Sanskrit Sentence Generator: A Prototype

by Madhusoodana Pai J

Submission date: 25-Jul-2019 09:44AM (UTC+0530)
Submission ID: 1154815137
File name: ssg.pdf (785.69K)
Word count: 15593
Character count: 82293

Sanskrit Sentence Generator: A Prototype

A dissertation submitted to the University of Hyderabad

for the award of the degree of

Doctor of Philosophy

in

Sanskrit Studies

by

Madhusoodana Pai J

15HSPH01

Department of Sanskrit Studies

School of Humanities

University of Hyderabad

Hyderabad

2019

Sanskrit Sentence Generator: A Prototype

A dissertation submitted to the University of Hyderabad

for the award of the degree of

Doctor of Philosophy

in

Sanskrit Studies

by

Madhusoodana Pai J

15HSPH01

under the guidance of

Prof. Amba Kulkarni

Professor, Department of Sanskrit Studies

Department of Sanskrit Studies

School of Humanities

University of Hyderabad

Hyderabad

2019

Declaration

I, **Madhusoodana Pai. J**, hereby declare that the work embodied in this dissertation entitled "**Sanskrit Sentence Generator: A Prototype**" is carried out by me under the supervision of **Prof. Amba Kulkarni**, Professor, Department of Sanskrit Studies, University of Hyderabad, Hyderabad and has not been submitted for any degree in part or in full to this university or any other university. I hereby agree that my thesis can be deposited in Shodhganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Madhusoodana Pai J

15HSPH01

Date:

Place: Hyderabad

Signature of the Supervisor

Certificate

This is to certify that the thesis entitled ***Sanskrit Sentence Generator: A Prototype*** Submitted by **Madhusoodana Pai. J** bearing registration number **15HSPH01** in partial fulfilment of the requirements for the award of **Doctor of Philosophy** in the **School of Humanities** is a bonafide work carried out by him under my supervision and guidance.

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

Further, the student has the following publication(s) before submission of the thesis for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of research:

1. **Sanskrit Sentence Generator**, 6th International Sanskrit Computational Linguistics Symposium, Indian Institute of Technology Kharagpur, West Bengal, India-721302, (forthcoming Bhandarkar Oriental Research Institute's symposium proceedings)

2. **Semantics of Morpho-syntctic Case-markers in Indian Languages: Sanskrit a Case Study**, Karnatakasamskrita-adhyayanam, Half yearly journal, Karnataka Samskrit University, Pampa Mahakavi Road, Chamarajapet, Bangalore – 560018, Karnataka, India. ISSN #2249-1104. (Same as the fourth chapter named **Handling Upapadas**)

and has made presentation in the following conferences:

1. **Sanskrit Sentence Generator**, International Conference on Churning of Indology by Bharatiya Vidvat Parishad & Tattvasamshodhana Samsat, Udupi, 4 to 6, January, 2019.

2. **Semantics of Morpho-syntctic Case-markers in Indian Languages: Sanskrit a Case Study**, 47th AICDL & International Symposium by School of Humanities and Languages, Central University of Karnataka Gulbarga, 20 to 22, June 2019.

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

No.	Course-Code	Course Title	Credits	Pass/Fail
1.	SK801	Research Methodology	4.00	Pass
2.	SK802	Padārthavijñānam	4.00	Pass
3.	SK812	Topics in Natural Language Processing	4.00	Pass
4.	SK830	Śikṣā and Prātiśākhya (Subject related reading)	4.00	Pass

Prof. Amba Kulkarni
Supervisor and Head of the Department
Department of Sanskrit Studies
School of Humanities
University of Hyderabad

Dean
School of Humanities
University of Hyderabad

Acknowledgments

Here by I express my salutations to my teachers late *Shri. Purushottama Pai*, late *Shri. Ramananda Bhatt*, *Shri. P. A. Sahasranaman*, *Smt. Ratna Bai*, *Smt. Chandri Bai*, and *Smt. Renuka*, whose noble living inspires me.

This dissertaion would not have been possible without the guidance and the continuous help of several individuals extended their valuable assistance in the preparation and completion of this study.

First and formost, I am grateful to my supervisor *Prof. Amba Kulkarni* for her guidance and support to complete this work, and giving an oppurtunity to work under her.

I express my sincere gratitude to *Prof. J. S. R. Prasad*, *Prof. Kavi Narayana Murthy*, *Prof. Aloka Parasher Sen* and *Dr. Parameswari Krishnamurthy* for their all sort of academic support and encouragement.

I thank *Dr. M. P. Unnikrishnan*, *Dr. V. Nithyanantha Bhatt* from Sukṛtīndra Oriental Research Institute, Kochi; *Prof. Peri Bhaskara Rao* from IIIT Hyderabad and *Dr. Kirtikanth Sharma* from IGNCA, New Delhi for their help.

My special gratitude to *Dr. Anil Kumar* who helped me with all the technical aspects of my research work. I also thank *Sanjeev Panchal* and *Sanal Vikram* for their continuos help.

My sincere thanks to *Shri. C.G. Krishnamurthi*, *Dr. Shivani*, *Dr. Anupama Ryali*, *Dr. Patel Preeti Khimji*, *Dr. Shailaja Nakkawar*, *Dr. Arjuna S. R*, *Dr. Rupa Vamana Mallya*, *Dr. Pavan Kumar Satuluri*, *Brijesh Mishra*, *Mridula Aswin*, *Abirlal Gangopadhyay*, *Anil Kumar Sanmukh*, *Malay Maity*, *Shakti Kumar Mishra*, *Amrutha Arvind Malvade*, *Saee Vaze*, *Adv. Jagadeesh Lakshman*, *Srinivas Indla*, *Radhakrishnamurthi*, *Dr. Chander Amgoth*, *Dr. Bharadvaj Vadloori*, *Mahipal*, *Yashwant*, *Yogesh*, *Sudarshan*, *Prabhakar*, and *Shriram* for their valuable support and help.

Different articles, books, thesis and internet links knowingly or even unknowingly influenced me to think on Sanskrit grammar and NLG. Wherever possible I have acknowledged all these authors. I am indebted to the authors *Prof. R. Vasudevan*

Potti, *Prof. K. V. Ramakrishnamacharyulu*, *Prof. Korada Subrahmanyam*, *Prof. Amba Kulkarni*, *Prof. Prabodhachandran Nair*, *Dr. Rakesh Dash*, and others for their knowledge sharing.

I hereby express my salutations to *my parents* for giving an oppurtunity for higher studies.

My hearty thanks to the office staff of the Department of Sanskrit Studies, IGM Library and the Administration Office for providing me all sorts of infrastructural support and the scholarship.

Without mentioning all the names, finally I thank everyone who have directly or indirectly helped me to complete my research work and thesis.

……

"kuto vā nūtanaṃ vastu vayamutprekṣituṃ kṣamāḥ .

vaco vinyāsavaicitryamātramatra vicāryatām".

……

(Jayantabhaṭṭa in Nyāyamañjari)

Contents

List of Figures

List of Tables

Nomenclature

0.1 List of Abbreviations

ATM	Automated Teller Machine
CBR	Case-based Reasoning
DCS	Digital Corpus of Sanskrit
GUI	Graphical User Interface
KR	Knowledge Representation
MG	Morphological Generator
MT	Machine Translation
NL	Natural Language
NLG	Natural Language Generator
NLU	Natural Language Understanding
SCL	Sanskrit Computational Linguistic
SSG	Sanskrit Sentence Generator
ST	Sentence Generator
STT	Speech to Text
TTS	Test to Speech
WAG	Workbench for Analysis and Generation

0.2 Note

- Terms like sambandha, Aṣṭādhyāyī, kāraka, adhikaraṇa, etc., are used in Roman script without mentioning their first case-suffixe. Word-endings with vowel and strīpratyaya kept as it is like Aṣṭādhyāyī, kartā etc. Word-endings with consonants like kārakam kept as kāraka. The words which have a popular usage like 'Sanskrit', 'The Vedas', 'The Upanishads, 'Devanagari' etc., are retained in English language according to their practice with capitalising the first letter in a sentence, headings, and propor noun. To denote the plural form, 's' is added after the term. (To keep either a prātipadika or a nominative case is an unsolved problem, which causes a non-uniformity in entire text. Hence the Sanskrit terms are used niether in prātipadika-form nor in the nominative form with the case marker.)
- Terms without change denote 'Tagset of Dependency Relation', like karaṇam, sampradānaṃ etc.
- 'He' is used in general to denote he or she wherever it is not used for a translation.
- Compound words with Sanskrit and English joined with a hyphen like kāraka-role, Padī-information etc.
- The term *padī* is used to differentiate ātmanepadī, parasmaipadī and ubhayapadī from the terms *pada* (word), pada-formation etc.
- Sanskrit terms like prātipadika, vibhakti, upasarga etc., are translated at its first occurence. e.g case marker (vibhakti)
- Sūtra number is given in adhyāya, pāda, sūtra format. For example the sūtra *dhruvamapāye'pādānam(1.4.24)* denotes first adhyāya, fourth pāda, twenty-fourth sūtra of Aṣṭādhyāyī. English Translation taken from the site[1] kept inside the quotaion mark.

[1] `https://www.sanskritdictionary.com`

Dissertation Related Papers Presented at Conferences

- Madhusoodana Pai J, Sanjeev Panchal and Amba Kulkarni, presented a paper titled **Semantics of Morpho-syntactic Case Markers in Indian Languages: Sanskrit a Case Study** in the 47th AICDL & International Symposium, School of Humanities and Languages, Central University of Karnataka Gulbarga, 20-22 June 2019.
- Madhusoodana Pai J and Amba Kulkarni, presented a paper titled **Sanskrit Sentence Generator** in the International Conference on Churning of Indology conducted by Bharatiya Vidvat Parishad & Tattvasamshodhana Samsat, Udupi, 4 to 6, January, 2019.

Chapter 1

Introduction

Sanskrit language has a rich tradition of grammar. Pāṇini's grammar is known for its almost exhaustive coverage of the then prevalent Sanskrit language and is still very easily accessible with several scholars having expertise with this grammar.

1.1 Motivation and Goal of Research

Pāṇini's grammar is compared with a computer programme for its computational aspect. It provides the essential grammatical rules to generate a sentence from its meaning structure based on this grammar.

Goal of the present research is to develop a sentence generator for Sanskrit based on Aṣṭādhyāyī, using the available morphological generator.

1.2 Why Sanskrit Generator?

The question has to be answered why a Sanskrit generator is necessary before differentiating the morphological generator and a sentence generator.

1.2.1 As a part of Grammar Assistance

For a classical language like Sanskrit which is for most of the people a second language and not the mother tongue, there is a need for computational aid. This need arises for the following reasons—

Memorization: In a Gurukula, a student memorizes Aṣṭādhyāyī and all the nominal and verbal forms with its derivation (prakriyā). This system is not followed being not possible to follow for a student studying in a primary or secondary level school. Because in modern education system Sanskrit is just one subject along with several

other modern subjects. So it becomes impossible to memorise all the Śabdarūpāvalī[1]. Sanskrit is an inflectional language. That means the case suffixes (vibhaktipratyayas) get attached to the stem (prātipadika) and during the attachment some phonological changes also take place. Thus for a new learner, it becomes difficult to memorize the forms.

Grammar: Without grammar a student can't understand the vocabulary in Sanskrit and he cannot form correct forms of words without applying the exceptional rules. e.g:- *Devena*, *Rāmeṇa*. All the akārānta nominals take the forms like *Deva*, hence the instrumental case will be *Devena*. But in the case of *Rāma*, it becomes *Rāmeṇa* by the rule *raṣābhyāṃ no ṇaḥ samānapade(8.4.1)*. Further, in case of feminine forms the suffix takes totally different form as *Ramāyāḥ*.

There are a set of words in whose presence a nominal stem gets a specific case marker. For example, in the presence of *saha*, the accompanying noun gets instrumental case suffix. The noun denoting the body part causing the deformity also gets an instrumental case suffix as in *akṣṇā kāṇaḥ* (blind with eyes). Most of these rules being language specific, the learner has to remember all the relevant grammar rules.

