl, 2009), Compound type analysis (Kulkarni and Kumar, 2013; Kulkarni and Kumar, 2011) and Generation (Satuluri and Kulkarni, 2013), Sentence analysis (Goyal et al., 2009; Kulkarni et al., 2020) and Generation (Kulkarni and Pai, 2019), Discourse analysis (Kulkarni and Das, 2012), Translation (Agrawal and Madaan, 2020), etc. tasks are being carried out with extensive efforts. Most of the tasks are grammar oriented. Other works which attempt to process the rhetoric of the poetic language are very limited. Except for the Meter identification (Melnad et al., 2015; Rajagopalan, 2018; Neil, 2023; Terdalkar and Bhattacharya, 2023) and a tool to identify and classify the *yamaka alaṅkāra* (Barbadikar and Kulkarni, 2023), the Natural Language Processing (NLP) tools are far away from processing poetic beauty in Sanskrit.

Although, some examples in languages other than Sanskrit for processing figurative language can be found. Shutova (2011) presented a computational approach to process metaphor using statistical methods. Englard (2013) used the rhetorical analysis of text to predict the author. For Hindi, Audichya and Saini (2021), worked out the *alaṅkāra*s in Hindi to present the hierarchical structure with a taxonomical listing of *alaṅkāra*s. However, the computational implementation was not exercised. Naaz and Singh (2022) were able to contribute by presenting three different tools for Hindi. 'Text2Mātrā' produces the *laghu* and *guru mātras* for the input, 'RPaGen' detects the rhyming quality of the poem and 'FoSCal' generates a score according to the quantity of *anuprāsa* used over the poem. For Sanskrit, the automatic Meter Identification task has been worked out, from various applications perspectives, by different scholars. Melnad et al. (2015), Rajagopalan (2018), Neil (2023), Terdalkar and Bhattacharya (2023) are some of the notable contributors to the available state-of-the-art Sanskrit Meter Identification systems.

Highly complex language structures, use of intended and implicit meaning, multiple meanings of a word, multiple words having a similar meaning, and unavailability of useful state-of-the-art tools are the factors that discourage the processing of decorative language used in poetry. The tradition of *alaṅkāraśastra* (poetics) is developed over a long period ranging from the 1st century AD. The study of figures of speech is an important stream of this tradition. Figures of speech

[1]Here, the repetition of sounds is desirable. In Sanskrit, there is one to one mapping of sound with the denoting letter and we compute letters, not sounds. Hence we use the terms letters and sounds interchangeably.

are employed by the poets to enhance the beauty of the poetry. Even in the Vedic literature, the use of such devices can be traced.

ĀCĀRYA BHARATA from 1st century AD, known as the father of Indian poetics, in his treatise NĀṬYAŚĀSTRA, describes only 4 *alaṅkāra*s. Whereas in KUVALAYĀNANDA of APPAYA DĪKṢITA (16th century AD) 125 types of *alaṅkāra*s are enlisted. Sanskrit has a rich tradition of poetics that is 2000 years old starting from BHARATA'S NĀṬYAŚĀSTRA. There are six primary schools of poetics in Sanskrit viz. *rasa, alaṅkāra, rīti, dhvani, vakrokti* and *auchitya*. The school of *alaṅkāra* is one of the most cherished schools. It is the ornamentation of poetry through the specific arrangement of syllables or words or astonishing meanings to enhance the rhetorical effect. We aim to concentrate on the computational analysis of the provided text essentially with the *alaṅkāra* point of view.

According to the school of *alaṅkāra, alaṅkāra*s are mainly of two types viz. *śabdālaṅkāra* (figures of sound) and *arthālaṅkāra* (figures of speech). The combination of these two is called as *ubhayālaṅkāra*. The count of *alaṅkāra*s differs from scholar to scholar. Approximately the count exceeds the number of 50. Some *alaṅkāra*s that use the phonetic or structural beauty may be easier to identify, but others would be quite tricky to recognize even for the experts in this field because of the involvement of deeper semantics. As we aim at dealing with these *alaṅkāra*s from the computational point of view, it is feasible to identify different syntactic constructions without considering the semantics in *śabdālaṅkāra*s like *yamaka* and *anuprāsa*. This research is aimed at the identification and classification of *anuprāsa* without considering the meaning.

It is a non-trivial task to analyze highly semantic and aesthetically rich texts without the help of machine learning or any advanced techniques of NLP. Like the Indian grammatical tradition, the rhetoric tradition has provided a robust theory upon which a foolproof rule-based system can be built. Hence, we relied upon a rule-based approach to accomplish this task. As we are dealing with *śabdālaṅkāra*, it allowed us to ignore the sense of the poetry making this task easier. We employ a simple rule-based algorithm after extracting various syntactic clues from the school of *alaṅkāra*s. For classification purpose, we select the best and most convincing, inclusive scheme proposed by VIŚVANĀTHA, a prominent scholar in the tradition of *alaṅkāraśāstra*.

2 *Anuprāsa*

Anuprāsa is a *śabdālaṅkāra*. Essentially, it is the repetition of consonants. This repetition should be in proximity such that one should remember the prior instance.[2] The phenomenon of *anuprāsa* is similar to Alliteration.[3]

Like *Yamaka*, *anuprāsa* holds an important place in the *alaṅkāra*s. In the tradition of *alaṅkāraśāstra*, *anuprāsa* was originally introduced as a subtype of *yamaka* viz. *mālā yamaka*.[4] *Yamaka* is a repetition of the longer sequence patterns of syllables engaged in the poetry (Barbadikar and Kulkarni, 2023), especially in metrical verses, whereas in *mālā yamaka* repetition of consonants is considered, which is similar to *anuprāsa*. In *yamaka* where repetition of the longer patterns are engaged in the poetry especially the metrical verses when employed in a more complex way, this might create a hindrance in the process of experiencing *rasa*.[5] Because the meaning is different in each repetition, the listener might find it difficult to understand the meaning of the complete verse and lose interest. But *anuprāsa* is considered to be a contributor to the emergence of *rasa*. *Anuprāsa* can be traced in any type of text suggesting any kind of *rasa*.

---

[2] *pūrvānubhavasaṃskārabodhinī yadi adūratā* //1.55, KĀVYĀDARŚA

[3] In alliteration, consonant sounds in two or more neighbouring words or syllables are repeated. The repeated sounds are usually the first, or initial, sounds as in "seven sisters", but repetition of sounds in non-initial stressed, or accented, syllables is also common: "appear and report."-"Alliteration." `https://www.merriam-webster.com/dictionary/alliteration`. Merriam-Webster, 2023.

[4] *nānārūpaiḥ svarairyuktaṃ yatraikaṃ vyañjanaṃ bhavet /*
tanmālāyamakaṃ nāma vijñeyaṃ paṇḍitairyathā//*16.84*, NĀṬYAŚĀSTRA

[5] *tadetatkāvyāntargaḍubhītam*/ in the vṛtti of 83rd kārikā, 9th chapter, KĀVYAPRAKĀŚA

3 The conceptual development of the types of *anuprāsa*

Anuprāsa is independent of the form of poetry, that is, it is used in prose format also, like in KĀDAMBARĪ of BĀṆABHAṬṬA. The classifications found are based on the variations in repetitions in terms of the categories of repeated consonants, number of repetitions, number of repeated consonants and mood emergence due to the combination of different consonants. Hence, we do not observe many variations in the definition and classification, but the number of subtypes considered varies.

In this section, we present a brief overview of various types of *anuprāsa* furnished by different scholars. BHARATA, known as the first scholar of the tradition, has enlisted only four *alaṅkāra*s namely *yamaka, upamā, rūpaka* and *dīpaka* among which *yamaka* was the only figure of sound. *Mālā yamaka*, a subtype of *yamaka* can be considered as the inspiration behind *anuprāsa*. Example of *mālā yamaka* given by BHARATA is,

> *asau hi rāmā rativigrahapriyā*
> *rahaḥpragalbhā ramaṇaṃ manogatam /*
> *ratena rātriṃ ramayet pareṇa vā*
> *na cedudeṣyattaruṇaḥ paro ripuḥ* //16.86, NĀṬYAŚĀSTRA

Other scholars after BHARATA considered *anuprāsa* as an individual *alaṅkāra* and classified it from different perspectives. BHĀMAHA provided only two types, whereas BHOJA extended the count to 6 types. Some scholars being excessively analytical tried to increase the count even more.

BHĀMAHA (6^{th} century AD) for the first time put forward *anuprāsa* as a separate *alaṅkāra* in his treatise KĀVYĀLAṄKĀRA. BHĀMAHA declares the arrangement of similar letters as *anuprāsa*.[6] Moreover, he provides two types of *anuprāsa*. One is *grāmyānuprāsa*. As the name suggests, the repetition of letters without any pattern or elegance is *grāmyānuprāsa*. Learned people assume it as an ordinary repetition.[7] For example,

> *sa lolamālānīlālikulākula galo balaḥ* / 2.6, KĀVYĀLAṄKĀRA

The another type is *lāṭānuprāsa*. It is a repetition of a complete *pada* (word). But the meaning of *pada* does not change. For example,

> *dṛṣṭiṃ dṛṣṭisukhāṃ dhehi candraścandramukhoditaḥ* / 2.8, KĀVYĀLAṄKĀRA

'Lāṭa' is a name of a geographical region. Poets belonging to the region 'Lāṭa' used to employ this kind of repetition in plenty. Most of the scholars in the tradition included *lāṭānuprāsa* in the classification of *anuprāsa*. *Lāṭānuprāsa* shows similarity with *yamaka alaṅkāra*. The only difference is that, in *yamaka* the repeated word or the sequence of sounds should possess different meanings in each repetition.

After BHĀMAHA, DAṆḌIN (8^{th} century AD) added the clause of proximity to *anuprāsa*. According to DAṆḌIN the repetition of letters such that the listener remembers the previous occurrence of the repeated letter is called *anuprāsa*.[8] He added that the repetition in *anuprāsa* nourishes the *rasa*.

UDBHAṬA (9^{th} century AD) introduced *chekānuprāsa* in his work KĀVYĀLAṄKĀRA-SĀRA-SAṄGRAHA (8^{th} AD). *Chekānuprāsa* is one repetition of two groups of consonants.[9] Here, the sequence may not be the same. The example given is as follows,

> *sa devo divasān ninye tasmin śailendrakandare /*
> *gariṣṭhagoṣṭhī-prathamaḥ pramathaiḥ paryupāsitaḥ* // 3.3, KĀVYĀLAṄKĀRA-SĀRA-SAṄGRAHA

[6] sarūpavarṇavinyāsamanuprāsaṃ pracakṣate| 2.5, KĀVYĀLAṄKĀRA
[7] grāmyānuprāsamanyttu manyante sudhiyo'pare| 2.6, KĀVYĀLAṄKĀRA
[8] pūrvānubhavasaṃskāra-bodhinī yadyadūrata| 1.55, KĀVYĀDARŚA
[9] chekānuprāsastu dvayordvayoḥ susadṛśoktikṛtau| 3.2, KĀVYĀLAṄKĀRA-SĀRA-SAṄGRAHA

In addition to this, *vṛttyanuprāsa* was also defined. Here, the combination of the repeated consonants is considered. According to it, three *vṛttis* are defined viz. *paruṣa* (harsh), *upanāgarikā* (soft) and *grāmyā* (other than the prior two). Again, the count and the definitions of *vṛttis* vary from scholar to scholar.

BHOJA from 11th century conducted a vast review on this *alaṅkāra*. In addition to *vṛttyanuprāsa* and *lāṭānuprāsa* he added 4 more types viz. *śrutyanuprāsa*, *varṇānuprāsa*, *padānuprāsa* and *nāma-dvirukti*. *Varṇānuprāsa* is similar to *chekānuprāsa*. Also, *padānuprāsa* and *nāma-dvirukti* can be included into *lāṭānuprāsa*.

JAYADEVA (12th century AD) in CANDRĀLOKA introduced two types of *anuprāsa* viz. *sphuṭānuprāsa*, which is the repetition of consonants within a *pāda* or a half of a *pāda* and *arthānuprāsa*, where the repetition of consonants occur in the two words which are connected semantically. For example,

> *candanaṃ khalu govinda-caraṇa-dvandva-vandanam*/ 5.6, CANDRĀLOKA

Here, the two words '*candanam*' and '*govind-caraṇa-dvandva-vandanam*' are connected with the 'upamāna-upameya' relation and possess the repetition of consonants '*nd*'.

The criteria for classifying *anuprāsa* into different types is basically the number of repetitions, consideration of the order of the repeated letters and what letters are being repeated. Focusing on these points different classifications are framed. Due to such finite dimensions of classification, we observe a limited number of types. Also, similar kinds of types are explained in various other classification schemes.

For the tool development, we follow one comprehensive classification. VIŚVANĀTHA'S *anuprāsa* classification provided in his treatise SĀHITYADARPAṆA has five classes that cover the extract of all the other interpretations available. Categories proposed by the rhetoricians like DAṆḌIN, UDBHAṬA, VĀMANA, RUDRAṬA, MAMMAṬA, JAYADEVA, BHOJA, etc. are covered under the umbrella of the classification of VIŚVANĀTHA.

4 Viśvanātha's classification

The tenure of VIŚVANĀTHA (14th century AD) comes in the later part of the tradition of *alaṅkāraśāstra*. He provides a well-defined and comprehensive theory for the classification of *anuprāsa* which facilitates the clarity for implementation. VIŚVANĀTHA'S classification includes other prominent classifications. Moreover, it uses widely accepted nomenclature. According to him, the *anuprāsa* is classified into 5 sub-classes. The examples for these 5 types are taken from 10th *pariccheda* of SĀHITYADARPAṆA.

1. ***Chekānuprāsa***
   Chekānuprāsa is the double occurrence of consonants with the same sequence. In each repetition, vowel endings may vary. In the example given below, one repetition of '*n-d-h*', '*v-r*' and '*p-v-n*' is in the same order. That means the order of the repeated consonants is not changed. The repetition of '*v-r*' is not changed to '*r-v*' irrespective of the changing vowels in between.

   > *ādāya bakulagandhānandhīkurvan pade pade bhramarān /*
   > *ayameti mandamandaṃ kāverīvāri-pāvanaḥ pavanaḥ//*

2. ***Vṛttyanuprāsa***
   Vṛtti is the mood or emotion. It is defined as the arousal of a specific mood resulting from a certain combination of letters. Repetition of one or many consonants in any order to produce a specific mood (*vṛtti*) is called *vṛttyanuprāsa*. The emotional effect differs according to the repetitive sound pattern and the combination of the letters used. This effect should complement the actual *rasa* of the poetry.

unmīlanmadhugandhalubdhamadhupavyādhūtacūtāṅkura-
krīḍatkokilakākalīkalakalairudgīrṇakarṇajvarāḥ /
nīyante pathikaiḥ kathaṃ kathamapi dhyānāvadhānakṣaṇa
prāptaprāṇasamāsamāgamaraṣollāsairamī vāsarāḥ //

In this example, the first foot has multiple repetitions of the consonant '*dh*'. In the second foot, there is repetition of the consonants '*k*' and '*l*' in any order. The third foot has the repetition of '*dh*' only once. The last foot has the repetition of '*p*', '*r*', '*s*' and '*m*' in different orders.