Gender: Sanskrit nouns have genders and unlike the other languages such as Dravidian where the gender depends on the animacy, sex, number, etc., in Sanskrit the gender is an integral part of the prātipadika. That means one has to 'remember' the gender of each prātipadika. The gender has no relation to the meaning/denotation of the word. Remembering the gender of the word is important in Sanskrit. Based on masculine gender (pulliṅga), feminine gender (strīliṅga) and neuter gender (napuṃsakaliṅga) prātipadika changes. For example wife in Sanskrit can be either a *patnī* in feminine gender or *dārā* in masculine gender or *kalatra* in neuter gender.

Padī-information: Ātmanepadī, parasmaipadī and ubhayapadī are the three types of verb forms. After adding a prefix (upasarga) to a root (dhātu), sometimes ātmanepadī changes to parasmaipadī and vice versa. Remembering the padī-information of the verb becomes important to generate the correct verb forms with and without prefixes.

[1] Rūpacandrikā, Dhāturūpāvalī, Śabdamañjarī, Siddharūpa etc., are some of the other texts taught in a prakriyā-tradition to memorise all the nominal and verbal forms.

For example—*gam* is a parasmaipadīdhātu, when it is used with the prefix *sam* becomes ātmanepadī; *yūyaṃ saṅgacchadhvam*.[2]

Atmanepadī and parasmaipadī are used in a sentences to denote special usages. For instance a sentence like *Rājā rājyaṃ bhunakti*, the verb used in parasmaipadī is in the sense 'to protect' (*rakṣa*) and the same verb used in a sentence *Rājā annaṃ bhuṅkte* in ātmanepadi denotes the meaning 'to eat' (*bhuj*).[3] A user should be aware of these usages. A speaker, by mistake, if uses a wrong padī, the sentence may not convey the desired meaning.

Special Usages: Sanskrit has special usages such as passive (karmaṇi) with transitive verbs (sakarmakadhātu) and bhāve (impersonal passive) with intransitive verbs (akarmakadhātu). Since the modern Indian languages do not have such usages, the learners find it difficult to write/speak sentences with these usages.

While the general rules for assigning the case suffix (vibhaktipratyaya) to a noun representing a specific kāraka-role are very few in number, there are several special and exceptional rules. These rules are about fifty. So a learner needs to remember all these rules.

Even if a person has passive control, due to the above mentioned problems, a speaker either shys away from speaking Sanskrit or ends up in speaking wrong Sanskrit.

Finally, the influence of mother tongue on Sanskrit speaking also results in wrong/nativized Sanskrit. And a speaker who does not want to adulterate Sanskrit with the influence of his native language would like to have some assistance, and if it were by a mechanical device such as a computer, it would be advantageous. Also, the drastic shift of technology from books to digital format and use of computer-aided programmes forces a learner to use computational tools. Computational tools play a major role in all fields of knowledge and study of language is not an exception.

[2] *samo gamyṛcchipracchisvaratyartiśruvidibhyaḥ(1.3.29)* "After the verbs *gam* 'to go', *ṛcch* 'become hard', *pracch* 'to ask', *svar* 'to find fault, *ṛ* 'to go', *śru* 'to hear', and *vid* 'to know' when used intransitively and preceded by *sam* the ātmanepadī-suffix is used."

[3] *bhujo'navane(1.3.66)*

1.2.2 As a part of NLP tools

Generator is an essential component of any Machine Translation (MT) system. It is also needed in systems such as—

Information Summarisation: Information Summarisation or Automatic Summarisation is the process of shortening the document with the help of a software, in order to create a summary with the major points of the original document. 'Document summarisation tries to create a representative summary or abstract of the entire document, by finding the most informative sentences'.[4] Advantage of information summarisation helps a user to know the content without reading the entire document and which saves a lot of time. Search engines are based on information summarisation systems. For another example, 'computer-based patient record systems may already have simple report generation facilities built in, which can be used to assist the healthcare professional' [Cawsey et al. (1997)].

Question Answering: Question Answering is an automatic process capable of understanding questions formulated in a natural language and responding exactly with the requested information. "This system should be able to determine the information-need expressed in a question, locate the required information, extract it, and then generate an answer and present it according to the requirements expressed in the question". This system should allow a comfortable and appropriate interaction by users [Mollá-Aliod and Vicedo (2010)].

With these above mentioned grammar assistance problems in mind, and also the possible usages in computational linguistics like information summarisation and question answering we decided to develop a software that can assist a human, in correct sentence generation.

1.3 Can Computer Assist in Composition?

From 1990 onwards various researches are ongoing in the field of Natural Language Generation (NLG) or Sentence Generation, Speech Analysis, Text to Speech (TTS),

[4] https://en.wikipedia.org/wiki/Automatic_summarization

Speech to Text (STT), Sentence Analysis, etc. Railway ticket vending machine, Automated Teller Machine (ATM), etc., are the result of these background systems, which help a human being direct. There are tools which assist the language learner to analyse and to generate a word or sentence. Following are some of the efforts done in different areas—

1.3.1 Grammar Assistance in Other Languages

WAG Sentence Generation System: Workbench for Analysis and Generation (WAG) Sentence Generation System includes various tools for developing Systemic resources like grammars, semantics, lexicons, etc. It generates single sentences from a semantic input from a user or output of a multi-sentential text generation system. This system mainly focuses on multi-sentential concerns and does not worry about the sentential issues [O'Donnell (1997)]. In terms of speaker and listener, this system talks about generator and analyser of a sentence. Advantage of this system is that it integrates with more ease into a system intended for dialogic interaction, such as a tutoring system.

Poem Machine: "Poem Machine is an interactive online tool for co-authoring Finnish poetry with a computationally creative agent" [Hämäläinen (2018)]. In the field of computational creativity, the discourse has moved lately more and more towards human-computer co-creativity. The interest does not lie anymore on how a computer can generate creative artifacts on its own, but rather how such systems can be used together with human creativity to assist a person in a creative task. Poem Machine can suggest phonetically similar words for words the user drags into the rhymer tool. The main target group for the system is primary school children, and its use as a part of teaching is currently under study.

Advertisement Slogan Generator: It focuses on domain specific automatic generation of Japanese advertisement slogans. "The system could be regarded as the world's first practical system that can generate advertisement slogans. There have been many works published for automatic sentence generation of a variety of domains" [Iwama and Kano (2018)]. However, there would be still no single method available at present

that can generate sentences for all of the domains. Each domain requires a specific generation method.

Going Dutch: Creating Simple NLG: This is a simple NLG system developed for Dutch. For each sentence, the Simple NLG input was written manually and the resulting realization was compared with the target sentence. The aim of this task is to show to what extent the work on text generation can done by computer [Jong and Theune (2018)].

Illustration of Word Usage: This work is carried out by Leela Raj Kuchibbotla, in the University of Hyderabad for the English language. According to the author's statement, deviating from the traditional NLG, given a word the system generates a variety of sentences illustrating the usage of that word in the English language. The system can be useful in creating an electronic dictionary, computer-aided instructions, and electronic manuals and the generated sentence can be a basis for more sophisticated NLG systems incorporating semantics, pragmatics, and world knowledge [Kuchibbotla (1995)].

1.3.2 Review of Available Tools in Sanskrit

In order to develop Sanskrit Sentence Generator, first, we review the work in the area of Sanskrit Computational Linguistics (SCL).

Aṣṭādhyāyī-Simulator: It is an attempt to discover programming concepts, techniques, and paradigms employed by Pāṇini in his Aṣṭādhyāyī, similar to a computer programming language. As it is named as Prakriyāsandarśinī, it shows the prakriyā of the pada-formation. How are the rules triggered? and if more than one rule is triggered then how is the conflict resolved? etc., are implemented in such a way that helps a user to understand the word generation process in Aṣṭādhyāyī with the help of a computer. Presently it works only for the akārāntapulliṅga-nominals, showing the step by step sūtras starting from prātipadikasaṃjñā up to the pada-formation. It takes a prātipadika as an input and pada as a generated output. Important features of this system are:-

1. "Interpretation of sūtras using the metalanguage described by Pāṇini in the Aṣṭādhyāyī
2. Faithful representation of sūtras
3. Automatic triggering of rules
4. Automatic conflict resolution

In the current implementation, the rules are represented using manually coded patterns. It will be interesting to see if the machine can interpret the rules automatically based on the vibhaktis and the meta rules" [Goyal et al. (2009)]. The tool is available under Saṃsādhanī[5].

Many researchers have worked in this direction. Anand Mishra, Shidhar Subbanna, Peter Scharf and Dhaval Patel have done important contributions [Mishra (2007), Scharf (2007), Subbanna and Varakhedi (2007), and Patel and Katuri (2016)].

Verb-Generator: Verb-Generator is a combined effort by Dhaval Patel and Sivakumari Katuri [Patel and Katuri (2016)]. It generates conjugational forms in kartṛvācya, karmavācya, bhāvavācya and karmakartṛvācya, showing the sūtras applied in each stage of formation of the verb. The generated words facilitate a comparative study of dhātus mentioned in the texts like Mādhavīyadhātuvṛtti, Kṣīrataraṅgiṇī and Dhātupradīpa.[6]

Samāsa-Generator: Samastapadavyutpādikā or Compound-Generator generates a Sanskrit compound from an alaukikavigraha. It provides an interface to test example covering all the examples listed in the samāsaprakaraṇa of Siddhāntakaumudī. A detailed process with samāsavidhāyakasūtra is generated in every stage of samāsa with interactive questions to generate the desired prātipadika of the samastapada. This tool is developed by Pavankumar Satuluri [Satuluri (2015)] in the Department of Sanskrit Studies, University of Hyderabad. Limitation mentioned by the developer is as follows: it handles only the binary compounds from an input with analytical paraphrase and user interactive or semi-automatic mode of generation. It does not deal with the svaraprakaraṇa.[7]

[5] http://sanskrit.uohyd.ac.in/scl
[6] https://www.sanskritworld.in/sanskrittool/Sanskritverb/tiGanta.html
[7] http://sanskrit.uohyd.ac.in/scl

The Sanskrit Heritage Site: The Sanskrit Heritage Site provides tools for the computer processing of Sanskrit. Various tools include Stemmer, Grammar, Sandhi, Reader, and Corpus. These tools use the finite-state methods to provide efficient lexicon representation, morphology generation, and segmentation by sandhi-recognition. The Sanskrit Heritage Site is developed and maintained by Gérard Huet. Segmenter on this site is a commendable effort for its efficiency among presently available Sanskrit computational tools. All the tools are available on this[8] site.

Gaṇakāṣṭādhyāyī: This software helps to search the sūtras of Aṣṭādhyāyī with its padapāṭha and anuvṛtti. It provides the process of generating various nominal declensions and verbal conjugations following the sūtras of Aṣṭādhyāyī. Features include Dhātupāṭha, Gaṇapāṭha with its display in Devanāgarī and Roman scripts. It also shows the Siddhāntakaumudī and Laghusiddhāntakaumudī texts with its translation in French and is a notable effort done by the developer Shivamurthy Swamiji.[9]

Sanskrit Library: The Sanskrit Library is a non-profit organization which provides digitized primary texts and computerized research in Sanskrit. These tools include tools for encoding, inflectional morphology, and metrical analysis. These are the combined efforts of Peter Scharf and Hayman.[10]

JNU: There are various Computational tools developed by School of Sanskrit and Indic Studies, Jawaharlal Nehru University. It includes tools for language processing, lexical resources, e-learning and text corpora for Sanskrit and other Indian languages.[11]

Digital Corpus of Sanskrit (DCS): DCS[12] is a platform to provide a corpus of Sanskrit texts with resolved Sandhis and full morphological and lexical analysis. It aims at text-historical research in Sanskrit linguistics and philology. Advantage to a researcher/user is that this site facilitates to search lexical units (words) and their collocations in a huge corpus, word retrieval from the dictionary through a simple query or

[8] http://sanskrit.inria.fr/
[9] http://www.taralabalu.org/panini/
[10] https://sanskritlibrary.org/
[11] http://sanskrit.jnu.ac.in/index.jsp
[12] http://www.sanskrit-linguistics.org/dcs/index.php

a dictionary page, statistical evaluation based on historical principles of lexical unit; its complete set of occurrences, and interlinear lexical and morphological analysis etc.

Saṃsādhanī: Saṃsādhanī is a collective name for the Computational tools for Sanskrit, developed by the Department of Sanskrit Studies, University of Hyderabad. It includes Morphological Analyzer, Sandhi-Splitter, Sandhi-Analyser, Transliterator, Nyāyacitradīpikā, Morphological Generator, and Compound Generator. Major contribution in the field of Sanskrit computational linguistics done by Saṃsādhanī is its Sanskrit-Hindi Accessor cum Machine Translator system also known as 'Anusārakam Anuvādakaṃ ca'. The Morphological Generator which presently generates different forms of noun, verb, selected kṛt and taddhita-forms are being used by the Sanskrit Sentence Generator (SSG).[13]

1.4 Difference between a Morphological Generator and a Sentence Generator

Why a Sentence Generator is required if there is already a morphological generator? It is a basic question that has to be answered related to the Sanskrit Sentence Generator.

This is welknown that morphological generator is the base for sentence generator. But both have different functions. A Morphological Generator is limited to word level and it does not have relational information with other entities like in a sentence. But a Sentence Generator has to focus on the following factors—

Number and person: The verb form reflects the number (vacana) and person (puruṣa) of the noun with which it agrees.

- *saḥ grāmaṃ gacchati*
- *tau grāmaṃ gacchataḥ*
- *te grāmaṃ gacchanti*

According to person and the use of conjunts in a sentence the declension of the verb changes.

- *saḥ ca ahaṃ ca grāmaṃ gacchāvaḥ*

[13] `http://sanskrit.uohyd.ac.in/scl`

- *ahaṃ ca tvaṃ ca grāmaṃ gacchāvaḥ*
- *saḥ ca tvaṃ ca grāmaṃ gacchathaḥ*

Voice: According to kartari prayoga (active voice), karmaṇi prayoga (passive voice), and bhāve prayoga (passive impersonal) the case markers of a noun and hence the declension of the verbal root changes and the verbal root mutually gets different case markers. Sanskrit also allows sentences where the non-finite verbal forms (kṛdanta) act as a finite verb as in the following sentences.

- *manuṣyaḥ kukkuraṃ vyāghraṃ kṛtavān.* (ktavatu)
- *manuṣyeṇa kukkuraḥ vyāghraḥ kṛtaḥ.* (kta)

In such cases also, the inflectional suffix of the kṛdanta should agree in number and person with that of the noun in the nominative case.

Word order: Most of the Sanskrit sentences are flexible in word order. In the case of viśeṣaṇaviśeṣyabhāva, uddeśyavidheyabhāva, and special indeclinables like *kim* word order is important.

1. *saḥ kim khādati*? (*kim* is a kāraka)
2. *kim saḥ khādati*? (*kim* may be a kāraka or yes/no)
3. *saḥ khādati kim*? (*kim* only a yes/no)

Similarly another special usage is use of *tatra* before *bhavat* for respect as in—

- *kva tatrabhavatī śakuntalā*?
- *ādiṣṭaḥ asmi tatrabhavatā kāśyapena.*

Concord of qualifier and qualified: The qualifier (adjective) gets the same gender, number and case marker as that of the qualified.[14]

Upapada and karmapravacanīya: With the set of words named upapada and karmapravacanīya, the nominal stem gets special case marker. For example in a sentence *grāmam paritaḥ vṛkṣāḥ santi*, the accusative case marker assigned to *grama* by the accompanying upapada viz. *paritaḥ*. In a sentence like *japam anu prāvarṣat* the word *japa* gets accusative by the karmapravacanīya-word *anu*.

[14] yalliṅgaṃ yadvacanaṃ yā ca vibhaktirviśeṣyasya.
talliṅgaṃ tadvacanaṃ sā ca vibhaktirviśeṣaṇasyāpi.

Special kāraka-role assignement: The participants of an action are termed *kārakas*. The definitions of these kārakas are provided by Pāṇini which are semantic in nature. However, the exceptional cases make them syntactico-semantic. For example, in the presence of the prefix *adhi* with the verbs *śīṅ*, *sthā* and *as*, the locus instead of getting the default *adhikaraṇa*-role gets a *karma* (goal)-role and subsequently accusative case marker, as in *saḥ grāmam adhitiṣṭhati* (He inhabits/governs the village) where *grāma* gets a karma-role, and is not an adhikaraṇaṃ.[15]

Constrain rules: Short forms of *asmad* and *tvad*, i.e. *mā*, *me*, *nau*, *naḥ*, *tvā*, *te*, *vāṃ* and *vaḥ* are never used at the beginning of a sentence and immediately before the particle *ca*, *vā*, *eva* and *hā*.[16]

A vākya (sentence) is not only a bunch of words but each component fulfills a relation or expectancy with its chief qualificand (mukhyaviśeṣya). Pāṇini not only gives rules for generating a pada, but also gives rules to form sentences by putting these smaller word units to become part of a meaningful bigger unit. His ultimate aim is not to construct correct words but a sentence.

In this perspective Sentence Generator is different than a Morphological Generator and becomes necessary.

1.5 The Organisation of the Thesis

This thesis has the following structure.

The first chapter discusses the necessity of a morphological generator and sentence generator, and about the problems faced by a learner while learning Sanskrit grammar. As a part of NLP and Computational Linguistics, applications based on NLG such as ATM, Text to Speech, Speech to Text, etc., are discussed following some of the grammar assistance tools available for other languages. Also, a review of Sanskrit computational linguistics tools available, and relation and difference between a morphological generator and a sentence generator are discussed in this chapter.

The second chapter discusses the architecture and the software of the SSG. Different approaches of NLG namely rule-based, statistical, hybrid, Big-Data, approach

[15] *adhiśīṅsthā"sāṃ karma(1.4.46)*

[16] *anudāttaṃ sarvamapādādau(8.1.18), yuṣmadasmadoḥ ṣaṣṭhīcaturthīdvitīyāsthayorvānnāvau(8.1.20), na cavāhāhaivayukte(8.1.24)*

infusing NLG and NLU adopted for SSG are discussed here. Four levels of Aṣṭādhyāyī to transform the thoughts in the minds of a speaker into a language string suggested by [Bharati et al. (1994) and Kiparsky (2009)], three mappings between these levels, and input and output specification for the system are the other topics which are discussed here.

"The grammar analyses sentences at a hierarchy of four levels of description, which are traversed by three mappings in the direction from semantics to phonology". The third chapter gives complete details of the second mapping of morphological spell out rules namely kāraka to vibhakti mapping. For generating the substantial forms, the case marker corresponding to the kāraka-role becomes necessary. Implementation of this major task is explained in the subtopics such as assigning case marker for kāraka-relation, handling adjectives, and handling finite verbs with the sūtras in the Aṣṭādhyāyī.

In the fourth chapter, a sample grammar rule for various upapadas explaining the information flow and morpho-syntactic classification are discussed. The focus of the chapter is on the semantic classification of the upapadas which have an importance in analysis and generation of a sentence. Relations like the reference point for direction, a reference point for comparison, locus showing the domain, association, dis-association, possessor-possessee, etc., are the semantic classification of upapadas mentioned in this chapter, which becomes an integral part of machine translation.

Testing, evaluation, and interface used for input to the generator are the topics covered in the fifth chapter.

Chapter six discusses the usefulness of Sanskrit Sentence Generator as a module for MT involving Sanskrit as a target language, voice-converter, and mutual improvement of Sanskrit Parser and Generator. It concludes with the limitations as well as the usefulness of this endeavor in the field of Machine Translations and the work to be implemented.

Chapter 2

Sentence Generator: Architecture

Language distinguishes a human being from all other living beings. Language is a means of communication. Communication is a two way process where a speaker transforms his thoughts into a language string and the listener when receives this language string in the form of sound waves, deciphers the encoded information into thoughts. The former process is termed generation and the latter on analysis.

An imitation to these combined human activities leads to Natural Language Generation (NLG) and Natural Language Understanding (NLU) in the field of Natural Language Processing (NLP) and Computational Linguistics.

NLG is the process of generating meaningfull sentences/text from information provided in a logical meaning representation. It may be thought of as the reverse of NLU.

Pāṇini's grammar provides the essential grammatical rules to generate a sentence from its meaning structure, which is an abstract representation of the verbal import (śābdabodha). It is the intermediate representation from which, using Pāṇini's rules, the desired sentence can be generated. At the same time, this meaning structure also represents the dependency parse of the generated sentence.

2.1 Text Generation by Human and a Computer

There is lack of information in the fundamental problems people struggle with when generating language and text generation by computers. For example, to generate a sentence like *Rāmeṇa saha Sītā vanaṃ gacchati*, the user can not get any kind of information as to what exactly the relation between Sītā and Rāma is. Similarly from the point of view of generation, one can not expect from a user to provide the name of the relation[1] as *sahārthaḥ*.

[1] In the case of upapada, sūtras like *sahayukte'pradhāne(2.3.19)* helps to assign a case marker, but not describe the name of the relation.

Human factor is unavoidable in text generation by human being or text generation by computer. Once this problem is minimised concept dictionary[2] can be developed. These concepts then can be mapped to the lexical terms of any language. A generator for any language is tuned to perform best for that language and hence is tightly coupled with the lexical terms and the grammar of that language. This also demands an interface between the concepts and the lexical terms [Kulkarni (2019)].

2.2 NLG: Different Approaches

In the late nineties of the last millennium, several NLG systems were developed which were general purpose [Dale (2000)]. These are developed using different approaches namely template-based, rule-based, hybrid, statistics-based and trainable NLG or BIG-data, etc. Each has its advantages and disadvantages.

Template-based Approach: A template-based generation is delimited in its scope by the set of templates. A programme that sends individualised bulk emails is an example of template-based generation. Words fill in slots used in Railway announcement, dialogues interface of ATM vending machines, natural sounding output used in Metros using text to speech technology is also based on template-based sentence generation. Advantage of template generation is that it is simple, easy to develop, domain-specific, good quality in output and it requires less linguistic expertise. The disadvantage is that it lacks generality and variety, there is difficulty in maintenance and to add new words, and growth is restricted to the domain.

Rule-based Approach: A rule-based system can generate sentences without any restriction, provided the rules are complete. But developers with linguistic sophistication are required to develop and maintain the generators. Rule-based systems can produce more varied text than template based system. Advantage of the rule-based system is that it handles the problem very easily compared to other models without a huge amount of corpus.

[2] `http://www.unlweb.net/wiki/UNL_Dictionary`

Hybrid Approach: There have been efforts to mix the use of rule-based and template based generation. A hybrid case-based generator requires a small annotated corpus and its rule-based adaptation ensures the adapted sentences grammatically correct in generation. This approach is more accurate than statistical ones. A system like SEGUE [Pan and Shaw (2004)], a hybrid system that combines both case-based reasoning (CBR) and rule-based approaches for NLG. "The basic advantage of a CBR is a machine learning paradigm that reuses prior solutions to solve new problems".

Statistics-based Approach: Recently statistics-based generation systems have achieved some good success [Vicente et al. (2017)]. Initial generation can be imperfect for this approach. The main disadvantage of these generators is that they require a large corpus to train the machine. Without enough training instances, machine's performance degrades.

BIG data: The recent trend in NLG, as with all other NLP systems is to use machine learning algorithms with BIG data.

Approach Infusing NLG with NLU: Most of the recent researches focus on dependency analysis and generating the sentence, using the same dependency tree. Language generation and analysis require a considerable amount of knowledge, including domain knowledge and linguistic knowledge at various levels. Thus, NLU and NLG are not simply 'reversible' except at a very abstract level, which helps analysis and generation to find out what kinds of performance are possible and where the difficulties are. Generation and parsing thus use the same tree representations and tree-building actions throughout. [3]

With the availability of a full-fledged generative grammar for Sanskrit in the form of Aṣṭādhyāyī, it is appropriate to use a rule-based approach for building the generation module. Without appealing to the 'world knowledge'[4], using Pāṇini's rules, the desired sentence can be generated. A lot of work in the area of Sanskrit Computational linguistics has taken place in the last decade, some of which is related to the

[3] "A system that which takes raw text as input, performs NLU analysis, and then performs the NLG task of transforming the intermediate representation into English language texts is an example to this approach."