3. ***Śrutyanuprāsa***
Śrutyanuprāsa is the repetition of a group of consonants with a similar manner of articulation. According to DAṆḌIN it also is beneficial to the *rasa*.[10] These are further sub-classified into five classes according to the place of articulation.

(a) *Kaṇṭhya* (Velar) - *{k, kh, g, gh, ṅ, h}*
(b) *Tālavya* (Palatal) - *{c, ch, j, jh, ñ, y}*
(c) *Mūrdhanya* (Retroflex) - *{ṭ, ṭh, ḍ, ḍh, ṇ, r, ṣ}*
(d) *Dantya* (Dental) - *{t, th, d, dh, n, l, s, v}*
(e) *Oṣṭhya* (Labiel) - *{p, ph, b, bh, m, v}*

For example, the following verse

dṛśā dagdhaṃ manasijaṃ jīvayanti dṛśaiva yāḥ /
virūpākṣasya jayinīstāḥ stumo vāmalocanāḥ //

has a repetition of Palatal varṇas '*j*' and '*y*'.

4. ***Antyānuprāsa***
Repetition of syllables at the end of the *padas* (words) or at the end of the foot. Specifically, after the penultimate vowel that is the last but one vowel of the *pāda* or *pada*. the For example,

keśaḥ kāśastabakavikāsaḥ kāyaḥ prakaṭitakarabhavilāsaḥ /
cakṣurdagdhavarāṭakakalpaṃ tyajati na cetaḥ kāmamanalpam //

In this example, the ends of the first and second feet match, similarly the ends of the third and fourth feet.

5. ***Lāṭānuprāsa***
Lāṭānuprāsa is not just the repetition of the consonants but the repetition of a word (*pada*) with similar meaning but different implications. This kind of *anuprāsa* is similar to *yamaka* as repetitions are considered for longer syllable sequences. Following is an example of *lāṭānuprāsa*.

smerarājīvanayane nayane kiṃ nimīlite /
paśya nirjitakandarpaṃ kandarpavaśagaṃ priyam //

In the above given example, the words '*nayane*' (meaning eyes) and '*kandarpam*' (meaning desire for love) are repeated with the same sense. The words are sometimes an independent word or a part of a compound word. According to the role of the word in the sentence, the implication changes. The first appearance of *nayane* in *smerarājīva*nayane (meaning - a woman with lotus like eyes) is in the form of an element of the compound, and contributes its meanings to form a meaningful compound. The other occurrence of '*nayane*' is an independent word to give the meaning as 'two eyes'. Similarly in '*kandarpam*', both repetitions possess the same meaning but the implication differs in each occurrence.

[10] *yayākayācicchrutyāyatsamānamanubhūyate/*
tadrūpāṃhipadāsattiḥsānuprāsārasāvahā// 1.52, KĀVYĀDARŚA

Type	Count	Order	Unit	Position in the input
lāṭa	≥ 2	same	word	next to each other or with a few interventions.
cheka	2	same	sequence of syllables without vowels	anywhere within a proximity of $8 + 2*$ length of syllable sequence.
vṛtti	a)> 2 b)≥ 2	any	a) sequence of syllables without vowels b) a consonant	anywhere within a proximity of $8 + 2*$ length of syllable sequence.
śruti	≥ 2	any	syllables from the same class	within a proximity of 8 syllables.
antya	≥ 2	same	syllables after the second last vowel.	end of feet and words.

Table 1: Differentiation from the implementation point of view

5 Implementation of Anuprāsa Identifier and Classifier

From the definitions of various *anuprāsa*s, we note that some types have more stringent conditions than others. Hence, the examples that satisfy more stringent conditions may also satisfy less stringent conditions and thus can be categorised under two different types of *anuprāsa*. For example, *lāṭānuprāsa* demands that the repetition is of words and not syllables. *Chekānuprāsa* demands the repetition of syllables in the same order. Thus, any example of *lāṭānuprāsa* is also potentially an example of *chekānuprāsa* as well. However, due to the stringent conditions of *lāṭānuprāsa*, it is appropriate to classify such an instance only under *lāṭānuprāsa*. A similar situation exists with other pairs as well. In order to decide the proper exclusive sequence in which these *anuprāsa*s should be identified, we look at the necessary conditions for each of them.

From the table 1, we understand that the natural order for identifying the *anuprāsa* type is *lāṭa, cheka* and *vṛtti*. The conditions of *śruti* and *antya* type of *anuprāsa* do not clash with any other classes and hence can be identified either in the beginning or at the end.

We use the frequencies of n-grams[11] for identifying *lāṭānuprasa* and frequencies of n-grams of the sequence of letters ignoring the vowels and their positions identifying the *chekānuprāsa* and the frequencies of n-grams of consonants for identifying *vṛttyanuprāsa* and only consonants having the same place of articulation and their positions in the input for identifying *śrutyanuprāsa*. The *pādāntyānuprāsa* is identified by looking at rhymes at the end of the *pādas*. For *padāntyānuprāsa*, we consider the rhyming in the space-separated word endings.

Unicode Devanagari is unsuitable for processing and identifying the n-gram and consonant frequencies since the basic units in Unicode are a mix of consonants with a vowel *'a'* inherent in them. Hence we convert the input internally into WX notation[12] and process it. In addition to Devanagari and IAST schemes, we also accept input in various other transliteration schemes such as Velthuis, SLP, Kyoto Harward, WX notation, etc.

Normalization of the input is an important step in processing. To analyse *antyānuprāsa*, the *daṇḍas* ('|') and spaces to mark the word and the foot boundary are preserved. For other types, the normalization of various elements is defined below.

- Spaces :
  In the oral tradition, the spaces between the words do not carry any significance. *Anuprāsa* deals with the sound patterns, and as such, we ignore the spaces between the words.

- *Anunāsikyas* :
  Since the *anuprāsa* is identified based on sound patterns, the variations in spelling need to be taken care of. Sanskrit allows some spelling variations concerning nasalization. All the homogenous nasal stops are converted into *anusvāras*. The *anusvāra* when followed by consonants, can be converted into homogenous nasal stop viz. *ṅ, ñ, ṇ, n* and *m*.

[11]The sequences of letters of length 'n' are called n-grams.
[12]`https://en.wikipedia.org/wiki/WX_notation`

For example, *'aṃbuja'* versus *'ambuja'*, *'aṃka'* versus *'aṅka'*. Similarly, the nasal stop *'m'* at the end of a word is written as an anusvāra when it is followed by a word starting with a consonant. We normalize all the nasal stops to *anusvāra*.

- Special characters:
  (1) A special character that needs special attention is the *avagraha*. The *avagraha* is a writing convention to indicate the elided *'a'* during the *sandhi* operation. Since for the purpose of *anuprāsa* identification, we look at the *sandhi*ed text only, we ignore the *avagraha* if it is present in the input text.
  (2)Similarly, the *daṇḍa* ('|') used to denote the sentence-end, or in the case of a verse, to denote the end of two *pādas*. Except for *pādāntyānuprāsa*, the *daṇḍa* is also ignored.

The broad algorithm is as follows.

- Read the sequence of letters.
- Convert it to WX notation.
- Check for *pādāntyānuprāsa* by dividing the input into 4 equal parts and comparing the sequence of letters after the second last vowel at the end of each part.
- Check for *padāntyanuprāsa* by comparing the word endings from the penultimate vowel to the end of the consecutive words. If the sequence of letters matches at least in two words mark the repeated sequence as *padāntyānuprāsa*.
- Get the n-grams ($n \geq 2$) along with their positions with and without vowels.
- Remove all the small n-grams that can be subsumed by the large n-grams with matching positions (index).
- If the frequency of n-gram with vowels is more than 1, mark it as *lāṭānuprāsa*.
- Else if the frequency of n-grams without vowels is 2, mark such sequences as *chekānuprāsa*.
- Else if the frequency of n-grams ($n \geq 2$) without vowels is greater than 2 and if the frequency of single consonants is greater than or equal to 2 the n-gram or the consonants are marked as *vṛttyanuprāsa*.
- If the frequency of consonants belonging to the same class is greater than 2, mark them as *śrutyanuprāsa* of the type to which these consonants belong.

The use of else if ensures that the classification prefers a type with a more stringent definition than the others.

As a general rule in *anuprāsa*, the repetitions should not be far away to make the reader forget the previous occurrence. If the distance is large, the instance will not be able to produce amusement for the reader. To strike out such cases we have added one function in which the distance is calculated through the indices of the repeated consonants. For a single letter repetition, the maximum distance is considered to be 4 to 5 *akṣara*s, that is 8 to 12 letters approximately, considering the frequent conjuncts in Sanskrit.

6 Interface

We have designed a user-friendly interface to access this tool. This is an integrated tool for both *yamaka* and *anuprāsa* (see figure 1). User can provide their input text in various available encodings. Figure 1 shows all five types of *anuprāsa* highlighted in red colour corresponding to the input given by the user. The highlighted sequence facilitates the user with a better comprehension of the *alaṅkāra* and helps the user understand the difference between each type of *anuprāsa* effectively and easily. The interface is available in the 'tools' section at `https://sanskrit.uohyd.ac.in/scl/`

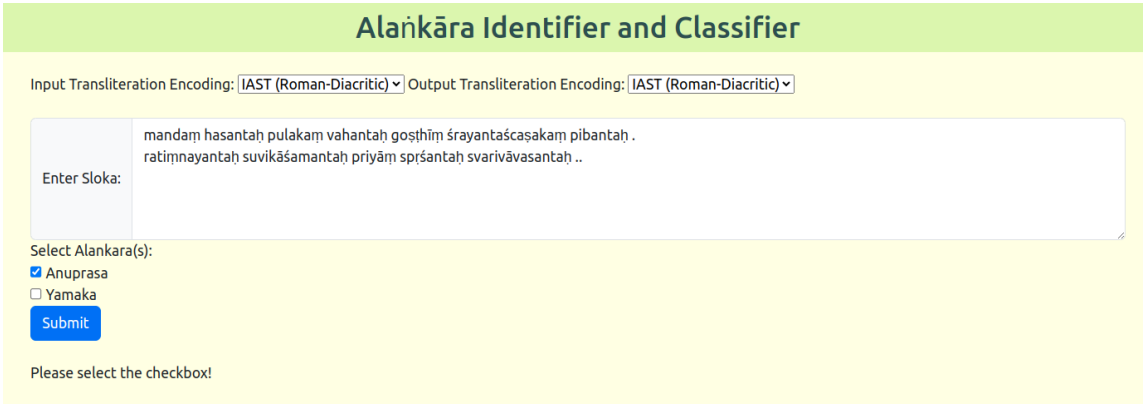


Figure 1: Alaṅkāra Identifier and Classifier: *Anuprāsa* input

Figure 2: Alaṅkāra Identifier and Classifier: *Anuprāsa* output

7 Evaluation

We tested our tool on a data set of 70 *ślokas* and 10 sample prose. The selected *ślokas* are primarily given as examples of *anuprāsa* in the *śāstr*ic texts and some from the RAGHUVAṂŚA of KĀLIDĀSA. The prose examples were passages consisting of 2 to 3 sentences from BĀṆABHAṬṬA'S KĀDAMBARĪ. Most of the examples contained more than one type of *anuprāsa*. This tool could successfully handle these *anuprāsa* instances. Since the three classes of *anuprāsa* viz. *lāṭānuprāsa*, *chekānuprāsa*, and *vṛttyanuprāsa* relax the conditions as we go from the first one to the third, the latter is a strict superset of the previous one. Hence, a pattern satisfying the conditions of *lāṭānuprāsa*, though it is an example of *chekānuprāsa* and *vṛttyanuprāsa*, is shown only under the *lāṭānuprāsa*. Similarly, the patterns satisfying the conditions for *chekānuprāsa* are not again displayed under *vṛttyanuprāsa*. Similarly, only those patterns that are not covered under *lāṭānuprāsa* and *chekānuprāsa*, are considered for *vṛttyanuprāsa*.

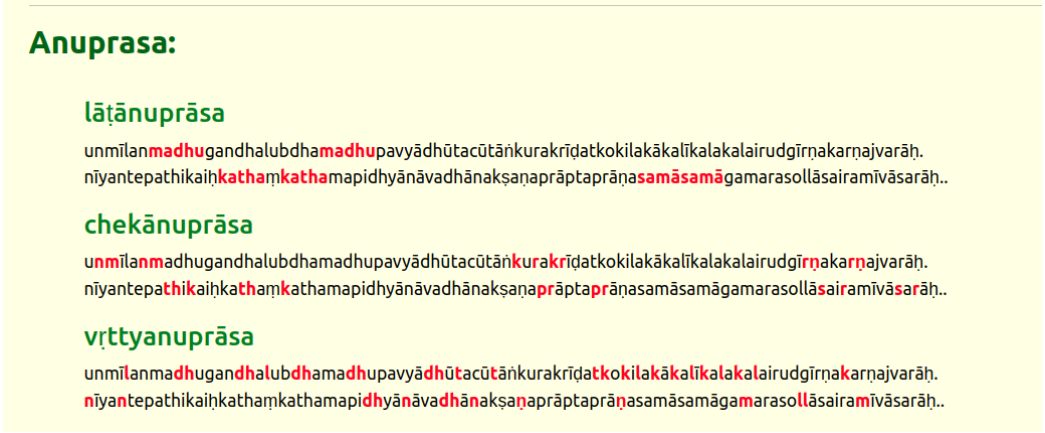


Figure 3: The cascade effect in *lāṭānuprāsa*, *chekānuprāsa* and *vṛttyanuprāsa*

Since this tool is not supported with word segmenter or meter identifier, for analysis of *antyānuprāsa* it completely relies on the spaces and the *daṇḍas* to mark the *pada* and *pāda* boundary. If the user has not provided the *daṇḍa* or spaces in the appropriate place, the tool is not able to identify *antyānuprāsa*.

8 Conclusion

We have discussed a tool useful for the identification and classification of Sanskrit poetry focusing on *anuprāsa*. This can be taken as a booster for figurative language processing in Sanskrit and other Indian languages as well. The concept of *anuprāsa* along with its classification is adopted by other Indian languages like Hindi, Marathi, Telugu, Kannada, etc. The same model with the necessary amendments can be deployed for the identification and classification of *anuprāsa* in modern Indian languages as well.

This module is extendable for other classifications presented in the tradition. The *vṛttyanuprāsa*, a type of *anuprāsa* can be researched extensively to identify *rasa* depending upon the repetition of clusters of consonants.

'Anuprāsa Identifier and Classifier' is useful for teaching this figure of sound by presenting a demonstration of different examples. While creating a masterpiece of poetry, a good poet does not deliberately enforce the figures in the poetry. Such upcoming masterpieces in Sanskrit can be tested with this tool.

References

Ācārya Viśveśvara. 2017. *Kāvyaprakāśa*. jñānamaṇḍala Limited, Varanasi.

Prateek Agrawal and Vishu Madaan. 2020. A Sanskrit to Hindi language machine translator using rule based approach. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON): System Demonstrations*, pages 13–15, Patna, India, December. NLP Association of India (NLPAI).

Milind Kumar Audichya and Jatinderkumar R. Saini. 2021. Towards Natural Language Processing with figures of speech in Hindi poetry. In *International Journal of Advanced Computer Science and Applications*.

V. Balasubramanyam. 2017. *Citram - Poetry of Sound*, volume 1. Rashtriya Sanskrit Sansktan, New Delhi.

Amruta Barbadikar and Amba Kulkarni. 2023. Yamaka identifier and classifier: A computational tool for the analysis of Sanskrit figure of sound (upcoming).

Satyadev Chowdhary. 1965. *Kāvyālaṅkāra*. Vasudev Prakashan, Delhi.

Durgaprasad and Kashinath Parab. 1886. *Kāvyālaṅkāraḥ*. Nirnayasagar Press, Mumbai.

Benjamin Englard. 2013. A rhetorical analysis approach to Natural Language Processing. In *ArXiv*.

Edwin Gerow. 1971. *A Glossary of Indian Figures of Speech*. Mouton & Co. N. V. Publishers, Hague.

Manomohan Ghosh. 1951. *The Nāṭyaśāstra*, volume 1. Asiatic Society of Bengal, Calcutta.

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

Pawan Goyal, Vipul Arora, and Laxmidhar Behera. 2009. Analysis of sanskrit text: Parsing and semantic relations. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, pages 200–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mari Hattori. 1997. On the rhyme (yamaka) in Sanskrit poetics. *Annals of the Bhandarkar Oriental Research Institute*, 78(1/4):263–274.

Amba Kulkarni and Monali Das. 2012. Discourse analysis of Sanskrit texts. In *Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects*, pages 1–16, Mumbai, India, December. The COLING 2012 Organizing Committee.

Amba Kulkarni and Anil Kumar. 2011. Statistical constituency parser for Sanskrit compounds. In *ICON 2011*.

Amba Kulkarni and Anil Kumar. 2013. Clues from Aṣṭādhyāyī for compound type identification. In *5th international SCLS 2013*.

Amba Kulkarni and Madhusoodana Pai. 2019. Sanskrit sentence generator. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 1–13, IIT Kharagpur, India, October. Association for Computational Linguistics.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. In *the Festscrift volume of Bh. Krishnamoorty, Indian Linguistics*.

Amba Kulkarni, Pavankumar Satuluri, Sanjeev Panchal, Malay Maity, and Amruta Malvade. 2020. Dependency relations for Sanskrit parsing and treebank. In *Proceedings of the 19th International Workshop on Treebanks and Linguistic Theories*, pages 135–150, Düsseldorf, Germany, October. Association for Computational Linguistics.

Keshav Melnad, Peter Scharf, and Pawan Goyal. 2015. Meter identification of Sanskrit verse. In *Sanskrit Syntax: Selected Papers Presented at the Seminar on Sanskrit Syntax and Discourse Structures*.

Shriramachandra Mishra. 1996. *Kāvyādarśa*. Chowkhamba Vidyabhavan, Varanasi.

Komal Naaz and Niraj Kumar Singh. 2022. Design and development of computational tools for analyzing elements of Hindi poetry. In *IEEE Access*.

Tyler Neil. 2023. Skrutable: Another step toward effective Sanskrit meter identification. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*.

S. Rajagopalan. 2018. A user-friendly tool for metrical analysis of Sanskrit verse. In *Computational Sanskrit & Digital Humanities, Selected papers presented at the 17th World Sanskrit Conference*.

C. Shankara Rama Sastri. 1956. *Kāvyālaṅkāra of Bhāmaha*. The Sri Balamanorama Press, Mylapore, Madras.

Pavankumar Satuluri and Amba Kulkarni. 2013. Generation of sanskrit compounds. In *ICON 2013*.

Pandit Rangacharya Raddi Shastri. 1938. *Kāvyādarśa*. Bhandarkar Oriental Research Institute, Pune.

Shrikrishnamoori. 1909. *Kāvyālaṅkārasūtravṛttiḥ*. Sri Vani Vilas Press, Srirangam.

Ekaterina V. Shutova. 2011. *Computational approaches to figurative language*.

Krishnan Sriram, Amba Kulkarni, and Gérard Huet. 2023. Validation and normalization of DCS corpus and development of the 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), January. Association for Computational Linguistics.

Renate Söhnen. 1995. On the concept and presentation of "yamaka" in early indian poetic theory. *Bulletin of the School of Oriental and African Studies*, 58(3):495–520.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2023. Chandojnanam: A Sanskrit meter identification and utilization system. In *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*.

Rudradev Tripathi. 1972. *Sanskṛta Sāhitya me śabdālaṅkāra*. Shri Lal Bahadur Shastri Kendriya Sanskrit Vidyapeeth, Delhi.

Vrajaratnadas. *Kāvyādarśa*. Vrajaratnadas, Shrikamalamani Granthamaka Karyalaya, Kashi.

START: Sanskrit Teaching, Annotation, and Research Tool - Bridging Tradition and Technology in Scholarly Exploration

[1]**Anil Kumar,** [2]**N. Shailaja,** [3]**Amba Kulkarni**
Department of Sanskrit Studies
University of Hyderabad
{[1]anil.lalit22, [2]shailajanakkawar}@gmail.com,
[3]ambakulkarni@uohyd.ac.in

Abstract

Sanskrit, a language renowned for its profound literature and philosophical insights, has remained a cornerstone of ancient wisdom. In this digital age, the fusion of tradition and technology has led to the emergence of transformative tools such as the Sanskrit Teaching, Annotation, and Research Tool (START), being developed by the Department of Sanskrit Studies, University of Hyderabad. This research paper delves into the intricate features, methodologies, and implications of START in reshaping the landscape of Sanskrit research and teaching. By exploring its advanced annotation capabilities, collaborative potential and broader impact on the digital humanities, this paper demonstrates how START is redefining the boundaries of scholarly exploration and analysis.

1 Introduction

The timeless wisdom present within Sanskrit texts has fascinated scholars for generations. Therefore, in spite of the literature being several centuries old, there is a continuously increasing population interested in understanding these texts. Several online computational tools exist that assist the learners in understanding the Sanskrit texts and enhancing their grammar. The traditional commentaries in the form of a book are being gradually replaced by E-readers. These E-readers were developed semi-automatically for popular texts such as Saṅkṣepa Rāmāyaṇam, ŚrimadBhagvad Gītā and Śiśupālavadham.[1] Recently there was a similar effort to transform the Kramapāṭha reader of the Rāmopākhyāna to an E-reader with a search facility(Scharf and Chauhan, 2023), whhich was inspired by from its printed version. These E-readers are not only useful for the readers, but also are useful for building and enhancing NLP tools, since they provide an annotated gold data for ML techniques.

Saṁsādhanī and the Sanskrit Heritage Platform, though being used extensively for learning and teaching Sanskrit, and also for developing annotated gold data for various tasks such as segmentation, morph analysis, dependency parsing, etc. a need was felt to develop a better user interface and management of various tools and resources to cater to the diverse needs of the students, teachers, indology researchers, and computational linguists.

With the penetration of Digital Technology in every sphere of life, no wonder that the teaching and learning is affected the most. Sanskrit is no exception. During COVID-19, there was a surge in the learners of Sanskrit and this gave a boost to several online teaching programmes. The online computational tools for Sanskrit also saw an upward surge in its usage since many online courses either recommended these tools or the students in search of online help found them to be supplementary to their online courses.

One such course was also offered by Indian Institute of Technology, Roorkee in collaboration with Sanskrit Bharati where the online Sanskrit Computational Tools Saṁsādhanī and Sanskrit Heritage Platform were introduced to the students pedagogically. While the teachers followed the traditional methods of teaching, the students were encouraged to use the tools for completing the exercises. Pāṇini's grammar is useful for generating the word forms, joining the words following

[1]https://sanskrit.uohyd.ac.in/scl/e-readers

the sandhi rules, or deciding the appropriate case suffix for a noun, etc.. But for analysis and understanding the text, it is not directly useful, since now the rules are to be used in a reverse way which may also lead to non-determinism. It is here the students found these online platforms useful. However there were several practical difficulties students faced while using these tools. The current platform was not mobile friendly, while most of the students prefer to use the tools on smartphones rather than on a laptop or a desktop. Many of them did not have any access to computing devices other than a mobile. Second major problem was due to the multiple solutions provided by the machine. Readers would like to see only one output rather than many, in the beginning.

Similar course was offered again by the University of Hyderabad in 2022 to teach Sanskrit from scratch using these computational tools. The aim of this course was to see how complete these tools are from a pedagogical perspective. This course, followed by a workshop for Sanskrit teachers at the University of Hyderabad in 2023 helped us to identify the features teachers and students would like to have in the display.

Another use of these tools is to develop a dependency tree bank for Sanskrit. The Sanskrit parser produced several possible dependency trees. While in most of the cases only one or two edges of the graph would go wrong, the interface for correcting the solution was not user friendly and hence annotators faced problems. Problems were also due to the compatibility issues among the devices, browsers, operating systems etc.

Yet another need was of the Indology researchers and computational linguists working with knowledge base or grammar based approaches. They needed a search facility over the annotated data. The existing interface for E-readers is developed targeting the readers, and hence does not provide a good search facility to the researchers.

These limitations of the tools from user's perspective, and the requirement of a large size corpus for the use of Machine Learning approaches led us to think of a platform that can be used simultaneously for learning, teaching, corpus annotation and research.

In the next section, we survey various efforts in these areas. In the third section, we describe the platform START which is developed to cater to the needs described above, describing various utilities available. Finally we describe the architecture of the platform.

2 Related work

To the best of our knowledge there does not exist any single platform that caters to the needs of diverse users such as students, teachers, computational linguists and Indologists. However, there are platforms that are designed towards serving some of them. We brief a few important ones.

- **Perseus Digital Library:**[2] This is an open-source project mainly focussing on Greek, Latin and Arabic providing a suite of services for interacting with textual collections. It provides an integrated reading environment with support of linguistic tools such as lemmatizer and morphological analyser with a link to the dictionaries that helps a user in contextual reading. Sanskrit Library [3] and Heritage Platform[4] started with a similar services. Recently two websites Vedaweb[5] and Ambuda[6] were developed which provide similar services for Sanskrit as are provided by the Perseus website. The first one is focussed on Rigveda and the second one on classical Sanskrit texts.

- **The Sanskrit Heritage Platform:**[7] This platform provides various Sanskrit computational tools such as segmentation, morphological analysis and generation and a shallow

[2] http://www.perseus.tufts.edu/hopper
[3] https://sanskritlibrary.org
[4] https://sanskrit.inria.fr
[5] http://vedaweb.uni-koeln.de/rigveda
[6] http://ambuda.org
[7] http://sanskrit.inria.fr

parser with lexicon linked to the bilingual Sanskrit-French and Sanskrit-English dictionaries. In addition, it has a well curated semi-automatically segmented corpus with lexicon driven and linked morphological analysis. It also hosts various short lessons targeted towards the new learners of Sanskrit, with complete word level analysis linked to the dependency parser of University of Hyderabad.

- **Saṁsādhanī:**[8] This platform developed at the University of Hyderabad offers various computational linguistic tools such as sandhi joiner, sandhi splitter, morphological analyser and generator, sentential parser, and Sanskrit-Hindi Translation, and lexical resources such as Amarakośa and Dhātuvṛttis. This platform is linked to the services of Sanskrit Heritage Platform, and provides an integrated environment for analysis of Sanskrit texts complementing the services.

- **DCS:**[9] The Digital Corpus for Sanskrit website is designed for text historical research in Sanskrit linguistics and philology. Users can search for lexical units and their collocations in a huge corpus of about 650,000 lines which is sandhi split as well as morphologically tagged.

- **SanskritShala:**[10] It is the recent addition to this list which provides a Neural Sanskrit NLP Toolkit with Web-Based Interface for Pedagogical and Annotation Purposes(Sandhan et al., 2023).

- **Sangrahaka**: This is an open-source tool that supports collaborative annotation and is tailored for Indian languages, enhancing regional NLP efforts(Terdalkar and Bhattacharya, 2021).

There are numerous platforms available for text annotations such as GATE, BRAT, INCEpTION, WebAnno, Bella, etc. As we will see below, these platforms are not geared towards teaching and research. Their sole aim is to facilitate annotation.

- **GATE**[11]: An open-source platform for text engineering that supports various types of annotations, such as named entities, relations, events, opinions, and coreference. GATE also provides a graphical user interface, a scripting environment, and a plugin architecture for extending its functionality.

- **BRAT**[12]: A web-based tool for collaborative text annotation that can handle span entities, relations, and attributes. BRAT also allows for the configuration of custom annotation schemes and visualizations.

- **Doccano**: A web-based tool for text annotation that supports sequence labeling, text classification, and sequence-to-sequence tasks. Doccano also offers features such as user management, project management, and data import/export(Nakayama et al., 2018).

- **INCEpTION**[13]: A web-based tool for intelligent text annotation that integrates machine learning and active learning techniques to assist human annotators. INCEpTION supports various types of annotations, such as named entities, relations, events, concepts, and sentiments.

[8] https://sanskrit.uohyd.ac.in/scl
[9] `http://www.sanskrit-linguistics.org/dcs`
[10] `http://cnerg.iitkgp.ac.in/sanskritshala/`
[11] `https://gate.ac.uk/`
[12] `https://brat.nlplab.org/index.html`
[13] `https://inception-project.github.io/`

- **WebAnno**[14]: WebAnno is an open-source web-based tool for annotating text data in natural language processing (NLP) tasks. It offers customizable schemas, collaborative annotation, and integration with NLP pipelines for efficient labeling.

- **Bella**[15]: A JavaScript-based tool for natural language processing tagging that supports part-of-speech tagging, chunking, named entity recognition, and dependency parsing. Bella also allows for the creation of custom tag sets and rules.

2.1 Why a new Platform?

While there are several of these platforms that may be extended/adapted towards our goals, we decided to go for a totally new platform due to the following considerations.

- None of these platforms is really geared towards teaching, and building E-readers.

- At the backend we use rule based tools which produce all grammatically possible solutions, which are prioritized using some heuristics or probabilistic models. We would like to provide access to all these analyses should an annotator finds the displayed analysis to be wrong.

- We also would like to use the annotated data to develop E-readers automatically. For languages like Sanskrit, where one needs a good support of lexical resources and grammar tools to understand any Sanskrit text, the annotated texts come handy for the readers with all necessary information at one place.

These reasons motivated us to develop this platform.

3 START: Sanskrit Teaching, Annotation and Research Tool

START is a web based tool that provides

1. **A dashboard for learning and teaching Sanskrit**
   With the help of various online computational tools such as sandhi joiner and sandhi splitter, morphological analyser and generator, sentential analyser and generator, etc. available on Saṁsādhanī Platform integrated with Sanskrit Heritage Platform, this dashboard provides an environment for the learning as well as teaching. The user interfaces for each of these tools are developed following the pedagogical requirement of a teacher. We describe them in section 3.2 in detail.

2. **A dashboard for annotation and E-reader builder**
   This dashboard facilitates an annotator to annotate any Sanskrit text of his choice. The Saṁsādhanī and Sanskrit Heritage Platforms provide the back-end support. Detailed description of this is provided in section 3.1.

3. **User management System (UMS)**
   START provides a User Management System (UMS) which emerges as a fundamental tool that empowers START to seamlessly control user access, streamline administrative tasks, and ensure a smooth user experience. It provides User Registration and Onboarding, Role-Based Access Control, Password Management and Activity Monitoring and Auditing. Table 1 explains the roles.