[4] sūtrās like *bhītrārthānāṃ bhayahetuḥ(1.4.25), dūrāddhūte ca(8.2.84)* etc., are exceptional which requires world knowledge.

word generators. So we decided to use the existing word generators and build a sentence generator, modeling only the sūtras that correspond to the assignment of case markers.

2.2.1 Rule-based Approach

Within a rule-based approach itself, two methods can be seen in the field of Sanskrit computational linguistics.

1. Generation simulating the Aṣṭādhyāyī
2. Generation using existing Word-Generators

Though the Aṣṭādhyāyī is compared to a computer programme, and some notable efforts in the direction of simulation of Aṣṭādhyāyī [Patel and Katuri (2016), Goyal et al. (2009), and Mishra (2007)], understanding the structure and organisation of Aṣṭādhyāyī [Subbanna and Varakhedi (2007), and Scharf (2007)] exist, but still we do not have an implementation that can generate a word from the root and suffix.

Using existing word generators to develop a sentence generator is comparatively easier. Further, such a sentence generator would produce an useful product that can be plugged-in in several other real-time products. Also, building a sentence generator using existing word generator will provide us a general architecture for building generators which can be used for other inflectionally rich languages such as Indian languages.

2.3 NLG: Input and Output

The effectiveness of the rule-based NLG depends on the efficiency of input knowledge representation often from non-linguistic input data, which will be reflected in the effectiveness of language generation also. So, the problem of NLG is twofold; selecting a Knowledge Representation (KR) as input and transforming the information to Natural language (NL) as an output. Technically it is described as 'what to say' and 'how to say' in the field of NLG.

> The above transfer can be looked at from the point of view of information. The speaker wants to convey some information to the hearer. Having decided on the information he wants to convey, he must decide how to

> code it in language. An utterance is the only thing actually received by the hearer, using which he gets the information. It follows, therefore, that the information is contained in the utterance, and the hearer must extract it by decoding it [Bharati et al. (1994)].

2.3.1 Input

The input to the generator is the thoughts in the minds of a speaker, typically represented following some kind of knowledge representation in a non-linguistic form. It is possible that this knowledge representation is generated automatically. For example, in the question answering system, based on the question of the user, computer generates a schemate that represents the knowledge to be conveyed to the user. This schemate thus is the input for the system. Alternately, if a user wants to generate a sentence corresponding to the thoughts in his minds, the thoughts are represented in a knowledge representation schemate.

Thus NLG may receive input either from some machine or from a user using some suitable Graphical User Interface (GUI). We have decided to develop the generator from the representation of śābdabodha (verbal import).

2.3.2 Output

The output is the grammatically correct sentence in a linguistic form.

2.4 SSG: Architecture

Pāṇini has given a grammar which is generative in nature. He presents a system of grammar that provides a step by step procedure to transform thoughts in the minds of a speaker into a language string. Broadly one may imagine three mappings in the direction from semantics to phonology [Bharati et al. (1994) and Kiparsky (2009)]. These levels are represented pictorially as in Figure 2.1.

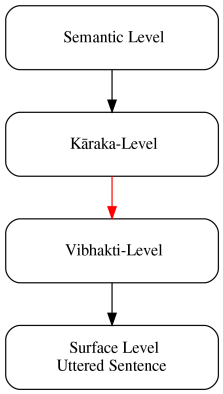

Figure 2.1: Levels in the Pāṇinian model

2.4.1 Semantic Level

This level corresponds to the thoughts in the mind of a speaker. The information is still at the conceptual level, where the speaker has identified the concept and has concretised them in his mind. The speaker, let us assume, for example, has witnessed an event where a person is leaving a place and is going towards some destination. For our communication, let us assume that the speaker has identified the traveling person as person#108, the destination as place#2019, and the action as move-travel#09. Also, the speaker has decided to focus on that part of the activity of going where the person#108 is independent in performing this activity, and that the goal of this activity is place#2019. This establishes the semantic relations between person#108 and move-travel#09 as well as between place#2019 and move-travel#09. Let us call these relations sem-rel#1 and sem-rel#2 respectively. This information at the conceptual level may be represented as in Figure 2.2.

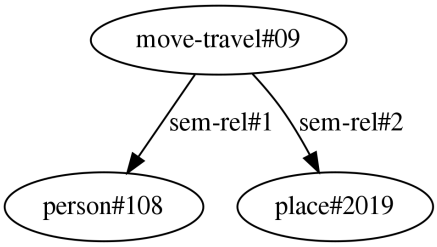

Figure 2.2: Conceptual representation of a thought

2.4.2 Kāraka-Level

In order to convey this, now the speaker chooses the lexical items that are appropriate in the context from among all the synonyms that represent each of these concepts. For example, for the person#108, the speaker chooses a lexical term, say *Rāma*, among the synonymous words *{ayodhyā-pati, daśarathanandana, sītā-pati, kausalyā-nandana, jānakī-pati, daśa-ratha-putra, Rāma, ...}*. Similarly corresponding to the other two concepts, the speaker chooses the lexical terms say *vana* and *gam* respectively. With the verb *gam* is associated the pada and gaṇa information along with its meaning.

Having selected the lexical items to designate the concepts, now the speaker chooses appropriate kāraka-labels corresponding to the semantics associated with the chosen relations. He also makes a choice of the voice in which to present the sentence. Let us assume that the speaker in our case decides to narrate the incidence in the active voice. The sūtras from Aṣṭādhyāyī now come into play. The semantic roles sem-rel#1 and sem-rel#2 are mapped to kartā and karma, following the Pāṇinian sūtras—

- *svatantraḥ kartā(1.4.54)*; which assigns a kartā role to Rāma.
- *karturīpsitatamaṃ karma(1.4.49)*; which assigns a karma role to vana.

Let us further assume that the speaker wants to convey the information as it is happening i.e., in the present tense (vartamāna-kāla). Thus at the end of this level, the available information is as shown in Figure 2.3.

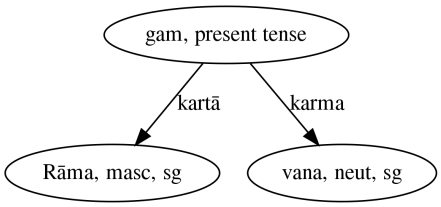

Figure 2.3: Representation in abstract grammatical terms

This information is alternately represented in simple text format as shown below.

word index	stem	features	role
1	Rāma puṃ	eka	kartā 3
2	vana napuṃ	eka	karma 3
3	gam parasmaipada bhvādi	vartamāna	kartari

The first field represents the word index which is used to refer to a word while marking the roles. The second field is the stem (with gender in case of nouns), the third field provides morphological features such as number, tense, etc. and the fourth field provides the role label and the index of the word with respect to which the role is marked. If this information is converted to verbal form, what we get the traditional śābdabodha such as *eka-vana-abhinna-karmaka,-eka-rāma-abhinna-kartṛka,-uttaradeśa-saṃyogānukūla-vartamānakālika-vyāpāraḥ.*

2.4.3 Vibhakti-Level

Now the sūtras from vibhakti-section of Pāṇini's Aṣṭādhyāyī come into play. Vana which is a karma, gets accusative (dvitīyā) case marker due to the sūtra *karmaṇi dvitīyā (anabhihite)(2.3.2).* Since the sentence is desired to be in active voice, kartā is abhihita (expressed), and hence it will get nominative (prathamā) case due to the sūtra - *prātipadikārtha-liṅga-parimāṇa-vacana-mātre prathamā(2.3.46).* The verb gets a laṭ lakāra due to vartamāna-kāla (present tense) by the sūtra -*vartamāne laṭ(3.2.123).* It also inherits the puruṣa (person) and vacana (number) from the kartā *Rāma*, since the speaker has chosen an active voice. Thus at this level, now, the information available for each word is as follows.

word index	stem	morphological features
1	Rāma puṃ	eka prathamā
2	vana napuṃ	eka dvitīyā
3	gam parasmaipada bhvādi	laṭ prathama eka

2.4.4 Surface Level

With this information, now each pada is formed using the available word generator. Sandhi at the sentence level is optional. If the speaker intends, then the sandhi-rules come into play and a sentence with sandhi is formed. Thus we get either *Rāmaḥ vanaṃ gacchati* or optionally *Rāmo vanaṅgacchati* as an output.

word index	stem	features	role
1	Rāma puṃ	eka	kartā 3
2	vana napuṃ	eka	karma 3
3	gam_1	vartamāna	kartari

Table 2.1: Input to Sentence Generator

2.5 SSG: Input and Output

In the above architecture, there are three modules:

1. A module that maps the semantic information in the form of abstract concepts and abstract semantic relations into the linguistic elements viz. the nominal/verbal stem and syntactico-semantic relations. We have not implemented this module yet. However we have conceptualised it as follows.

   A user interface is planned, to model this part, through which the speaker selects the proper lexical terms as well as declares his intention selecting the syntactico-semantic relations and the voice. The gender associated with the nominal stem is provided by the interface, and the user does not have to bother about it. The user only provides the nominal stem, chooses the number and its role with respect to the verb. In the case of verbs, the user selects the verb based on its meaning, and the information of pada and gaṇa is automatically picked by the interface, coding this information in the form of a subscript. The user also chooses appropriate relations between the words. The user interface takes care of exceptional cases hiding the language specific information from the user. The output of this module, for the example sentence under discussion, is as shown in the Table 2.1.

2. A module that maps the syntactico-semantic relations to the morpho-syntactic categories such as case marker and position (in the case of upapadas, for example)

   The next chapter will describe the second module in detail that maps the syntactico-semantic relations into morpho-syntactic categories. The input to the generator is a set of quadruplets as shown in the Table 2.1. The first element provides the index, the second the stem, the third the morphological features and the last on the relation and the index of the second relata (viz. anuyogin). The current

version recognises only following expressions for stem-feature combinations, where '?' represents optionality, '*' is the kleene operator for zero or more occurrences.

1 {Noun}{Gender}{Vacana}?

2 {Noun}{taddhita}?{Gender}{Vacana}?

3 {Upasarga}*{Verb}{Sanādi_suffix}{Kṛt_suffix}{Vacana}?

4 {Upasarga}*{Verb}{Sanādi_suffix}{prayoga}{lakāra}

Vacana is not specified if it has an adjectival relation with other words. This representation is the same as the internal representation of the output of the Saṃsādhanī[5] parser. We call this representation, an intermediate form, or the meaning structure. It represents the verbal import of the sentence in abstract form, hiding the details of which linguistic unit codes what information.

3. A module that composes a surface form/word form from the morphological information.

The third module corresponds to the word generation. Given the morphological information, this module produces the correct form of the word. For this module, the word-generator developed in-house[6], which is also a part of Saṃsādhanī tools is being used. We decided to produce the output in unsandhied form. Hence, for this example, the output would be *Rāmaḥ vanaṃ gacchati*.

Conclusion

In this chapter we have discussed various NLG approaches and its input and output specification, SSG architecture consisting four levels from semantics to surface-level with its three mappings, internal representation[7] of the input in the prescribed format and the output generated by the system. Thus this chapter becomes a basis to explain the second mapping from karaka-level to vibhakti-level namely morphological spellout module which is the core of the SSG.

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

[6] `http://sanskrit.uohyd.ac.in/scl`

[7] In NLU the input to the system always be a linguistic information, but in the case of NLG it will be a non-linguistic information.

Chapter 3

Kāraka to Vibhakti Mapping

"The rules of the Aṣṭādhyāyī fall into three broad classes, each of which effects a mapping between what we could see as different levels of representation. Thus, the grammar analyzes sentences at a hierarchy of four levels of description, which are traversed by three mappings in the direction from semantics to phonology". [Kiparsky (2009)]

Section 2.4.1 discusses four levels of sentence generation from semantic to surface level and section 2.5 explains its three mappings, namely—

1. Assignment of kārakas and abstract tense
2. Morphological spellout rules
3. Allomorphy and phonology

The focus of this chapter is on the second module viz. morphological spellout rules.