4. **Encoding Management**
   The character encoding plays a crucial role in ensuring accurate representation and seamless exchange of textual information, especially in the case of Sanskrit, since it is written in several Indian scripts as well as specially designed Roman script, and also in various

[14] https://webanno.github.io/webanno/
[15] https://github.com/dennybritz/bella

Roles	Description
Admin	User Account Creation and Deletion, User Role and Permissions Management, User Support, Monitoring and Reporting
Editor	Group Creation and Management, Monitoring and Reporting
Annotator	Annotation Creation, Viewing Content, Monitoring and Reporting
Viewer	Viewing Content, Read-only Access, Monitoring and Reporting, Limited Interaction

Table 1: Different Roles in User Management System

transliteration schemes, especially in electronic format. The Encoding Management of START provides users to access the data in different Roman transliteration schemes such as WX, IAST, Velthuis, Harvard Kyoto, and also various Indian scripts such as Devanagari, Telugu, Oriya, etc. The encoding management ensures that the data gets stored uniformly in WX notation and is displayed in the script or transliteration scheme of user's choice, remembering the user's choice across sessions. (See Fig 1).

5. **Content Management**
   START also provides a robust Content Management system which is evolved to not only organise and display content but also provide users with the ability to add, delete, and edit data. With this facility, one can import the texts from existing websites, and then edit according to the version one is following.

6. **A dashboard for Indologists**[16]
   The data in the content management system and also the one annotated will also be accessible to the Indology researchers through this dashboard. This dashboard will have an advanced search facility with grammatical and lexical queries. Further, this data can also be exported to any of the standard formats such as University of Hyderabad tagged data format, or CONLLU format etc. for the use of Machine Learning algorithms.

The Mind-map of START is shown in Fig 2.
Now we describe the two dashboards in detail.

3.1 A dashboard for annotation and E-reader builder

This dashboard is mainly for annotating the data. Typically the Sanskrit texts are found in sandhied form, and need segmentation before we proceed for any annotation. Majority of popular texts are also in poetic/verse form. This also imposes another requirement for processing that the input text be marked with sentential boundaries providing a meaningful unit for dependency parsing. It is also necessary to get the prose form of the input text so that the translation into other languages becomes easier. The dashboard caters to all these needs. The dashboard provides access to various tools such as

- **Segmenter:** The Sanskrit Heritage segmenter is plugged in which provides the best possible segmentation(Sriram et al., 2023). The annotator can edit the text if the segmentation is wrong at any place (See Fig 3).
- **Word and Sentence analyser:** The Saṁsādhanī sentential parser is used to get the dependency parse of the segmented text. The best solution is provided in the form of a table with collapsible rows. For each word, the best morphological analysis in the context

[16]This is a part of the design, but is still not ready.

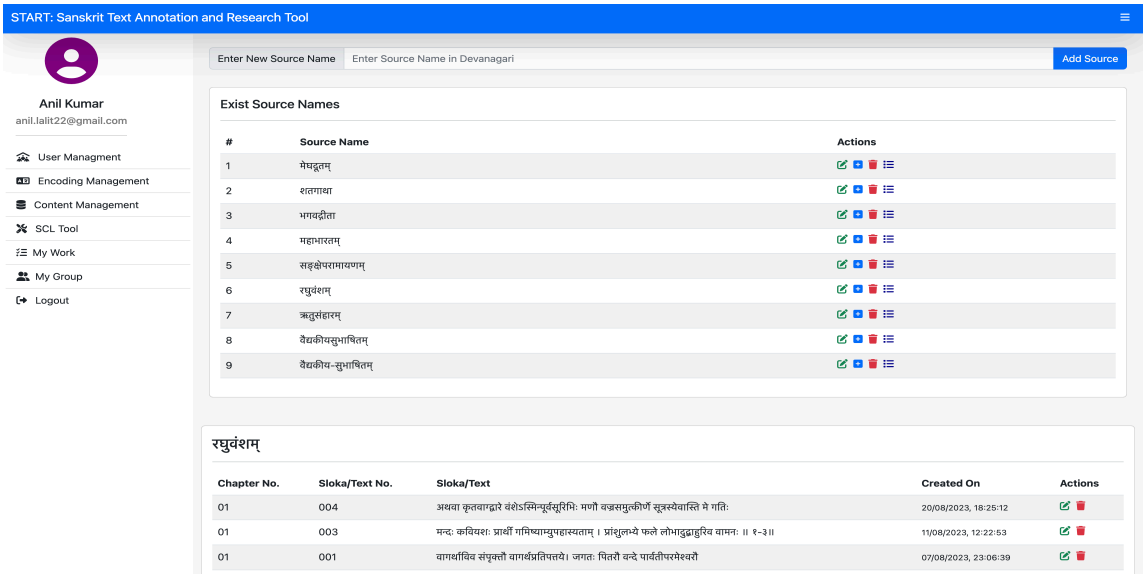


Figure 1: Content Management

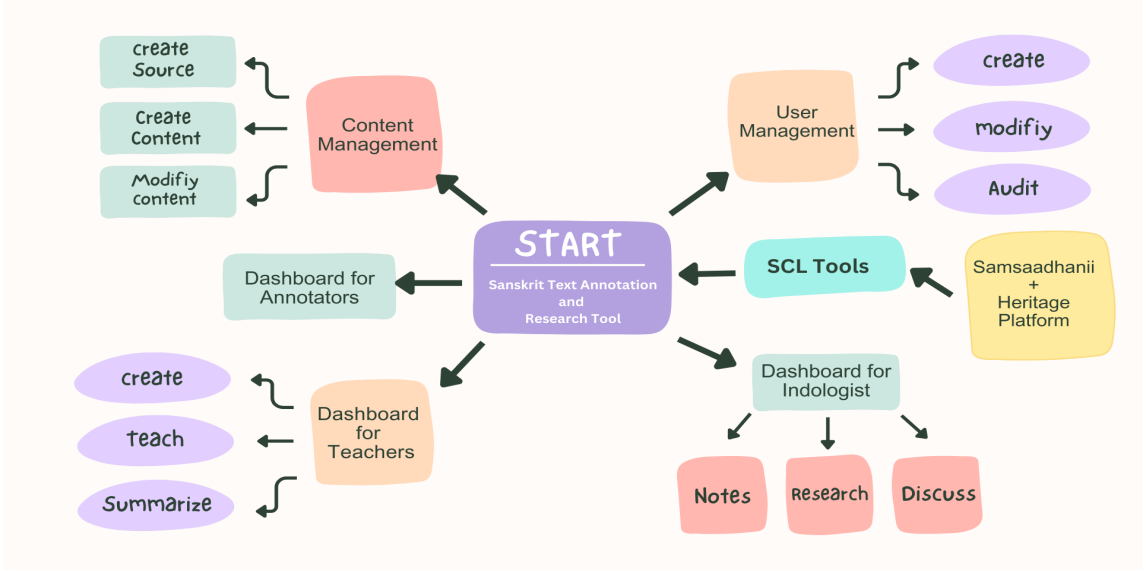


Figure 2: Mind-map of START

and its relation to other word in a sentence are provided, along with a link to the dictionary entry. The analysis generated at each stage is manually editable (See Figure 4). In the case of morphological analysis and dependency parsing, the annotator is provided with alternative possible analyses in the drop down menu from which s/he can select the correct one if the machine produced analysis is wrong. The dependency graph is displayed in the form of a tree, with colored nodes, color reflecting the grammatical category of the word. This way, the annotator can visualise the analyses and confirm her/his annotation (See Figs 4 and 5).

Figure 3: E-Reader Dashboard

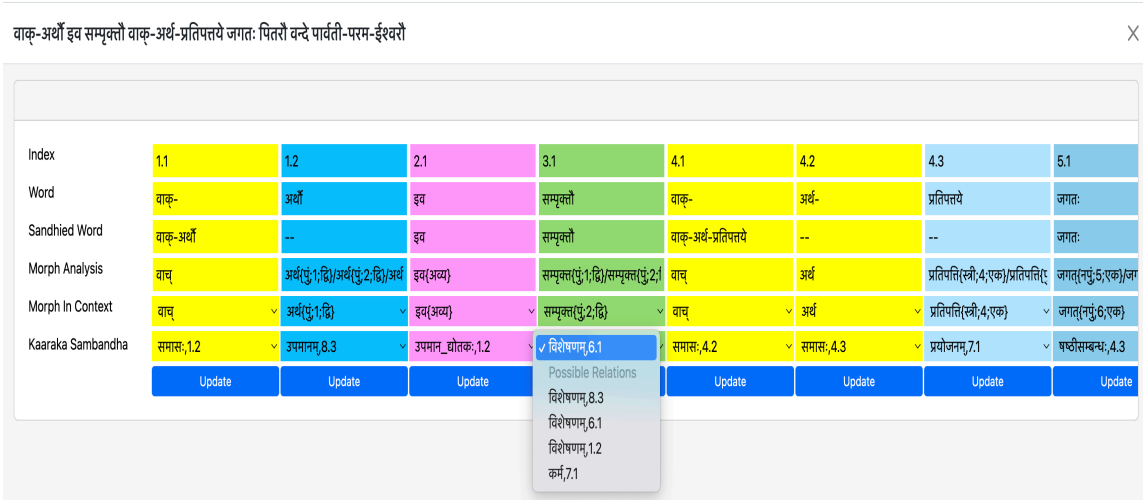


Figure 4: Grammatical analysis of the given sloka

- **Prose Order Generator:** Finally we also provide a tool to generate the prose word order from the dependency graph. The output of this tool is also editable, in case the user is not happy with the machine generated word order. In the current version, we have not yet plugged in the prose word order generator.
- **Translator:** We plan to plug-in the Sanskrit-Hindi/Telugu/Marathi translation systems, once they are ready for deployment. The tool also allows one to edit the translation, or provide a manual translation of the text into other languages.

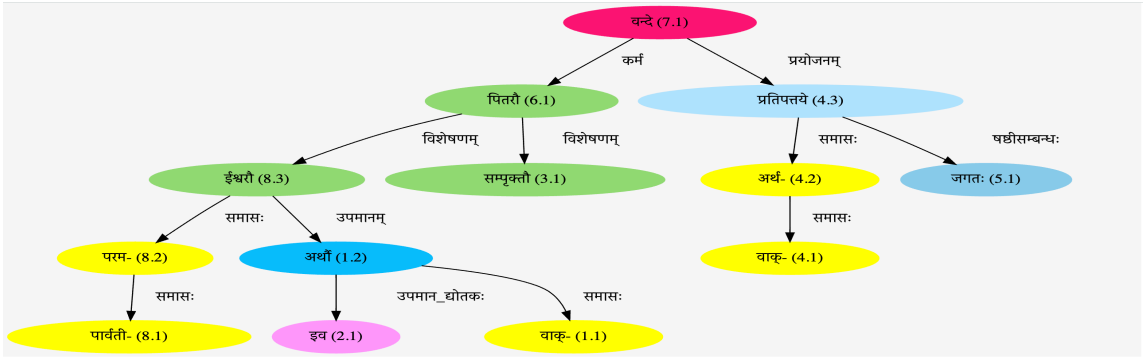


Figure 5: Dependency Tree

- **E-reader generator:** Once the analysed data is saved, the E-reader generator generates a page corresponding to the input dynamically showing analyses at all levels in a well structured manner with collapsible rows (See Fig 6). It is very important that the annotator herself visualises this data graphically to ensure that there are no mistakes (See Fig 7). E-reader builder provides this facility. Further, once such annotated data is certified by the editor, it is also made available to the general public, with a consent of the annotator and editor, on the E-reader platform, so that readers interested in reading such texts get benefited.

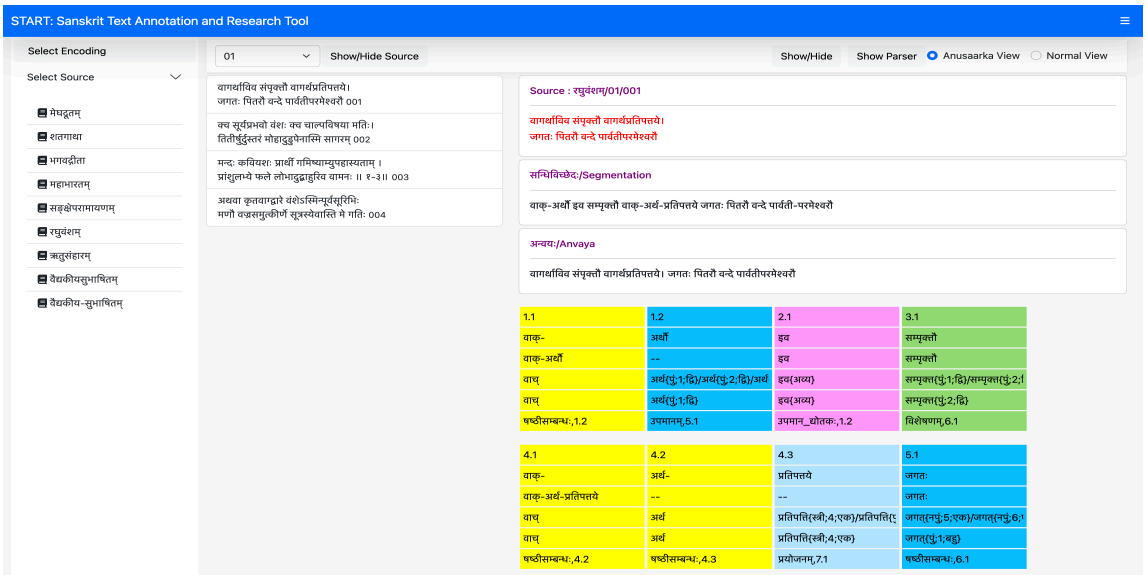


Figure 6: e-Reader

3.2 A Dashboard for learning and teaching Sanskrit

Under this mode, several computational tools such as sandhi joiner and splitter, morphological analyser and generator, sentential analyser and generator, a web interface for Amarakośa, and a concordance of important dhātuvṛttis are available (See Fig. 8). We describe these tools in brief highlighting the pedagogical aspect.