The thoughts in the mind of a speaker which are semantic in nature are first transformed into syntactico-semantic relations. Rules for these transformations are provided by Pāṇini in his Aṣṭādhyāyī under the adhikāra of *anabhihite(2.3.1)*. These sūtras map these syntactico-semantic relations to the corresponding abstract morphological information such as vibhakti. Abstract tense is replaced with the case marker by the sūtras *vartamāne laṭ(3.2.123)* etc. Finally the rules for pada-formation come into play to generate the words. In this chapter we discuss the implementation of all these rules.

3.1 Morphological Spellout Module

There are four major tasks that are carried out in this module.

1. Assigning case marker to the substantive based on its syntactico-semantic role, In Pāṇini's grammar we come across 3 different types of case marker assignment. They are
   (a) Case marking for a kāraka-relation,
   (b) Case marking in the presence of certain words called upapadas,

(c) Case marking expressing the noun-noun relations

All these sūtras are found in the third section of the second chapter of Aṣṭādhyāyī from 2.3.1 till 2.3.50.

2. Inheriting morphological features of the adjectives from their heads
3. Assigning morphological features for finite verbs such as person and number, and
4. Assigning lakāra corresponding to the tense, aspect and modality of the verb.

Now we explain each of these steps below.

3.1.1 Assigning Case Marker

For generating the substantial forms, we need the case marker corresponding to the kāraka-role. Pāṇini assigns the vibhakti if it is not alread expressed by other means such as verbal suffix or kṛdanta-suffix. The adhikārasūtra here is *anabhihite(2.3.1)*. The vibhakti for six kārakas is given by—

- *kartṛkaraṇayostṛtīyā(2.3.18)*

  "In denoting the agent (*svatantraḥ kartā(1.4.54)*), or the instrument (*sādhakatamaṃ karaṇam(1.4.42)*), the third case-marker (tṛtīyāvibhakti) is used".
- *karmaṇi dvitīyā(2.3.2)*

  "When the object is not denoted by the termination of the verb i.e. when the verb does not agree with it, the second case-marker (dvitīyā vibhakti) is attached to the word".
- *caturthī sampradāne(2.3.13)*

  "In denoting the sampradānakāraka (*karmaṇā yamabhipraiti sa sampradānam (1.4.32)*), the fourth case-marker (caturthī vibhakti) of the Dative is used after the noun".
- *apādāne pañcamī(2.3.28)*

  "When the apādānakāraka (*dhruvamapāye'pādānam(1.4.24)*), is denoted, the fifth case-marker (pañcamī vibhakti) is used".
- *saptamyadhikaraṇe ca(2.3.36)*

  "The seventh case-marker (saptamī vibhakti) is used when the sense is that of location (*ādhāro'dhikaraṇam(1.4.45)*), as well as after the words meaning 'distant' and 'near'".

These are the default vibhaktis. When the kartā (karma) is expressed by the verbal suffix, then kartā (karma) gets the nominative case suffix by *prātipadikārthaliṅga-parimāṇavacanamātre prathamā*(2.3.46).

However, there are some problems, from the user's perspective, in the selection of a kāraka. Under the adhikāra of *kārake(1.4.23)* Pāṇini provides semantic definitions of kārakas. Each of these definitions is followed by a list of exceptional cases through which Pāṇini extends the scope of the semantic definitions of the kāraka. These extensions are of two types.

1. Where the associated semantics is totally different from normal expectations. For example, In the presence of the prefix *adhi* with the verbs *śīṅ*, *sthā* and *as*, the locus instead of getting the default adhikaraṇaṁ-role, gets a karma (goal)-role, as in *saḥ grāmam adhitiṣṭhati* (He inhabits/governs the village) where *grāma* gets a karma-role, and is not an adhikaraṇaṁ. Now this is an exception to the rule, and only the native speaker of Sanskrit might be aware of this phenomenon. The user, based on his semantic knowledge, would consider *grāma* a locus, and the generator then will fail to generate the correct form. In such cases, we propose an user interface that alerts the user to provide correct kāraka-assignment.

2. Another problem is with cases of exceptions under apādāna and sampradāna. For a verbal root *bhī* to mean *to be afraid of*, according to Pāṇini's grammar, the source of fear is termed apādāna. But this is not obvious to a user who has not studied Pāṇini's grammar. He may treat it as a cause. Similarly, in the case of motion verb *gam*, the destination, according to the Pāṇini's grammar is a karma, but due to the influence of native language such as Marathi or Malayalam, the speaker may think it as an adhikaraṇaṁ. In such cases as well, we propose an user interface that alerts the user to provide correct kāraka-assignment.

Another case is of the relation between two nouns such as part and whole, kinship relations, or relation showing the possession, as in *vṛkṣasya śākhā* (the branches of a tree), *Daśarathasya putraḥ* (son of Dasharatha) and *Rāmasya pustakam* (Rama's book). In all these cases Sanskrit uses a genitive case. Pāṇini does not discuss the semantics associated with all such cases, neither he proposes any semantic role in such cases. He deals with all such cases by a single rule *ṣaṣṭhī śeṣe(2.3.50)* assigning a genitive

case in all the residual cases. While for analysis purpose, it is sufficient to mark it as a generic relation, for the generation purpose, the user would like to specify the semantics associated with it as part-and-whole-relation, or kinship, etc.

But in the corrent implementation, our generator expects ṣaṣṭhī (genitive) as the relation. Here also, an user interface should allow the user to provide fine grain semantic relations such as *avayava-avayavī*, or *janya-janaka* or *sva-svāmibhāva* etc. and then this interface should map all these relations internally to genitive (ṣaṣṭhī) before calling the generator. As of now we have not implemented any user interface and hence expects from user the correct relations.

One more set of relations between nouns is due to the upapadas (accompanying words). In the presence of an upapada, the accompanying word gets a specific case marker. For example, in the presence of *saha*, the accompanying word gets an instrumental case. This is again language specific, and hence non-native speakers of Sanskrit may go wrong in speaking sentences that involve upapadas. We discuss handling of upapadas in the next chapter.

Handling Causatives: In Sanskrit a causatives suffix (ṇic) is added to the verbal root to change the sentence from non-causative to causative. In kartari ṇic prayoga, the prayojakakartā being expressed by the verbal suffix gets nominative case. If the verb is transitive, the karma gets dvitīyā vibhakti by anabhihite karmaṇi dvitīyā. The prayojyakarma however behaves in a different way with different verbs. Next, in the case of karmaṇi ṇic prayoga, karma being abhihita gets nominative case and prayojakakartā gets instrumental case. Now when the verb is dvikarmaka, which of the two karmas is expressed and which is unexpressed is decided on the basis of the verbal root. In the case of verbal roots *duh, yāc, pac, daṇḍ, rudhi, pracchi, chī, brū, śāsu, jī, math, muṣ* mukhyakarma gets accusative case gauṇakarma gets nominal case. In the case of verbal roots *nī, hṛ, kṛṣ, vah* gauṇakarma gets accusative case and mukhyakarma gets nominal case [1]. Following Pāṇini's grammar, we have classified the verbs into semantic classes as below.

- akarmaka (intransitive)
- sakarmaka (transitive)

[1] *pradhānakarmaṇyākhyeye lādīnāhurdvikarmaṇām . apradhāne duhādīnām ...(akathitaṃ ca (Mahābhāṣyam))*

  - verbs in the sense of to motion, knowledge or information, eating and the verbs which have literary work as their object
    * verbs in the sense of motion

- dvikarmaka (ditransitive)-type 1
- dvikarmaka (ditransitive)-type 2

This list then takes care of the proper vibhakti assignment in all the type of causatives. See Appendix.1 for the summary of all rules.

3.1.2 Handling Adjectives

Consider the following input to the system, which has viśeṣaṇa in it.

word index	stem	features	role
1	vīra		viṣeṣaṇam 2
2	Rāma puṃ	eka	kartā 3
3	vana napuṃ	eka	karma 3
4	gam_1	vartamāna	kartari

Table 3.1: Example with adjective

Note here that no morphological features have been provided for the viśeṣaṇa. In order to generate the correct word form of the word vīra, we need its gender, number, and case (liṅga, vacana, vibhakti). Only information available to the generator from the user is that *vīra* is a viśeṣaṇa of the second word. The required information is inherited from the parent node i.e. the viśeṣya. If the adjective is a derived participle form of a verb, which itself may have kāraka expectancies, we provide the necessary verbal root and the participle suffix also as input parameters for generation. For example, in Table 5.1, vyūḍhaṁ is an adjective of pāṇḍavānīkaṁ, and the stem and the features for it are provided as *vi+vah1* and *bhūtakarma* respectively. For example, the sūtra *gatibuddhipratyavasānārthaśabdakarmākarmakāṇāmaṇi kartā sa ṇau(1.4.52)* assigns a karma role and hence accusative case suffix to the prayojya-kartā, if the verb has one of the following meaning - motion, eating, knowledge or information related, or it is a verb with literary work as a karma or it is an intransitive verb.

3.1.3 Handling Finite Verbs

In the case of verb form generation, the verb form generator needs the information of

- padī,
- gaṇa,
- puruṣa,
- vacana, and
- lakāra.

to generate the verb form.

Pāṇini has given sūtras to assign lakāras for different tense and mood. They are

- *luṅ(3.2.110)*

  "The affix luṅ (Aorist) comes after the verb in the sense of past time".
- *anadyatane laṅ(3.2.111)*

  "The affix laṅ (Aorist) comes after a verbal root used in the sense of past before the commencement of the current day".
- *abhijñāvacane lṛṭ(3.2.112*)

  "When a word implying 'recollection' is in connection with it, a verb takes the affix lṛṭ (2nd Future) in the sense of the past before the commencement of the present day".
- *parokṣe liṭ(3.2.115)*

  "The affix liṭ (Perfect Tense) comes after the verb in the sense of the past before the commencement of the current dayand unperceived by the narrator".
- *vartamāne laṭ(3.2.123)*

  "The affix laṭ (Present Tense) comes after a verb when denoting a present action".
- *laṭaḥ śatṛśānacāvaprathamāsamānādhikaraṇe(3.2.124)*

  "The affixes śatṛ and śānac are substitutes of laṭ (Present Tense) when agreeing with what does not end with the 1st (Nominative Case) case-affix".
- *lṛṭ śeṣe ca(3.3.13)*

  "The affix lṛṭ (2nd Future) is used after a verb in the remaining cases where futurity pure and simple is indicated and also where there is construction with it another verb denoting an action performed for the sake of the future action".
- *lṛṭaḥ sad vā(3.3.14)*

  "The affixes called sat i.e. śatṛ and śānac are optionally the substitutes of lṛṭ (2nd Future)".

- *anadyatane luṭ(3.3.15)*

  "The affix luṭ (1st Future) comes after a verbal root in the sense of what will happen but not in the course of the current day".

- *kṣipravacane lṛṭ(3.3.133)*

  "When the word kṣipra 'quickly' or its synonym is in construction with the verb, the future affix lṛṭ (2nd Future) is used after the root when 'hope' is expressed in a conditional form".

- *āśaṃsāvacane liṅ(3.3.134)*

  "The affix of the Potential is used in denoting Futurity after a root, when the upapada in composition with it is a word expressing 'hope'".

- *hetuhetumatorliṅ(3.3.156)*

  "The affix liṅ (Benedictive) is optionally used after those verbs which express the condition and its consequence".

- *loṭ ca(3.3.162)*

  "The affix loṭ (Imperative) is used after a root in the sense of commanding etc".

- *āśiṣi liṅloṭau(3.3.173)*

  "The affixes liṅ (Benedictive) and loṭ (Imperative) come after a verb by which 'benediction' is intended".