- **Sandhi** : Sandhi is an enhanced version of Saṁsādhanī's sandhi Joiner having 3 teaching modes such as Basic, Intermediate and Advanced. The basic mode can be used for teaching sandhi rules in the schools and also its very useful for beginners. The Intermediate and

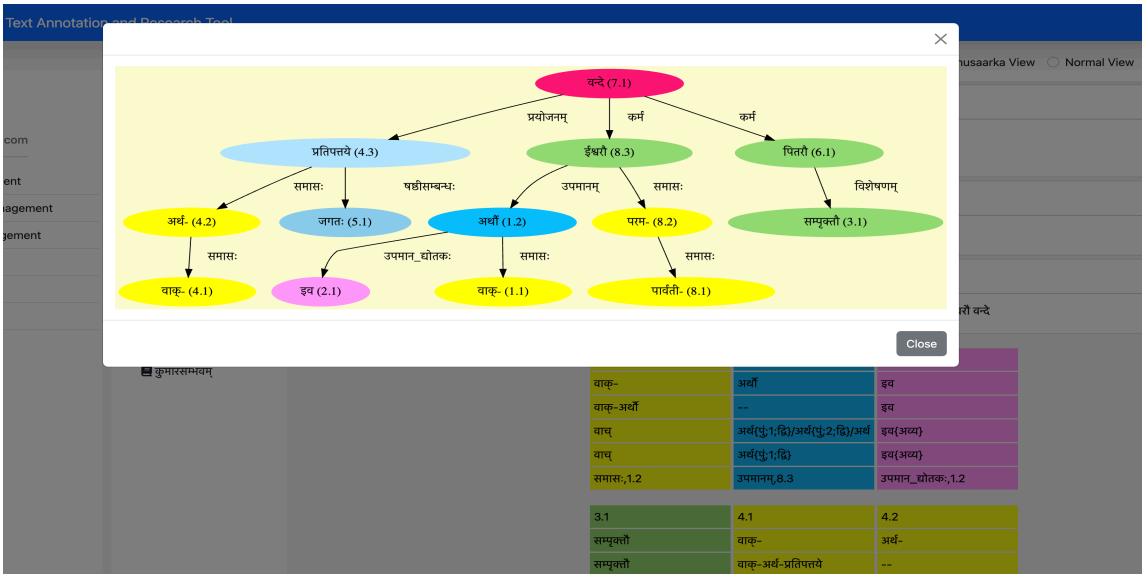


Figure 7: e-Reader with Parser

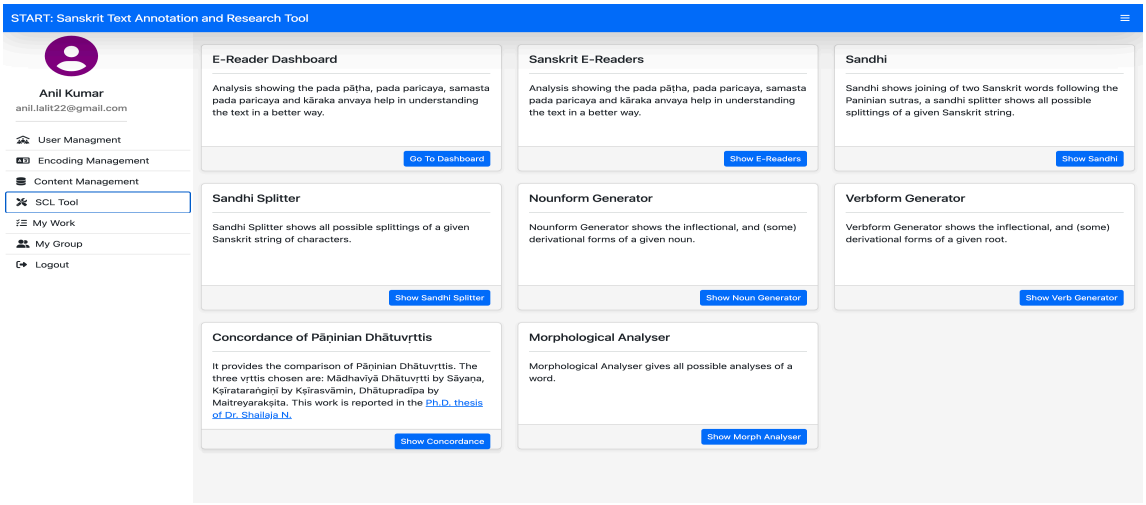


Figure 8: List of SCL tools

Advanced modes are for Sanskrit scholars or lovers who want to learn sandhi with Pāṇinian grammar. (See Fig 9)

- **Sandhi-Splitter** : Sandhi-splitter automatically splits or separates words that have undergone sandhi changes back into their original forms. It can be used for splitting single word or a full sentence or a paragraph. At the back-end, the Heritage Sanskrit Engine that prioritizes the solutions based on probability models is used.
- **Noun-form Generator** : It generates inflected noun forms of a given nominal stem (prātipadikam) in all the seven cases, and three numbers.
- **Verb-form Generator**: It generates the conjugational verb-forms of a verbal root (dhātu) with various tense, mood, voice, person, number, and one or more prefixes.
- **Morphological Analyser** : It provides all possible analyses of a given word. This is very much useful for a reader who has doubts about the morphological analysis of any word.

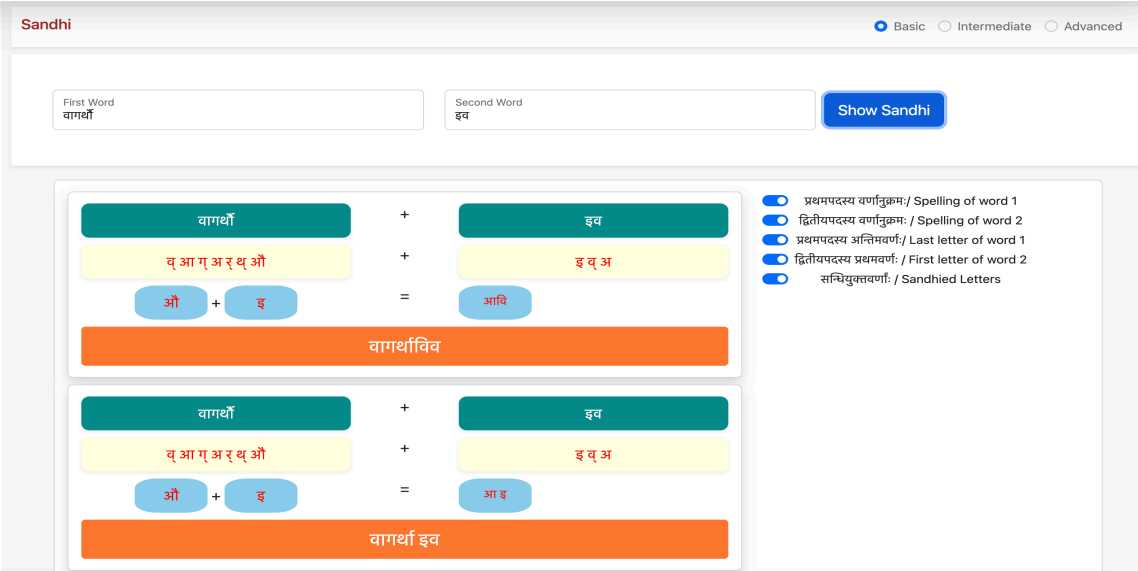


Figure 9: Sandhi

- **Concordance of Pāṇinian Dhātuvṛttis** : The current tool provides the comparison of Pāṇinian Dhātuvṛttis. The three vṛttis chosen are Mādhavīyā Dhātuvṛtti by Sāyaṇa, Kṣīrataraṅgiṇī by Kṣīrasvāmin and Dhātupradīpa by Maitreyarakṣita. This resource is very much useful for the Sanskrit scholars and students.
- **Sentential Parser**: This is part of the E-reader Dashboard.
- **Sentence Generator, Compound Word Generator and Amarakośa**: The interfaces for these are being developed. The original tools are available under Saṁsādhanī.

3.3 User Profile

Under user profile, each user can track his/her own work. The facility to form a group is useful for the team leader to monitor the progress of each member. A teacher can use this feature to design different assignments for different classes. If the answers are provided by the teachers in advance, the system can provide instant feedback to the students on their performance.

4 Technological Stack in Development

The combined utilization of Angular 13, Bootstrap 5, CGI, MongoDB 4, Python3 Flask, and Graphviz forms a powerful technological stack that empowers START with dynamic frontend interfaces, robust backend functionality, and insightful data visualization capabilities. This comprehensive toolkit lays the foundation for a feature-rich, responsive, and user-centric application.

- **Angular:**[17] Angular is a robust open-source framework for creating dynamic, single-page web applications, known for its component-based architecture and two-way data binding.
- **Bootstrap:**[18] Bootstrap is a widely-used open-source CSS framework that offers a collection of pre-designed templates and components, streamlining the process of building responsive and visually appealing websites.
- **CGI:** CGI (Common Gateway Interface) is a standard protocol for web servers to communicate with external programs, enabling dynamic content generation and interaction between web servers and applications.

[17] https://angular.io/
[18] https://getbootstrap.com/

- **MongoDB:**[19] MongoDB is a NoSQL database system that employs a document-oriented model for data storage, providing scalability and flexibility for managing large volumes of unstructured data.
- **Python Flask:**[20] Flask is a lightweight and flexible web framework for Python, simplifying web application development by offering essential tools and libraries, making it suitable for small to medium-sized projects.
- **Graphviz:**[21] Graphviz is an open-source graph visualization software that facilitates the creation of diagrams and graphs for visualizing complex relationships and structures in data or information systems.

5 Impact on Digital Humanities

START's innovation transcends the realm of Sanskrit studies, influencing the broader landscape of digital humanities. Its robust annotation framework, collaborative ethos, and analytical tools serve as a blueprint for similar initiatives in other languages and disciplines. As the digital humanities continue to evolve, START paves the way for a more interconnected and enriched scholarly community.

6 Future Horizons

The future of START holds tantalizing prospects. Integration with machine learning algorithms, natural language processing, and visualization tools could further amplify the platform's analytical capabilities. Additionally, efforts to expand the corpus of annotated Sanskrit texts could yield a more comprehensive understanding of the language's nuances, intricacies, and cultural significance.

7 Conclusion

The Sanskrit Teaching, Annotation and Research Tool (START) stands as a testimony to the harmonious convergence of ancient wisdom and modern technology. Its transformative annotation system and broader impact on the digital humanities signal a new era of scholarly exploration and inquiry. In embracing START, researchers embark on a journey that not only uncovers the depths of Sanskrit literature but also redefines the very essence of interdisciplinary and intertemporal scholarly engagement.

Acknowledgements

This research was made possible through the generous financial support of the IoE Directorate, University of Hyderabad for the project 'An Integrated Digital Platform for Language Learning, Teaching and Computational Linguistics' (2021-23).

References

The Text Encoding Initiative Consortium. 2021. Text encoding and interchange (TEI): `https://tei-c.org/guidelines/`.

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

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

[19] `https://www.mongodb.com/`

[20] `https://flask.palletsprojects.com/en/2.3.x/`

[21] `https://graphviz.org/`

Oliver Hellwig and Sebastian Nehrdich. 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, October-November. Association for Computational Linguistics.

Gérard Huet. 2004. Design of a lexical database for Sanskrit. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*, pages 8–14, Geneva, Switzerland, August 29th. COLING.

Amrith Krishna, Shiv Vidhyut, Dilpreet Chawla, Sruti Sambhavi, and Pawan Goyal. 2020. SHR++: An interface for morpho-syntactic annotation of Sanskrit corpora. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7069–7076, Marseille, France, May. European Language Resources Association.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. 2010. Sanskrit compound processor. In Girish Nath Jha, editor, *Sanskrit Computational Linguistics*, pages 57–69, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human `https://github.com/doccano/doccano` software available from `https://github.com/doccano/doccano`.

Jivnesh Sandhan, Anshul Agarwal, Laxmidhar Behera, Tushar Sandhan, and Pawan Goyal. 2023. Sanskritshala. In *A neural sanskrit nlp toolkit with web-based interface for pedagogical and annotation purposes.*

Peter M Scharf and Dhruv Chauhan. 2023. Rāmopākhyāna. In *A web-based reader and index. In Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference, pages 146–154, Canberra, Australia (Online mode), January. Association for Computational Linguistics*.

Krishnan Sriram, Amba Kulkarni, and Gérard Huet. 2023. Validation and normalization of DCS corpus and development of the Sanskrit heritage engine's segmenter. In *Proceedings of the Computational Sanskrit and Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2021. Sangrahaka: A tool for annotating and querying knowledge graphs. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*.

The Śabdabrahman exercise platform

Peter M. Scharf
President, The Sanskrit Library
Adjunct Professor, IIIT Hyderabad
scharf@sanskritlibrary.org

Harsha Pamidipalli
Software engineer, The Sanskrit Library
MS by research student, IIIT Hyderabad
harsha@sanskritlibrary.org

Abstract

The Śabdabrahman exercise platform (SBE), at `sabdabrahman.org`, is an on-line interactive Sanskrit instructional platform that offers immediate feedback and focused help to students at every step of analyzing and translating the Sanskrit sentences in a first-year Sanskrit textbook. Steps include transliteration from Devanagari to standard Romanization and vice versa, analysis of sandhi, identification of nominal and verbal inflection, syntax, compound analysis, and translation. Student submission at each step is evaluated, errors highlighted, and links supplied to relevant help. Source documents are prepared in XML in accordance with the Text-Encoding Initiative guidelines. The platform is coded in Flask and ReactJS and hosted at Amazon Web Services (AWS). The platform has been successfully used in first-year Sanskrit courses offered through The Sanskrit Library (`sanskritlibrary.org`).

1 Introduction

The decline of systematic instruction in Sanskrit and the increase of superficial conversational Sanskrit instructional materials dissappoint students who seek an introduction to the depth and complexity of the Sanskrit language and literature. The result is that students fail to be introduced to genuine historical texts so go from "I don't know" to "I don't care". Nevertheless, it is possible to use digital technology in an intelligent way to engage enthusiastic students in the beauty and depth of Sanskrit in a rigorous manner. The Śabdabrahman exercise platform (SBE), accessible at `sabdabrahman.org`, is an interactive platform that leads students through every step of understanding and translating into English the Sanskrit sentences in the first-year Sanskrit textbook by Scharf (2022a) while providing detailed feedback and focused help. The platform is currently available to students enrolled in first-year Sankrit courses at the Sanskrit Library `sanskritlibrary.org/courses.html`. Soon to be added to the platform are the verses and prose paraphrases of them by Scharf (2022b) for second-year Sanskrit students. Śabdabrahman is a dynamic platform that combines a rich repository of resources with interactive tools. In an age where digital technologies have reshaped educational paradigms, this website offers an unparalleled opportunity to delve into the rich tapestry of Sanskrit literature and linguistics. We shall navigate through the various facets of Śabdabrahman, elucidating its architecture, key features, and the user experience it offers.

2 Motivation

In a first-year Sanskrit course outside of India, as in the first-year course in any ancient language, or in any language course that focuses on developing the ability to read texts, students are confronted with several aspects of the language at once: script, phonetics, morphology, syntax, and semantics. In daily homework assignments, Sanskrit students transcribe from Devanagari to Roman script with diacritics, analyze inter-word prosodic changes (*sandhi*), identify the inflection of each word, that is, the declension of nominal forms and conjugation of finite verbs, analyze the syntactic structure of each sentence and translate. In a homework set which typically

consists of thirty or more sentences, the student might repeatedly encounter the same difficulty or make the same mistakes. However, the student has no opportunity to clarify his understanding or correct his mistakes until he submits the homework assignment and the instructor corrects it and hands it back days or perhaps a week later. Lingering doubt and repeated errors lead to frustration on the part of the student. The instructor too typically spends hours correcting the students' transcription from Devanagari to Roman script with diacritics, analysis of inter-word prosodic changes (sandhi), inflectional identification, syntax, and translation. The correction of the first several steps is extremely time-consuming and tedious. Frequently students proceed to a new exercise set before they receive corrections on their first set; hence they often repeat the same mistakes, not only within an exercise set but also in subsequent sets, leading to further frustration. The instructor, to his exasperation, therefore has to repeat the same comments numerous times. Moreover, because the student receives comments on the first set several days after completing it, due to the delay he might no longer recall the context of instructor comments so might not attend to the correction with due attention. In short, the information lag makes learning inefficient.

Ideally it is desirable for the student to receive immediate focused feedback on each step of each sentence in each exercise set, that is, to receive immediate confirmation of correct work, to receive immediate focused pin-pointing of mistakes, and to be directed to and have immediate access to relevant information to explore issues about which he has doubt or confusion. We have developed an on-line interactive exercise platform that offers these features.

Most currently available language learning software is oriented towards teaching business and travel themes for aural/oral communicative use of modern languages (Murray and Barnes 1998: 253). Popular apps such as DuoLingo, Rosetta Stone, Babbel, Pimsleur, etc. are all geared toward these goals. The development of comprehensive language-learning software involves assembling numerous experts and is an expensive undertaking from conception and design to implementation and evaluation (Turel and McKenna 2014: 1200). The development of comprehensive language-learning software for classical culture-bear languages is slim. The system we have developed is unprecedented.

3 Overview of the Śabdabrahman exercise platform (SBE)

The interactive Śabdabrahman exercise platform (SBE) provides the student with immediate focused help at each stage of working through a sentence. This help enables the student to correct his own mistakes. The platform does not provide the correct answer but rather pinpoints the error and provides links to assistance. Each stage of working through a sentence is dealt with in a separate pane. Each pane has a sidebar provided with links to appropriate help and also evaluates the student's submitted work.

First the platform asks the student to transcribe the Devanagari sentence (Figure 1). It then compares the student's transcription of Devanagari with the correct transcription and highlights differences (Figure 2). The highlit error is linked to an appropriate document: a page showing simple signs or dependent vowel signs and their Romanization, or a page showing conjunct consonant signs.

Once the transcription is correct, a confirmation message is given and the student is presented with the second pane where he is asked to analyze sandhi. SBE implements sandhi on the student's sandhi-analysis, compares the result with the original question and highlights the errors in red (Figure 3). Where doing sandhi to the student's sandhi-analysis does reproduce the correct original, but the analysis is erroneous or insufficient, SBE compares the student's sandhi-analysis with the correct analysis, highlights differences in blue and displays brief instructions upon mouse-over (Figure 4). Clicking on the highlit differences links to the appropriate sandhi table.

Once the sandhi analysis is correct, a confirmation message is given and the student is presented with the third pane in which he is asked to select the words' lemma, i.e. the root of

Figure 1: SBE transliteration question

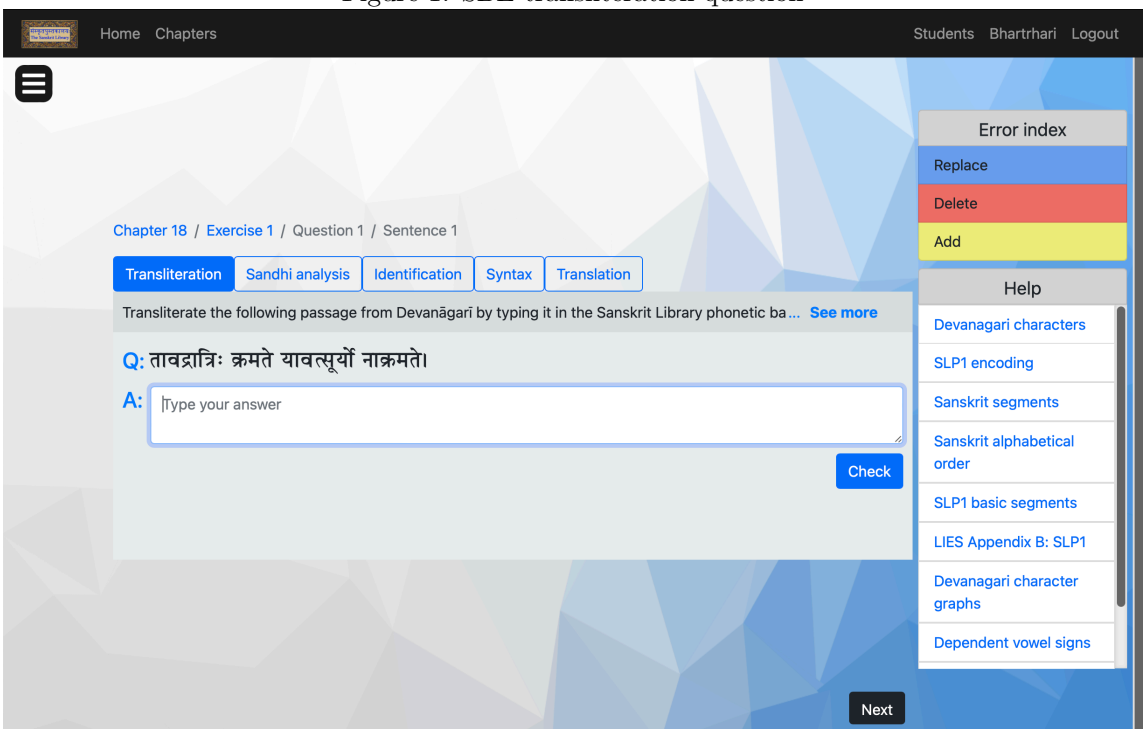


Figure 2: SBE transliteration error highlighting

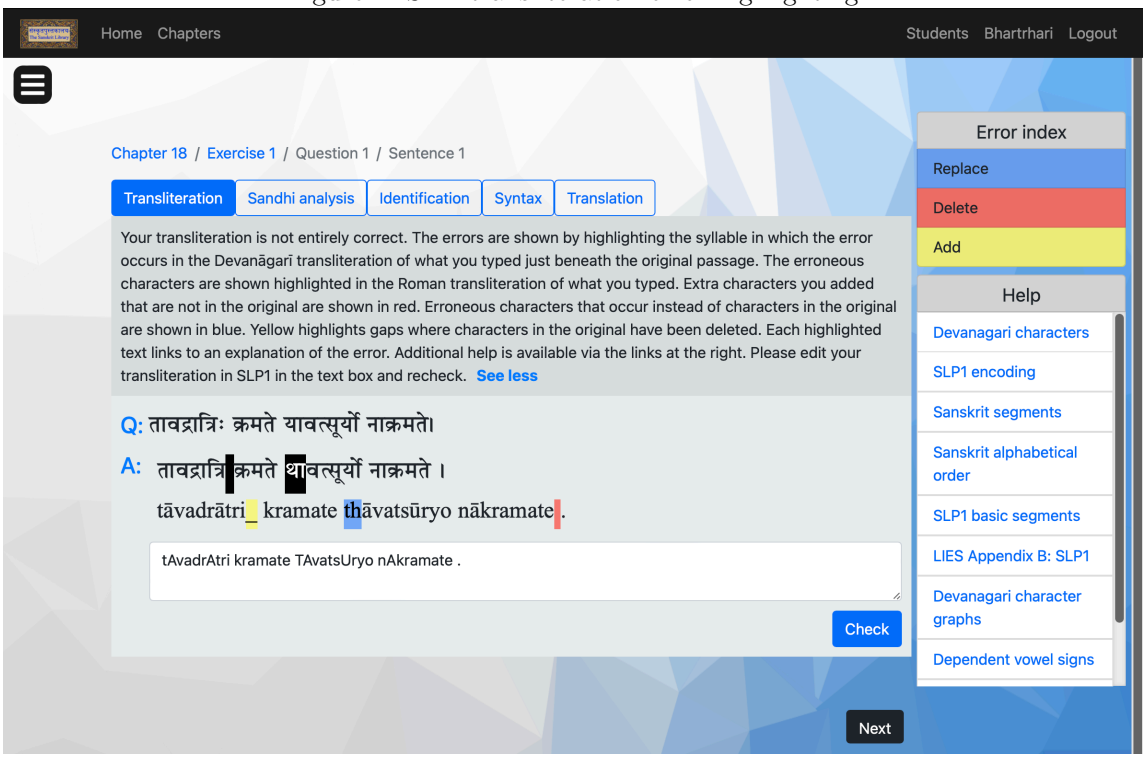


Figure 3: SBE sandhi analysis pane with highlit errors

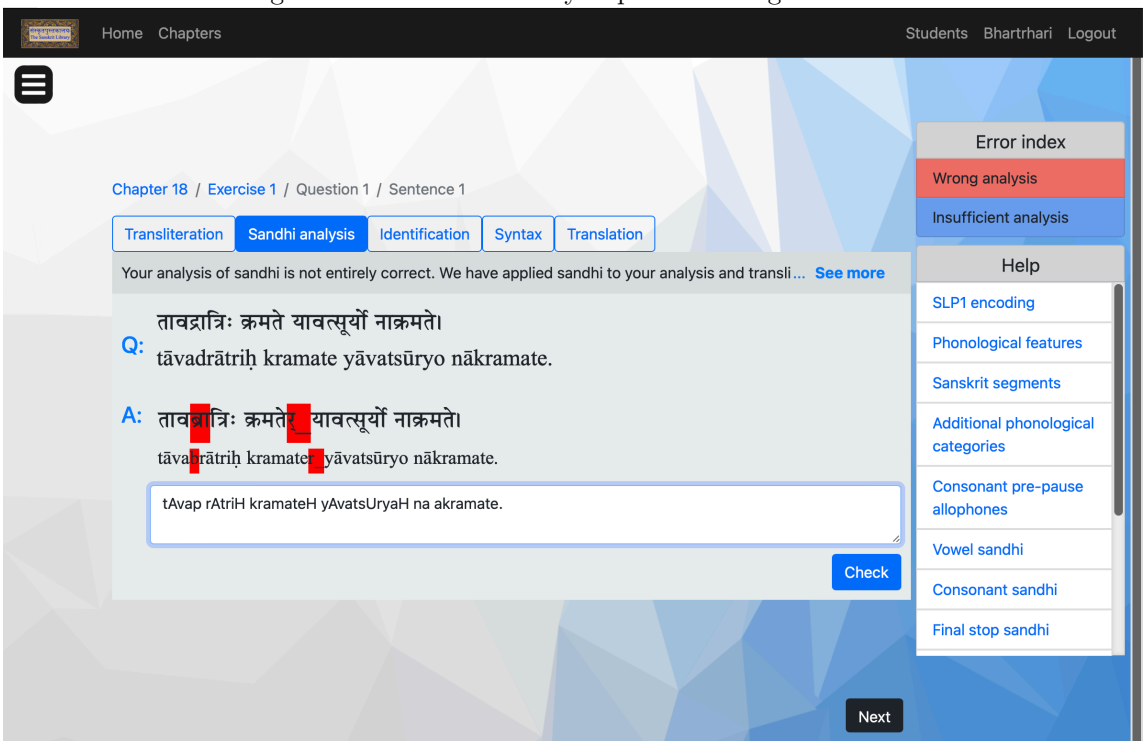
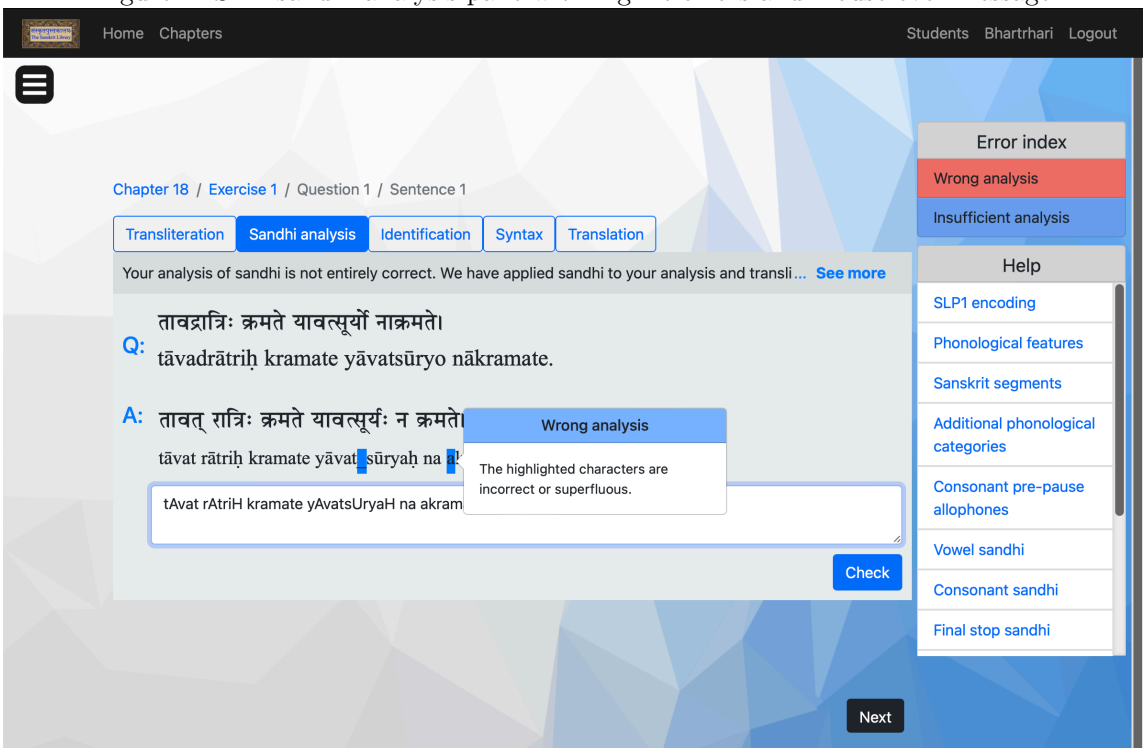


Figure 4: SBE sandhi analysis pane with highlit errors and mouse-over message



each finite verb or the stem of each nominal, from a glossary, choose its correct lexical category, and enter its inflectional identifier (Figure 5). A dialogue box that shows the possible options for each parameter assists with inflectional identification if desired (Figure 6). Experienced students can simply type the inflectional identifier in the text box. The morphological identification of each word can be checked individually or all at once. Incomplete items and mistakes are flagged (Figure 7). If the student has selected a root or stem but made an error in its inflectional identification, SBE provides a link to the inflectional paradigm appropriate to the selected stem.

When the student has correctly identified all of the inflectional morphology, SBE provides a confirmation message and the student can proceed to the fourth pane in which he is asked to analyze the syntax of the sentence, and the fifth pane in which he is asked to translate the sentence. He is free to move between the syntax and translation panes. Syntactic relations are divided into two classes: primary and secondary. Primary relations include identification of the main verb of a clause and relations denoted by nominal declension such as kāraka relations, qualification, and possession. Secondary relations include predication, and relations indicated by particles such as conjunction, alternation, and contrast. We save detailed discussion of these syntactic relations for another occasion. For each word, the student selects the primary or secondary relation, and the target or targets of the relation. A target is another word in the sentence to which the word is subordinate in that relation. Relations are selected by flex search from a list of relations that are possible for the given item, and targets are selected by flex search from a list of words in the sentence. Errors are flagged when the student checks. Once primary relations are correct, SBE makes available a graph showing primary relations, or, if all relations are correct, a graph showing both primary and secondary relations (Figure 8). Primary relations are shown in gold with blue arrows to their targets; secondary relations in silver with green arrows to their targets.