These sūtras are implemented as a hash data structure that maps the tense and mood to the lakāra. The voice determines the person and number of the verbal form. If the voice is kartari (karmaṇi), then the person and number information is inherited from the kartā(karma). In the case of impersonal passive (bhāve), the person and number are assigned the values third (prathama-puruṣa) and singular (eka-vacana) respectively. A note on the information of puruṣa is in order. As we notice, the information of person is not provided with a noun stem in the input. Then from where does the machine get this information? Here we use Pāṇini's sūtras:

- *yuṣmadyupapade samānādhikaraṇe sthāninyapi madhyamaḥ(1.4.105)*

  "When the pronoun yuṣmad 'thou' is understood, and also when the same is expressed, the attendant word is in agreement with the verb, then there is the verbal termination called the middle (second person)".

- *asmadyuttamaḥ(1.4.107)*

  "When the pronoun asmad 'I', is understood and also when expressed, the at-

tendant word is in agreement with the verb, then there is the verbal termination called the Highest or first person".

- *śeṣe prathamaḥ(1.4.108)*

  "In the other cases, viz. where 'thou' or 'I' are not the attendant words in agreement with the verb, there is the verbal termination called the Lowest (third person)".

Next comes the information about padī and gaṇa. We notice that, though the majority of the verbs belong to a single gaṇa, there are several dhātus which belong to more than one gaṇa. For example the very first dhātu in the dhātupāṭha viz *bhū* belongs to two different gaṇas viz bhvādi and curādi. It is the meaning which distinguishes one from the other. *Bhū* in bhvādigaṇa is in the sense of sattāyām (to exist) and the one in the curādigaṇa is in the sense of prāptau (to acquire). A detailed study of the verbs belonging to different gaṇas is carried out by [Shailaja (2014)]. She has indexed these dhātus for distinction. The verb generator of Saṃsādhanī uses these indices to distinguish between these verbs. The speaker, on the other hand, would not be knowing these indices. So we provide a user interface to the user wherein the user can select the dhātu, gaṇa and its meaning, and the interface assigns a unique desired index automatically.

If a verb has ubhayapadī both the parasmaipadī and ātmanepadī forms would be generated. Otherwise only the form with associated padī would be generated. Certain verbs use different padīs to designate different meanings. For example, the verb *bhuj* has two meanings viz. *to eat* and *to rule* or *to govern*. In the sense of *to eat*, the verb has only ātmanepadī-forms and in the sense of *to govern*, it has only parasmaipadī-forms. In such cases, the user interface is desirable to hide all these complexities from the user.

Chapter 4

Handling *Upapadas*

Upapada is defined as *samīpasthaṃ padaṃ* (A word standing near or accompanying other) and the case marker (*vibhakti*) which is used in connection with the *upapada* is known as *upapadavibhakti*.[1] It denotes a non-thematic relation in a sentence.[2]

For example, Pāṇini in the rule *saha yukte'pradhāne(2.3.19)* states "when the word *saha*, is joined to a word the latter takes the third case". It means the word *saha* always expects a third case word. e.g.

3. **Skt:** ***Rāmeṇa saha Sītā vanaṃ gacchati***
   Gloss: Rāma {inst.} with Sītā{nom.} forest{acc.} goes.
   Eng: Sita goes to the forest with Rama.

Classification of Upapadas

Upapadas may be classified in two different ways.

- Based on their morpho-syntactic properties
- Based on the semantics they express

4.1 Morpho-syntactic Level Classification

Based on the case marker upapadas govern, upapadas may be classified into six classes as follows with example and respective sūtras—

[1] upapadaṃ āśritya jāyamānā vibhaktiḥ upapadavibhaktiḥ
[2] kāraka and kāraketara relations roughly translated as thematic and non-thematic relations but they are not the same. See [Bharati and Kulkarni (2011) and Ramakrishanamacharyulu (2009)]

4.1.1 Upapadadvitīyā

4. **Skt**: ***grāmaṃ abhitaḥ vṛkṣāḥ santi.***

   Gloss: village{acc.} both the side{loc.} tree{nom.} be

   Eng: There are trees on both side of the village.

The word *abhitaḥ* assigns an accusative case (dvitīyāvibhakti) to the word which it accompanies. In this example, *grāma* gets the accusative case marker. In the same manner, the words *paritaḥ*, *samayā*, *nikaṣā*, etc., assign an accusative case marker. Following Pāṇinī's sūtras provide the list of such words.

- *antarāntareṇayukte(2.3.4)*

  "A word joined with (or governed by) the word *antarā* or *antareṇa* takes the accusative case marker".
- *abhitaḥparitaḥsamayānikaṣāhāpratiyogeṣu ca dṛśyate. (Vārttikam)*

  "Words accompanying *abhitaḥ, paritaḥ, samayā, nikaṣā, hā, prati*, also gets the accusative case marker".
- *ubhasarvatasoḥ kāryā dhiguparyādiṣu triṣu*

  dvitīyāmreḍitānteṣu tato'nyatrāpi dṛśyate. (*kārikā on 2.3.2*) "Words accompanying *ubhayataḥ, sarvataḥ, dhik, uparyupari, adhyadhi, adho'dhaḥ* get accusative case marker".
- *enapā dvitīyā(2.3.31)*

  "With a word ending with the affix *enap* (*enabanyatarasyāmadūre'pañcamyāḥ (5.3.35)*), the accusative case marker is used, as well as the sixth (*ṣaṣṭhyatasarthapratyayena (2.3.30)*)".
- *pṛthagvinānānābhistṛtīyānyatarasyām(2.3.32)*

  "When joined with the words *pṛthak* 'without', *vinā* 'without' and *nānā* 'without', the third case-affix is used, optionally (as well as the fifth and the second with the sūtras *apādāne pañcamī(2.3.28)*, and *enapā dvitīyā(2.3.31)*)".

4.1.2 Upapadatṛtīyā

5. **Skt**: ***Rāmaḥ keśaiḥ prasitaḥ asti***

   Gloss: Rāma{nom.} hair{inst.} engrossed{nom.}

   Eng: Rāma is engrossed with hairs.

Here the word *prasitaḥ* governs the case marker of *keśa* and assigns third case (tṛtīyāvibhakti) and seventh case optionally. Following sūtras were reffered for making a list of similar upapadas.

- *tulyārthairatulopamābhyāṃ tṛtīyānyatarasyāṃ(2.3.72)*
  "The third or the sixth case-affix (*ṣaṣṭhī śeṣe(2.3.50)*) may optionally be employed, when the word is joined with another word meaning 'like to or resemblance', excepting *tulā* and *upamā*".
- *pṛthagvinānānābhistṛtīyānyatarasyām(2.3.32)*
  "When joined with the words *pṛthak* 'without', *vinā* 'without' and *nānā* 'without', the third case-affix is used, optionally (as well as the fifth and the second)".
- *prasitotsukābhyām tṛtīyā ca(2.3.44)*
  "In conjunction with the words *prasita* and *utsuka* 'greatly desirous of', the third case-affix is used after the word, as well as the seventh".
- *sahayukte apradhāne(2.3.19)*
  "When the word *saha* 'with', is joined to a word the latter takes the third case, when the sense is that the word in the third case is not the principal but the accompaniment of the principal thing".

4.1.3 Upapadacaturthī

6. **Skt**: ***Gurave namaḥ***
   Gloss: Teacher{dat.} salutations{ind.}
   Eng: Saluations to the Teacher

Here the word *namaḥ* has an expectency of dative case (caturthīvibhakti). Following sūtras were referred for making a list of similar upapadas which governs the dative case marker.

- *caturthī cāśiṣyāyuṣyamadrabhadrakuśalasukhārthahitaiḥ(2.3.73)*
  The fourth as well as the sixth-case-affix may be used, when blessing is intended in connection with the words *āyuṣa* 'long life', *madra* 'joy', *bhadra* 'good fortune', *kuśala* 'welfare', *sukha* 'happiness', *artha* 'prosperity' and *hita* 'good'.
- *namaḥsvastisvāhāsvadhālaṃvaṣaḍyogācca(2.3.16)*
  The fourth case-affix is used in conjunction with the words *namaḥ* 'salutation',

svasti 'peace', *svāhā* , *svadhā* (terms used in offering oblations to Gods and pitṛs, respectively), *alam* , 'a match for', 'sufficient for' and *vaṣaṭ* , a term for oblation.

- *alamiti paryāptyarthagrahaṇam*

4.1.4 Upapadapañcamī

7. Skt: ***grāmāt dūraṃ parvataḥ asti***.
   Gloss: village{abl.} far away{loc.} mountain{nom.} be
   Eng: The mountain is far away from the village.

When in conjunction with words having the sense of *dūra* (distant) the fifth case marker (pañcamīvibhakti) is optionally used. The word *dūram* indicates the semantic information of location. Following *sūtras* give further information.

- *anyārāditarartedikśabdāñcūttarapadājāhiyukte(2.3.29)*
  "When a noun is joined with words meaning 'other than' or with *ārāt* 'near or remote', or *itara* 'different from' or *ṛte* 'without', or words indicative of 'directions' (used also with reference to the time corresponding to them) or with words having *añcu* 'to bend' as the last member of the compound and expressive of direction, or with words ending with the affix *āc* (*dakṣiṇādāc(5.3.36)*) or *āhi* (*āhi ca dūre(5.3.37)*), the fifth case-affix is used".
- *apaparibahirañcava pañcamyā(2.1.12)*
  "The words *apa , pari , bahi* and indeclinables ending in *añcu* may optionally be compounded with a word ending in the fifth case affix and the compound so formed will be avyayībhāva".
- *dūrāntikārthaiḥ ṣaṣṭhyanyatarasyām(2.3.34)*
  "When in conjunction with words having the sense of *dūra* 'distant' and *antika* 'near', the sixth-case-affix is optionally used".
- *pṛthakvinānānābhistṛtīyānyatarasyām(2.3.32)*
  "When joined with the words *pṛthak* 'without', *vinā* 'without' and *nānā* 'without', the third case-affix is used, optionally (as well as the fifth and the second)".

4.1.5 Upapadaṣaṣṭī

8. **Skt**: ***grāmasya purastāt parvataḥ asti.***

   Gloss: village {gen.} in front of {loc.} mountain {nom.} be

   Eng: The mountain is in front of the village.

Hereby the rule *ṣaṣṭhyatasarthapratyayena(2.3.28)*, Pāṇini states that genitive case (ṣaṣṭī-vibhakti) is used when there is a connection with words ending with affixes having the sense of the affix *atasuc* (*dakṣiṇottarābhyāmatasuc(5.3.28)*), The affix *atasuc* comes in the sense of 'direction', 'locality' or 'time' after the words *dakṣiṇa* and *uttara.* Here the word *purastāt* derived with the *atasuc* suffix. Hence it governs the case marker of *grāma.* Remaining list of words mentioned in the following sūtras.

- *enapā dvitīyā(2.3.31)*

  "With a word ending with the affix *enap* (*enabanyatarasyāmadūre'pañcamyāḥ (5.3.35)*), the accusative case marker is used, as well as the sixth (*ṣaṣṭhyatasarthapratyayena (2.3.30)*)".
- *caturthī cāśiṣyāyuṣyamadrabhadrakuśalasukhārthahitaiḥ(2.3.73)*

  "The fourth as well as the sixth-case-affix may be used, when blessing is intended in connection with the words" *āyuṣa* 'long life', *madra* 'joy', *bhadra* 'good fortune', *kuśala* 'welfare', *sukha* 'happiness', *artha* 'prosperity' and *hita* 'good'".
- *tulyārthairatulopamābhyāṃ tṛtīyānyatarasyāṃ(2.3.72)*

  "The third or the sixth-case-affix may optionally be employed, when the word is joined with another word meaning 'like to or resemblance', excepting *tulā* and *upamā*".
- *dūrāntikārthaiḥ ṣaṣṭhyanyatarasyām(2.3.34)*

  "When in conjunction with words having the sense of *dūra* 'distant' and *antika* 'near', the sixth-case-affix is optionally used".
- *pṛthakvinānānābhistṛtīyānyatarasyām(2.3.32)*

  "When joined with the words *pṛthak* 'without', *vinā* 'without' and *nānā* 'without', the third case-affix is used, optionally (as well as the fifth and the second)".
- *ṣaṣṭhyatasarthapratyayena(2.3.28)*

  "The sixth-case-affix is used when used in connection with words ending with affixes having the sense of the affix *atasuc* (*dakṣiṇottarābhyāmatasuc(5.3.28)*)".