The translation pane presents the student with all of the information he has previously completed correctly to assist him in composing his translation as well as a brief translation of each lemma as given in the glossary (Figure 9). When the student enters a translation in the translation pane text box, SBE checks for a match against the possible correct translations provided. If it does not match any, SBE verifies whether all necessary terms have been included or not, calculates the closeness to the correct translation, and provides an appropriate message. If the student simply enters the word translations provided, SBE recognizes the translation as incomplete (Figure 10). SBE also includes nominal, verbal, and participle identification drills, and compound analysis not shown here.

All answers including translations are saved and made available to the instructor to go over with the students in their subsequent meeting. SBE itself provides entirely complete feedback to the student on all clearly categorical steps, but the translation may involve subtle nuances beyond the scope of currently available technology to evaluate. The instructor may view just the translation, or additional steps so that they may be shown to students for explanatory purposes in class meetings (Figure 11). SBE tracks the completed exercises of each student for both the student's benefit, and the instructors evaluation. Percentages are displayed of completed questions within each exercises, lesson, and text.

4 Source

The questions and correct answers are prepared in a set of coordinated XML files structured in accordance with the Text-Encoding Initiative guidelines. Scharf (2018) discusses and illustrates the general features of the TEI markup of verses and their constituent verse quarters and words. Ajotikar and Scharf (2023) describe the use of TEI to mark up commentaries. The Search and Retrieval of Indic Texts (SARIT) Website includes detailed guides to how to structure a Sanskrit text in accordance with the TEI Guidelines under the About SARIT menu `https://sarit.indology.info`. Hence we pass over detailed description of the structure within each