- *svāmīśvarādhipatidāyādasākṣipratibhūprasūtaiśca(2.3.39)*

  "The sixth and the seventh case-affixes are used after words when they are joined with *svāmin* 'master', *īśvara* 'lord', *adhipati* 'ruler', *dāyāda* 'an heir', *sākṣin* 'witness', *pratibhū* 'a surety' and *prasūta* 'begotten'".

4.1.6 Upapadasaptamī

9. **Skt:** ***Rāmaḥ pitari sādhuḥ asti***

   Gloss: Rāma {nom.} father {loc.} good {nom.} be.

   Eng: Rāma is well mannered to his father.

Here the word *sādhu* governs the case marker of *pitṛ* and assigns a locative case marker (saptamīvibhakti). Remaining list of words are mentioned in the following *sūtras.*

- *āyuktakuśalābhyāṃ cāsevāyām(2.3.40)*

  "In conjunction with the words *āyukta* 'engaged', and *kuśala* 'skilful', when meaning entire absorption in an engagement, the sixth and the seventh case-affixes are used after a word".
- *prasitotsukābhyām tṛtīyā ca(2.3.44)*

  "In conjunction with the words *prasita* and *utsuka* 'greatly desirous of', the third case-affix is used after the word, as well as the seventh".
- *sādhunipuṇābhyām arcāyāṃ saptamyaprateḥ(2.3.43)*

  "In conjunction with the words *sādhu* 'good' and *nipuṇa* 'skilfull' when they denote respect, the seventh case-affix is used, provided that the word prati is not used".
- *svāmīśvarādhipatidāyādasākṣipratibhūprasūtaiśca(2.3.39)*

  "The sixth and the seventh case-affixes are used after words when they are joined with *svāmin* 'master', *īśvara* 'lord', *adhipati* 'ruler', *dāyāda* 'an heir', *sākṣin* 'witness', *pratibhū* 'a surety' and *prasūta* 'begotten'".

Complete list of expectancies included in the Appendix .3.

4.2 Semantic Level Classification

The morpho-syntactic classification does not provide any information about the semantics expressed by the relation between upapada and the substantive whose case is governed by the upapada. Pāṇini does not discuss any semantics associated with the upapadas. Therefore in the tagging guidelines, prepared for the annotation of dependency relation,[3] this relation was termed as *upapadasambandhaḥ*. So according to these guidelines, the analysis of the sentence

10. **Skt:** ***grāmam paritaḥ vṛkṣāḥ santi***
    Gloss: forest {acc.}, surround {nom}, tree {nom}, be
    Eng: Trees surround the village.

would be as shown in Figure 4.1.

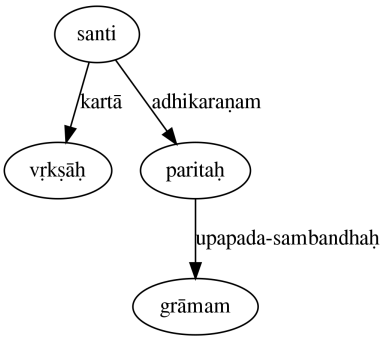

Figure 4.1: Morpho-syntactic analysis of the sentence (10)

Now the question is how easy or natural it is to comprehend this analysis for a person who does not know Sanskrit grammar? Different upapadavibhaktis are very much specific to Sanskrit language. Most of the modern Indian languages use genitive case marker. The concept of upapadavibhakti is very much specific to Sanskrit language. So such an analysis is not completely comprehensible for a person who does not know the term *upapadasambandhaḥ*. Also from the point of view of generation, it is unrealistic to expect from a user who wants to take help of machine to generate a Sanskrit sentence to provide an *upapadasambandhaḥ* as a relation between *grāmam*

[3] available at
`http://sanskrit.uohyd.ac.in/scl/GOLD_DATA/Tagging_Guidelines/tag_proposal_consortium_28Oct2014.pdf`

and *paritaḥ*. Again somebody who does not have any background of Sanskrit grammar, and who is interested in using machine to generate a sentence, a relation such as *upapadasambandhaḥ* being not semantic the user may fail to specify this relation. However, if we can provide semantics associated with this relation, it would be easy and natural for any user to specify the analysis of a sentence with such a relation. Further any user can understand the analysis produced by machine if the relations were semantic rather than morphosyntactic. We studied the semantics associated with the upapadas governing all the six upapadavibhaktis. It was noticed that the relations may be classified as follows.

- Reference point for direction (*sandarbhabinduḥ*)
- Reference point for comparison (*tulanābinduḥ*)
- Locus showing the domain (*viṣayādhikaraṇam*)
- Determination (*nirdhāraṇam*)
- Purpose (*prayojanam*)
- Exclamatory (*udgāravācakaḥ*)
- Predicative Adjective (*kartṛsamānādhikaraṇam*)
- Association (*saha-arthaḥ*)
- Dis-association (*vina-arthaḥ*)
- Possessor-possessee (*svāmī*)
- Source (*srotaḥ*)

We discuss each of them in detail.

4.2.1 Reference Point for Direction (sandarbhabinduḥ)

Let us consider the previous example sentence(10). *Paritaḥ* is a word which has an expectancy of a reference point. Here *grāma* acts as a reference point for the word *paritaḥ*. Hence we call this relation a reference point for direction, and analyse sentence (10) as in Figure 4.2.

Note here that the case assigned by each of these upapadas is different. But still we have grouped them into one class since for all these upapadas the word which they accompany are the point of reference.

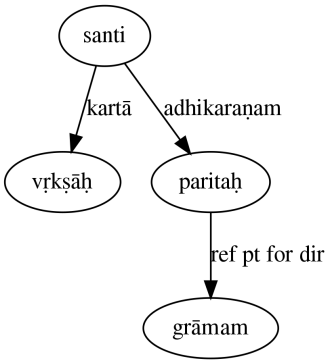

Figure 4.2: Semantic analysis of the sentence (10)

A list of upapadas[4] having reference point of direction as follows—*ārāt*, *abhitaḥ*, *abhyāśā*, *abhyāśān*, *abhyāśāt*, *abhyāśaḥ*, *abhyāśam*, *abhyāśe*, *abhyāśena*, *adhaḥ*, *adharāt*, *adhareṇa*, *adhastāt*, *adhodhaḥ*, *adho'dhaḥ*, *adhyadhi*, *agrataḥ*, *antaḥ*, *antikān*, *antikāt*, *antikam*, *antike*, *antikena*, *avāc*, *avācī*, *avaḥ*, *avarastāt*, *avarataḥ*, *avastāt*, *bahiḥ*, *itarā*, *itara*, *nikaṣā*, *nikaṭān*, *nikaṭāt*, *nikaṭam*, *nikaṭe*, *nikaṭena*, *pūrvā*, *pūrvān*, *pūrva*, *paścāt*, *paścimā*, *paścimān*, *paścimaṃ*, *parastāt*, *parataḥ*, *paritaḥ*, *prāñca*, *prācī*, *pratīcī*, *pratyañca*, *pratyak*, *puraḥ*, *purastāt*, *puratāḥ*, *samīpān*, *samīpāt*, *samīpam*, *samīpe*, *samīpena*, *samayā*, *sarvataḥ*, *ubhayataḥ*, *upari*, *upariṣṭāt*, *uparyupari*, *uttarā*, *uttarāhi*, *uttarān*, *uttarāt*, *uttaram*, *uttarataḥ*, *uttare*, *uttareṇa*, *udīcī*, *udac*, *udak*, *viprakṛṣṭā*, *viprakṛṣṭān*, *viprakṛṣṭāt*, *viprakṛṣṭam*, *viprakṛṣṭe*, *viprakṛṣṭena*, *dūrā*, *dūrān*, *dūrāt*, *dūram*, *dūre*, *dūreṇa*, *dakṣiṇā*, *dakṣiṇāhi*, *dakṣiṇān*, *dakṣiṇāt*, *dakṣiṇam*, *dakṣiṇataḥ*, *dakṣiṇe*, *dakṣiṇena*.

4.2.2 Point of Reference for Comparison (tulanābinduḥ)

There is another set of upapadas which require a point of reference for comparison. For example, in the sentence

11. **Skt**: ***Rāmaḥ Śyāmena tulyaḥ asti***.

    Gloss: Rāma {nom.}, Śyāma {inst.}, comparable {nom.} be

    Eng: Ram is comparable to Syāma.

[4] In the entire list (see Appendix .3, Appendix .4), we can see words with suffix and without suffix. Here the term without case marker (prātipadika) like *utsuka* shows its variation in all the forms in pulliṅga and napuṃsakaliṅga, and in all the number (vacana). To cover the strīliṅga separate entry like *utsukā* is kept and exact words like *paritaḥ, abhyāśāt* etc., kept as it is. Terms like *adhareṇa* is an indeclinable (subantapratirūpaka-avyaya) constructed with the suffix *enap* is to be differentiated from the subanta-forms.

Rāma is being compared with Śyāma, and thus Śyāma is the reference point for comparison. See Figure 4.3.

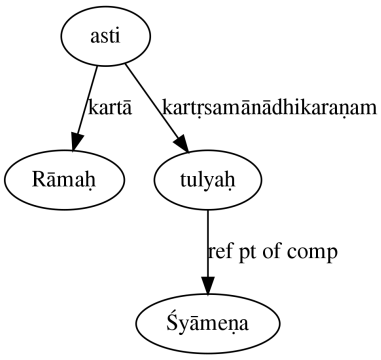

Figure 4.3: Semantic analysis of the sentence (11)

A list of upapadas having reference point of comparison as follows—*bhinnā*, *bhinna*, *anyā*, *anya*, *samā*, *sama*, *samānā*, *samāna*, *sadṛśā*, *sadṛśa*, *sadṛkṣā*, *sadṛkṣī*, *sadṛkṣa*, *vilakṣaṇā*, *vilakṣaṇa*, *tulyā*, *tulya*.

4.2.3 Locus Showing the Domain (viṣayādhikaraṇam)

The upapadas such as *lagna, āsakta, anurakta*, etc. mark a relation of *viṣayādhikaraṇam* with the accompanying words in locative case suffix.

12. **Skt:** ***saḥ paṭhane lagnaḥ asti***

    Gloss: He {nom.}, study {loc.}, immerse {nom}, be

    Eng: He is immersed in studies

Here the word *paṭhana* becomes the locus showing the domain accompanied by the upapada *lagna*. See Figure 4.4

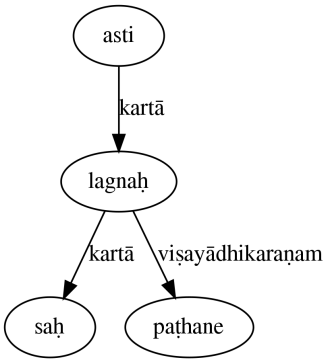

Figure 4.4: Semantic analysis of the sentence (12)

A list of upapada having locus showing the domain as follows—*āsaktā*, *āsakta*, *āyuktā*, *āyukta*, *anuraktā*, *anurakta*, *asādhu*, *kuśalā*, *kuśala*, *lagnā*, *lagna*, *nipuṇā*, *nipuṇa*, *prasūtaḥ*, *prasitā*, *prasita*, *prasitaḥ*, *sādhu*, *utsukā*, *utsuka*.

4.2.4 Determination (nirdhāraṇam)

In the sentence

13. **Skt**: ***gavāṃ prasūtā asti***
    Gloss: go {loc.}, born {nom.}, be.
    Eng: delivered among the cows.

the upapada *prasūta* determines the word *go*. See Figure 4.5.

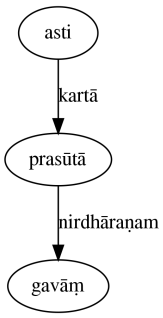

Figure 4.5: Semantic analysis of the sentence (13)

Upapadas *prasūtā*, and *prasūta* are the other upapadas having the determination role in a sentence.