Figure 5: SBE identification pane with lemma flex search

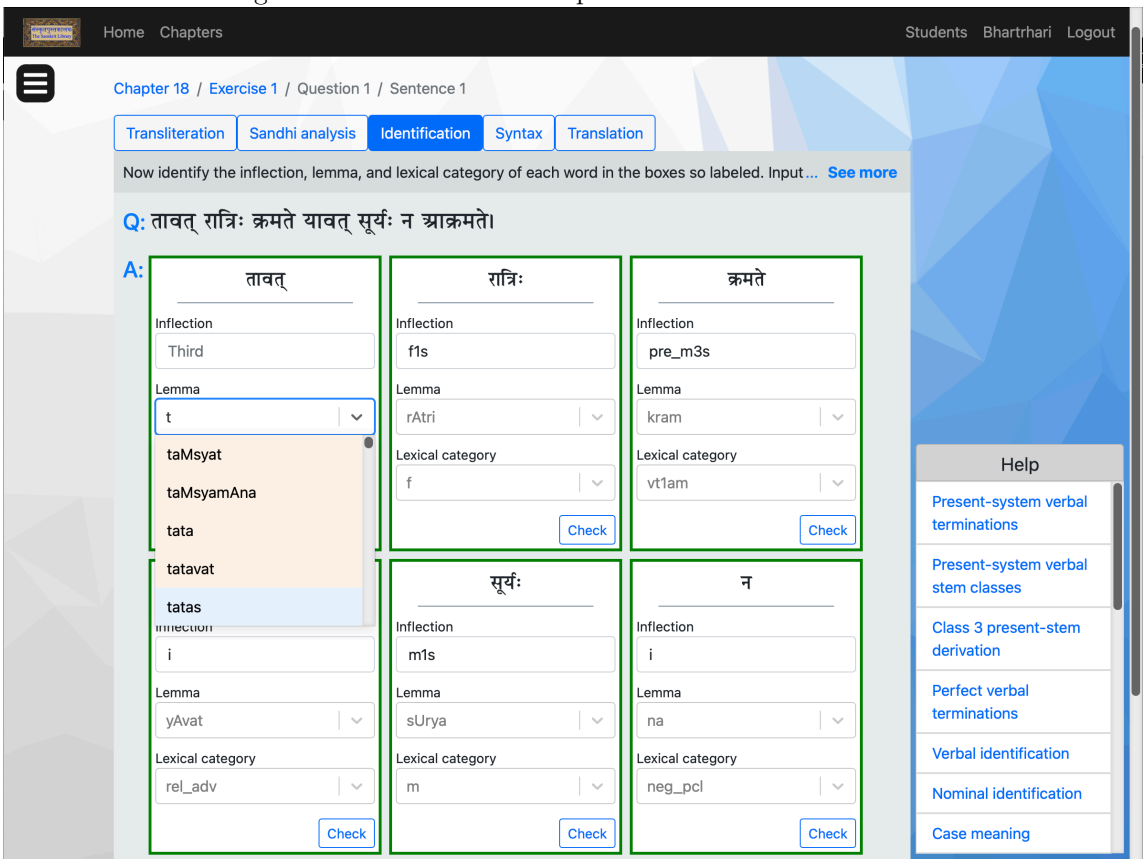


Figure 6: SBE identification pane with inflection identification dialogue box

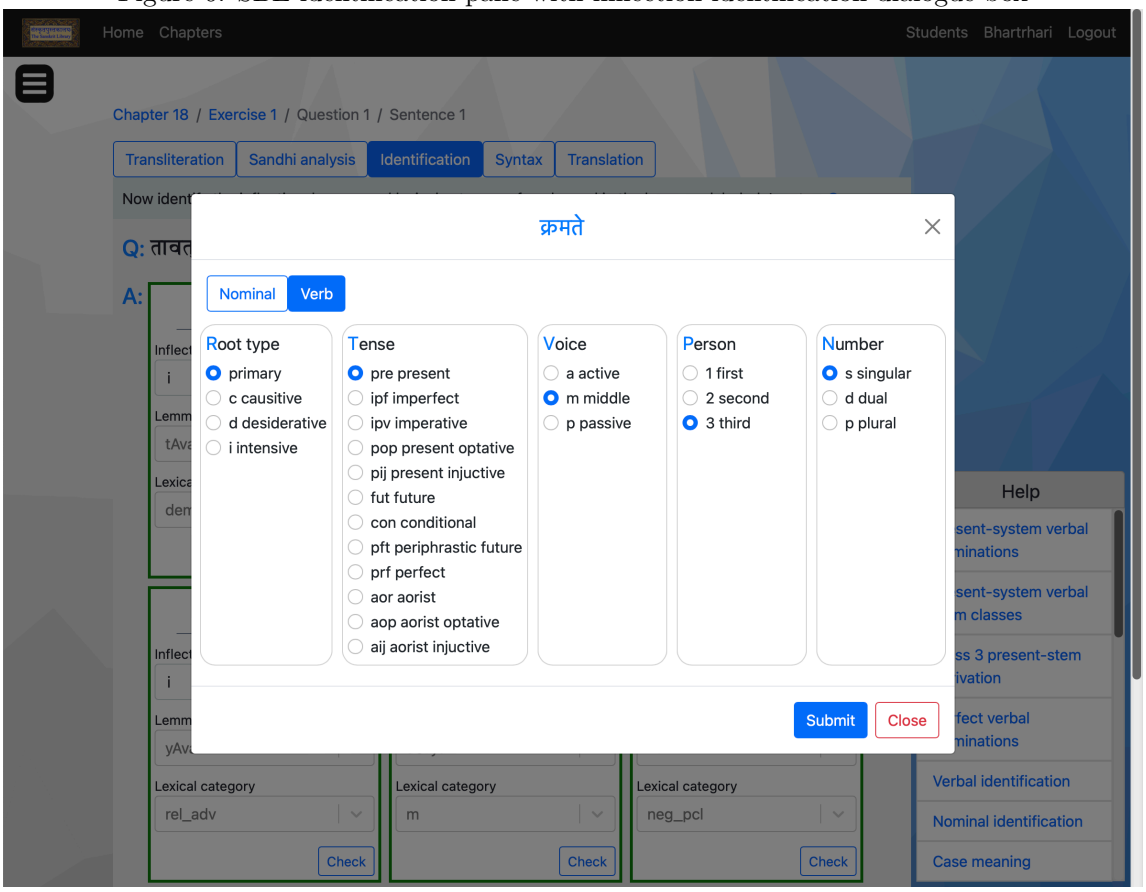


Figure 7: SBE identification pane error flagging

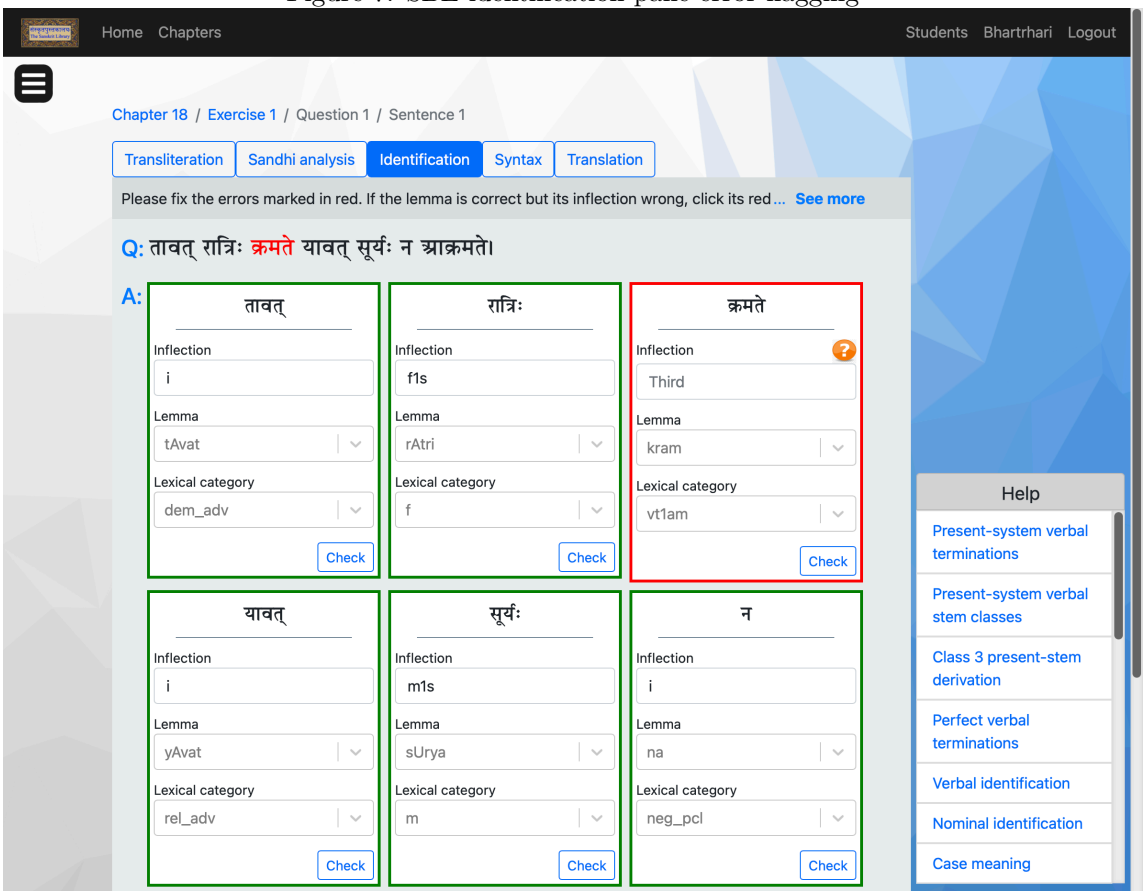


Figure 8: SBE syntax tree with primary and secondary relations

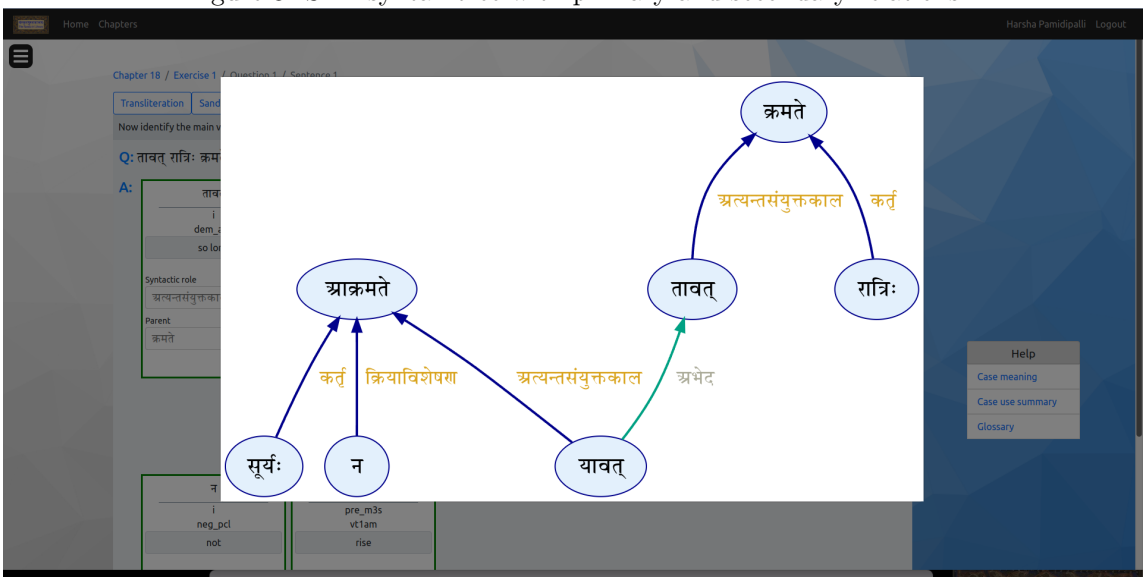


Figure 9: SBE glossary

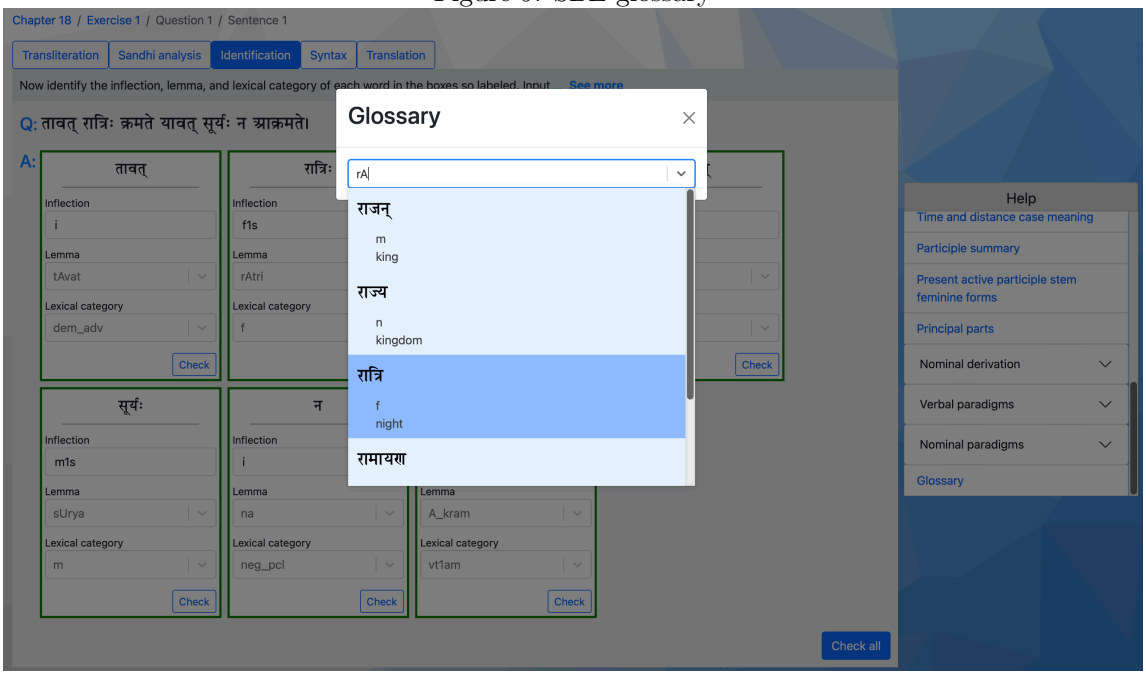


Figure 10: SBE translation pane recognizing an incomplete lemma-only translation

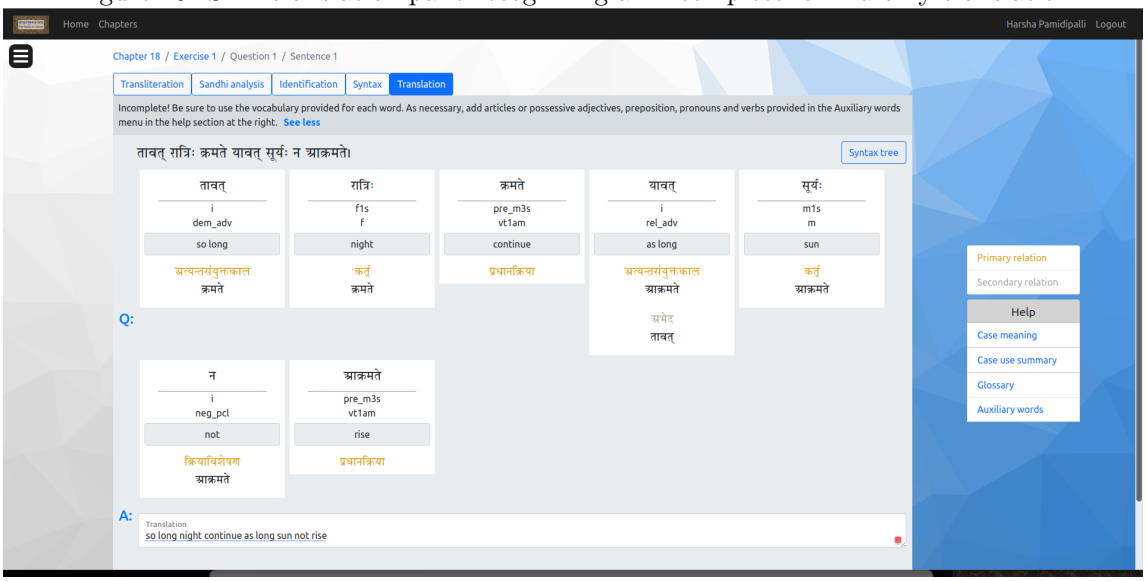
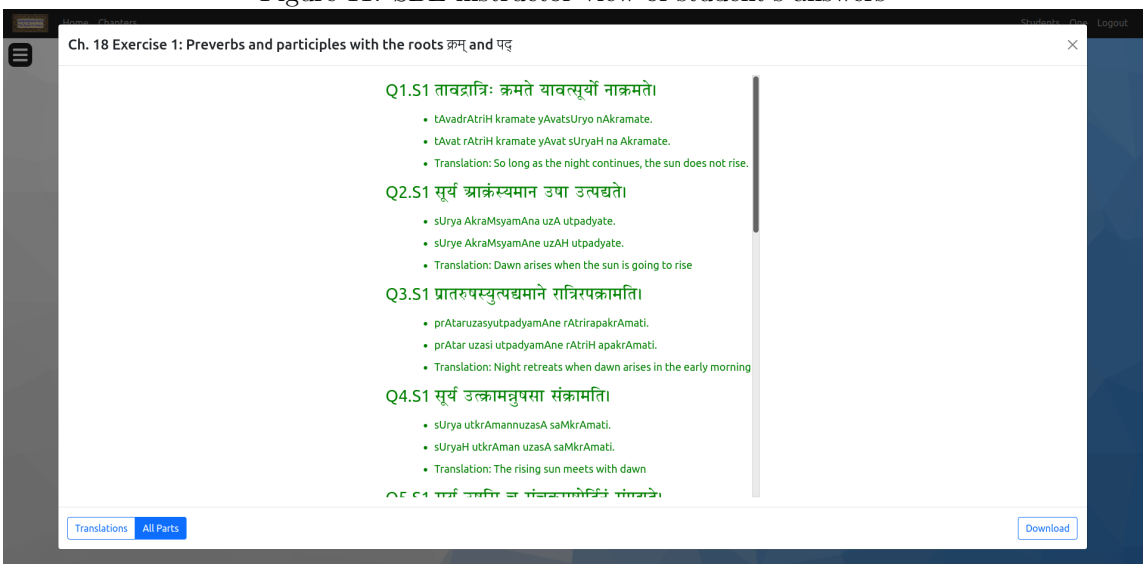


Figure 11: SBE instructor view of student's answers



file in the set of files that constitute the source of the Śabdabrahman platform but instead just briefly describe the files and their relation. Necessary files include a morpheme file, a morpheme translation file, and an English translation file. Files are coordinated using the xml:id of each sentence, word, and morpheme as key. In the morpheme file each sentence is analyzed into words and their constituent morphemes, each word is provided with an inflectional identifier and syntactic relation attributes, and each morpheme is given a lexical identifier. A sandhi-analyzed sentence file, and question file can be produced programmatically from the morpheme file by assembling words in sequence and applying sandhi between them. In fact the files were initially created in the opposite order, that is, by manually analyzing sandhi in a question file to produce a sandhi-analyzed sentence file, then programmatically creating a word file with blank attributes for morphological and syntactic identification and morphemic analysis, then manually filling in those blanks.

5 Website structure

SBE is structured like Scharf (2022a), the first-year Sanskrit text book *Śabdabrahman: a linguistic introduction to Sanskrit*, on which it was initially based. This structure makes it intuitive for users to navigate, access content, and engage with the platform. This section provides an in-depth exploration of the website's structural framework, elucidating its hierarchy, navigation, and organizational principles.

5.1 Hierarchy and navigation

SBE adopts a well-structured hierarchy that facilitates intuitive navigation for users. The website's hierarchy includes the following features:

Navigation Bar: Provides quick access to key sections such as homes, chapters, and user profiles.

Menus: A simple main menu and sub-menus within each main section enable users to drill down to specific content and help.

Internal links: Hyperlinks and breadcrumbs connect related content within and across pages, enhancing user navigation.

5.2 Sections and pages

The website is divided into the following distinct sections, each dedicated to a specific aspect of Sanskrit learning and exploration:

Home: A simple landing page displays a welcome message and a link to the Chapters section.
Chapters: A top-level menu displays all the chapters. Each chapter shows the exercises included in it, and each exercise shows the questions within it.
Resources: A menu provides access to a repository of textual and multimedia resources, including classical Sanskrit texts, translations, pronunciation guides, and dictionaries at The Sanskrit Library (`sanskritlibrary.org`).
User profiles: Users can create and manage their profiles and track their progress. Teachers have access to student exercises and their progress.

Each section contains multiple pages, ensuring comprehensive coverage of topics and resources related to Sanskrit studies.

5.3 Sitemap and diagrams

To provide a visual representation of the website's structure, an illustrative sitemap and diagrams are included in this section. These diagrams offer a bird's-eye view of how various pages and sections are interconnected.

5.4 User-centric design

The website's structure is designed with the user experience in mind, ensuring that learners, regardless of their familiarity with Sanskrit, can easily find, access, and engage with content. User feedback and usability studies have played a crucial role in refining the website's structure, leading to an intuitive and learner-friendly interface.

5.5 Key features and functionalities

The following are the core features and functionalities that define the user experience in SBE:

Intelligent help: SBE provides intelligent and extensive feedback when a user makes a mistake. Software not only evaluates and classifies errors but also highlights them and links to apt help. This enables the user to strengthen his understanding of the relevant concept and submit the correct answer.
Interactive exercises: To enhance the learning experience, SBE provides interactive tools and resources.
Glossary: A comprehensive, searchable dictionary with part-of-speech lexical identification, and translation of every word in the text.
User-friendly input tools: Dialogue boxes and flex lists ease the users selection.

These resources empower learners to increase their understanding and hone their language skills in a hands-on manner.

6 User experience

In this section, we examine the user-centric design of SBE, focusing on how the website's layout, design, and interactive elements contribute to an effective and enjoyable Sanskrit learning journey. A user-centric approach is integral to the website's success in catering to a diverse audience.

Intuitive navigation: SBE prioritizes user-friendly navigation, ensuring that learners of all levels can easily find their way around the website. The clear and organized menu structure guides users to the resources, courses, and community features they seek. Intuitive breadcrumbs, internal links, and a user-friendly search function further enhance navigation.

Responsive design: SBE employs a responsive-design approach, adapting seamlessly to various devices, including desktops, tablets, and smartphones. This flexibility ensures that users can engage with Sanskrit learning content on their preferred devices, promoting accessibility.

Clear course progression: Courses in SBE are structured with a clear progression, allowing users to systematically advance their Sanskrit language skills. Learners can easily track their progress through modules, lessons, and exercises, motivating them to continue their studies.

Interactive Learning: Interactive elements, such as quizzes, exercises, and pronunciation guides, actively engage users in the learning process. These features reinforce comprehension and retention while maintaining an engaging and dynamic learning environment.

User profiles and progress tracking: User profiles in SBE empower learners to monitor their progress, view achievements, and set personalized learning goals. Progress tracking not only motivates users but also allows them to pick up where they left off in their Sanskrit studies.

The user experience in SBE is thoughtfully designed to cater to a wide-ranging audience, from beginners to advanced scholars. By offering intuitive navigation, and interactive learning tools, the website promotes a positive and engaging environment for Sanskrit enthusiasts. In the following sections, we will explore the technical aspects, including the technology stack and any notable achievements in enhancing the user experience.

7 Technical details

This section delves into the technical underpinnings of SBE, shedding light on the technologies, infrastructure, and considerations that enable the website to function seamlessly. Understanding these technical aspects provides valuable insights into the website's robustness and scalability.

7.1 Technology stack

SBE leverages a well-defined technology stack to deliver its services. Key components may include:

ReactJS: The user interface is built using the Javascript framework ReactJS. A responsive and interactive frontend has been developed making use of its own bootstrap and many supported libraries.

Flask: Python programming language's Flask framework has been used to develop the backend. A REST API is created to enable interaction between the client (frontend) and the server (backend).

Database system: An SQLite database is used to store all the data pertaining to the questions, exercises, and chapters. User information and their answers are also stored here. Flask's SQL Alchemy is used to interact with the database.

Web server: Amazon Web Services (AWS) is used to host both the frontend and backend.

7.2 Hosting and infrastructure

SBE operates on a reliable hosting infrastructure, ensuring high availability and performance. Key aspects to consider include:

Hosting provider: Amazon Web Services (AWS) is the hosting provider. The website is deployed in the cloud using Amazon's Elastic Compute Cloud (EC2) instance. EC2 provides scalable computing capacity as a virtual server. At present SBE is hosted on a t4g.small EC2 instance.

Server specifications: The t4g.small instance provides 2 virtual CPUs and 2 GB of RAM. This instance is powered by AWS Graviton2 processors, which are based on 64-bit Arm Neoverse cores. Amazon's Elastic Block Store (EBS) is linked with the instance to get a storage capacity of 25 GB.

Scalability: T4g.small instances provide baseline CPU performance that can temporarily burst to handle increased workloads. One can customize scaling behavior to match the application's needs or enable autoscaling, which will duplicate the instance to handle greater load.

Security measures: Security is paramount for SBE. The security measures in place to protect user data include:

SSL encryption: A library named certbot is used to generate an SSL certificate. An SSL certificate is crucial to secure data during transmission over the internet. It provides data privacy, integrity, and authentication, which are essential for online security, trust, and regulatory compliance.

User authentication: Google's Firebase is used to authenticate users with email and password. A simple graphic User Interface (GUI) helps the professor or administrator create multiple user accounts at once. Once a user's account is created, the user is notified and given a temporary password. Users can log in with this and change their password.

7.3 Maintenance and updates

Through regular testing and feedback from the users, we make many revisions and upgrades to the website. In order to update the website securely, we minimized the room for human error by automating the whole process. We have a bash script in place that will take backups and sync all the files with the server. Apart from this, we backup our database every day and maintain the last three days' backups.

The whole project is maintained on Github for version control. All the updates and changes are logged as and when an update is made to the website. Appropriate comments and documentation help maintain a clear and transparent record of all the changes to the website.

8 Use

The SBE system has been used already by eighty-eight students in several courses with glowing evaluation. The principal advantages are that SBE provides immediate focused feedback, makes help easily accessible, and interaction with it is engaging, makes learning more fun, and allows more effective use of class time. One student summed up his experience in the Sanskrit Library's first-year Sanskrit class using the Śabdabrahman exercise platform as follows:

> How refreshing to have instant and definitive responses to questions! As you work through assigned transliteration, sandhi, and parsing you get feedback virtually immediately. And the volume of exercises assigned ensures that newly-learned material is reinforced.
>
> I have found SBE's quick feedback invaluable for learning the basics of the language, allowing for solid progression in a much more time-efficient way than in a traditional classroom. With SBE, time in class with the professor (via Zoom) can be more productively focused on exploring deeper principles, instead of learning basics. The dynamic of SBE-plus-classroom meetings enhances overall progress, making the learning of Sanskrit more fulfilling and satisfying than pure self-study alone. I felt like I had been walking for a long time on the side of the road, and then someone stopped their car and gave me a ride.

9 References

Ajotikar, Tanuja P. and Peter M. Scharf (2023). "Development of a TEI standard for digital Sanskrit texts containing commentaries: A pilot study of bhaṭṭi's Rāvaṇavadha with Mallinātha's commentary on the first canto." In: *Proceedings of the Computational Sanskrit & Digital Humanities: Selected papers presented at the 18th World Sanskrit Conference*. Ed. by Amba Kulkarni and Oliver Hellwig. Canberra, Australia (Online mode): Association for

Computational Linguistics, pp. 128–45. URL: https://aclanthology.org/2023.wsc-csdh.9.

Murray, Liam and Ann Barnes (1998). "Beyond the 'wow' factor—evaluating multimedia language learning software from a pedagogical viewpoint." In: *System* 26.2, 249–59.

Scharf, Peter M. (July 2018). "TEITagger: Raising the standard for digital texts to facilitate interchange with linguistic software." In: *Proceedings of Computational Sanskrit & the Digital Humanities: selected papers presented at the World Sanskrit Conference*. University of British Columbia, Vancouver.

–. (2022a). *Śabdabrahman: a linguistic introduction to Sanskrit*. The Sanskrit Library.

–. (2022b). *Saṅkṣiptamahābhāratam: the Mahābhārata in a nutshell: a one-chapter narration in forty-three verses presented as an independent-study reader in Sanskrit*. The Sanskrit Library.

Turel, Vehbi and Peter McKenna (2014). "Design of language learning software." In: *Software Design and Development: Concepts, Methodologies, Tools, and Applications*. IGI Global, pp. 1200–21.