4.2.5 Purpose (prayojanam)

In a sentence like

14. **Skt**: ***Rāmāya kuśalaṃ bhūyāt***
    Gloss: Rāma {nom.}, good {nom.}, be.
    Eng: Let good be to Rāma.

the upapada *kuśala* expresses a role of purpose to Rāma. See Figure 4.6.

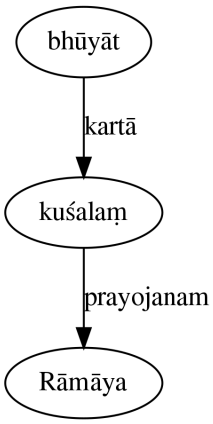

Figure 4.6: Semantic analysis of the sentence (14)

Similar upapadas having role of purpose as follows—*āyuṣyam*, *bhadram*, *śam*, *arthe*, *cirañjīvitam*, *hitam*, *kṛte*, *kuśalam*, *madram*, *nirāmayam*, *sukham*, *svāhā*, *svadhā*, *svasti*, *vaṣaṭ*,

4.2.6 Exclamatory (udgāravācakaḥ)

15. **Skt**: ***durjanam dhik bhavatu***

    Gloss: evil person {nom.}, damn {nom.}, be.

    Eng: Damn to evil persons.

Here the word *dhik* is used to defame the evil person and act as exclamatory relation to the word *durjana*. See Figure 4.7.

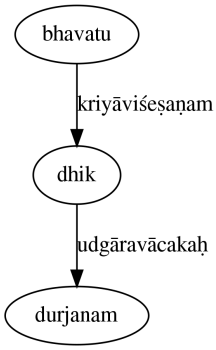

Figure 4.7: Semantic analysis of the sentence (15)

Upapadas *dhik*, and *hā* having the capacity to express the meaning exclamation and defamation in a sentence.

4.2.7 Predicative Adjective (kartṛsamānādhikaraṇam)

16. Skt: ***mallaḥ mallāya alaṃ bhavati***

    Gloss: Wrestler {nom.}, wrestler {inst.}, sufficient{nom.}, be.

    Eng: Wrestler is sufficient (to fight) with the wrestler.

Here the upapada *alam* is used to denote the meaning sufficiency with the wrestler. (See Figure 4.8).

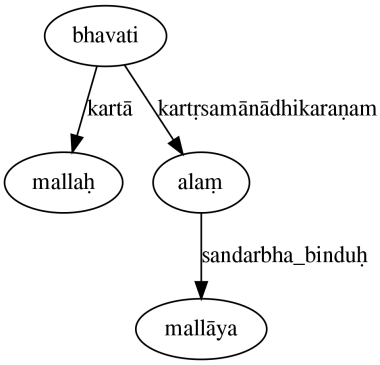

Figure 4.8: Semantic analysis of the sentence (16)

The upapada *alam* has the capacity to express the relation, predicative adjective in a sentence.

4.2.8 Association (saha-arthaḥ)

In all the examples that we saw above, the upapadas governed a word and also were governed by some other word in a sentence. For example, in Figure 4.1, the word *paritaḥ* governs *grāmam* and is governed by the verb *santi*. Now let us consider the following sentence

17. Skt: ***Sītā rāmeṇa saha vanaṁ gacchati***

    Gloss: Sītā {nom.}, Rāma {inst.}, with {nom}, forest{accu.}, go

    Eng: Sītā goes to the forest with Rāma.

In this sentence, *rāma* receives an instrumental case marker due to the presence of the upapada *saha*. This was analysed, as per the earlier tagging guidelines, as shown in Figure 4.9.

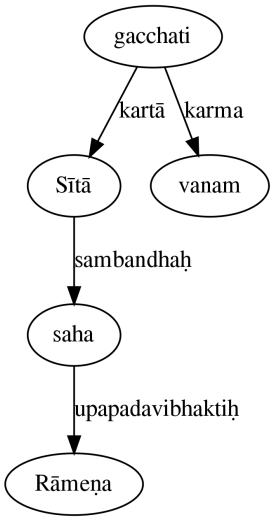

Figure 4.9: Morpho-syntactic analysis of the sentence (17)

In this figure, the relation between *saha* and *Rāma* is marked as upapadavibhaktiḥ and that between *sītā* and *saha* as just a *sambandhaḥ*. While the term *upapadavibhaktiḥ* is grammar specific, the term sambandhaḥ is non-commital to what kind of relation it is. Such an analysis, from user's point of view is of no use. The user can not get any kind of information as to what exactly the relation between *Sītā* and *Rāma* is. Similarly from the point of view of generation, one can not expect from a user to provide the name of the relation as upapadavibhaktiḥ, for the reasons stated earlier, and the term sambandhaḥ does not carry any information about the nature of relation between the relata. Let us look at the semantics involved here.

In this sentence the agreement of the verb is with *Sītā*, and not with *Rāma*. According to the sūtra *sahayukte 'pradhāne*(2.3.19), "*saha* is used with the apradhāna (sub-ordinate)-kāraka". Thus in this example, *Sītā* is the kartā of the action associated with the verb *gacchati*. And *Rāma* is the saha-kartā(associative agent). The upapada *saha* does not have any semantic role, unlike the upapada *paritaḥ* in the above example. We propose the dependency tree for sentence (17) as shown in Figure 4.9.

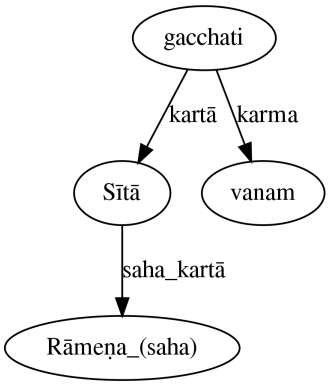

Figure 4.10: Semantic analysis of the sentence (17)

Other words following this semantics are *sākam*, *sārdham*, *saha*, and *samam*.

4.2.9 Dis-association (vinā-arthaḥ)

18. Skt: ***Rāmaṃ pṛthak Śyāmaḥ vartate***

    Gloss: Rāma {inst.}, without {nom.}, Śyāma {nom.}, be.

    Eng: Rāma stays without Śyāma.

Here the upapada *pṛthak* express a dis-association relation between Rāma and Śyāma. See Figure 4.11.

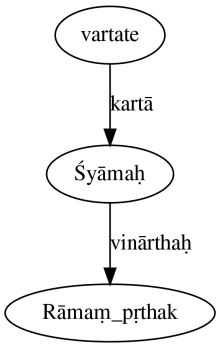

Figure 4.11: Semantic analysis of the sentence (18)

The set of words *antarā*, *antareṇa*, *nānā*, *pṛthak*, *vinā*, exhibit similar semantics.

4.2.10 Possessor-possessee (svāmī)

19. **Skt**: ***Sītā gavāṃ svāminī asti***

    Gloss: Sītā{nom.}, cow{geni.}, owner{nom.}, be.

    Eng: Sītā is the owner of the cows.

Here the upapada shows a sva-svāmibhāvasambandha . See Figure 4.12.

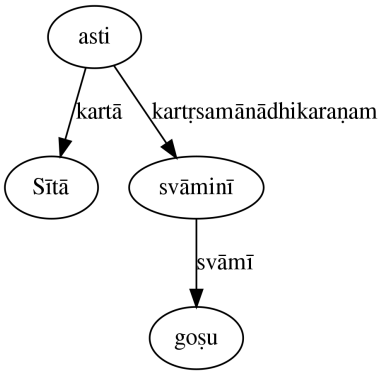

Figure 4.12: Semantic analysis of the sentence (19)

A list of upapada shows possessor-possessee semantics as follows—*īśvara*, *adhipatiḥ*, *pratibhū*, *sākṣiṇī*, *sākṣiṇaṃ*, *sākṣin*, *svāmin*, *svāminī*, *dāyāda*

4.2.11 Source (srotaḥ)

Upapadas *ārabhya*, and *prabhṛti* show the semantic of source.

Other Relations Upapadas shows different relations exception to the above mentioned relations are *alam*, *namaḥ*, *ṛte*, prāk, and arthaḥ.

Conclusion

In this chapter we have discussed the case of upapadas. We noticed that the Pāṇinian treatment of upapadas is purely morpho-syntactic and hence is not useful from both the analysis as well as generation point of view. The morpho-syntactic is a phenomenon internal to language, in natural language processing, machine translation,

information retrieval or question answering systems and so on, we need semantic representation. We propose the semantic analysis of the relations due to *upapadas*. We noticed that these relations semantically can be classified into classes. Such a semantic analysis is useful for both analysis as well as generation. In the Appendix .3 we have provided a morphosyntactic classification of upapadas. This is required for assigning the correct case marker to the words which these upapadas accompany. Appendix .4 provides a semantic classification of the upapadas.

Chapter 5

Testing and Evaluation

Every piece of software needs proper testing and evaluation before it goes to real use. The NLP industry is dominated by texts for analysis. And the analysers need to be evaluated for their precision and recall. During analysis, one comes across several ambiguities which result into low precision and sometimes low recall owning to the incomplete lexicon. When it comes to NLG, the situation is different. The generative systems are tested for their completeness and correctness. One needs to ensure that what the generator generates is correct from a grammatical point of view. Similarly one needs to ensure that it has the capability to generate all types of sentences the language allows.

5.1 Test Bed

A list of around 1000 sentences is manually collected covering the wide range of syntactic phenomenon and also verbs with different expectancies. These sentences were manually parsed/providing the input in the prescribed format to the generator. The output of the sentence generator is compared with the given sentence. A few sample inputs and their outputs are shown below.

Sample Input and Output

kartā

- **input**

  1 rāma puṃ eka kartā 2

  2 pac1 kartari vartamānaḥ

  output: rāmaḥ pacati.

- **input**

  1 rāma puṃ eka kartā 2

  2 gam1 karmaṇi vartamānaḥ

  output: rāmeṇa gamyate.

- **input**

  1 rāma puṃ eka kartā 2

  2 gam1 bhāvaḥ napuṃ eka kartā 3

  3 bhū1 kartari vartamānaḥ

  output: rāmasya gamanaṃ bhavati.

prayojakakartā

- **input**

  1 devadatta puṃ eka prayojakakartā 4

  2 viśvāmitra puṃ eka prayojyakartā 4

  3 odana puṃ eka karma 4

  4 pac1 ṇic kartari vartamānaḥ

  output: devadattaḥ viśvāmitreṇa odanaṃ pācayati.

prayojyakartā

- **input**

  1 devadatta puṃ eka prayojakakartā 3

  2 viśvāmitra puṃ eka prayojyakartā 3

  3 pac1 ṇic kartari vartamānaḥ

  output: devadattaḥ viśvāmitreṇa pācayati.

- **input**

  1 mātṛ strī eka prayojakakartā 4

  2 bāla puṃ eka karma 4

  3 kṣīra puṃ eka prayojyakartā 4

  4 pā1 ṇic kartari vartamānaḥ

  output: mātā bālaṃ kṣīraṃ pāyayati.

- **input**

  1 mātṛ strī eka prayojakakartā 5

  2 dhātrī strī eka karaṇam 5

  3 bāla puṃ eka karma 5

  4 dugdha napuṃ eka prayojyakartā 5

  5 pā1 ṇic kartari vartamānaḥ

  output: mātā dhātryā bālaṃ dugdhaṃ pāyayati.

- **input**

  1 rāma puṃ eka kartā 3

  2 grāma puṃ eka karma 3

  3 gam1 karmaṇi vartamānaḥ

  output: rāmeṇa grāmaḥ gamyate.

karma

- **input**

  output: 1 śatru puṃ bahu mukhyakarma 2

  2 ji1 kartari vartamānaḥ **output**: śatrūn jayati.

- **input**

  1 bālaka puṃ eka kartā 3

  2 paṭh1 karma 3

  3 iṣ2 kartari vartamānaḥ

  output: bālakaḥ paṭhitum icchati.

mukhyakarma

- **input**

  1 gopāla puṃ eka kartā 4

  2 go puṃ eka mukhyakarma 4

  3 dugdha napuṃ eka gauṇakarma 4

  4 duh2 kartari vartamānaḥ

  output: gopālaḥ gāṃ dugdhaṃ dogdhi.

gauṇakarma

- **input**

  1 gopāla puṃ eka kartā 4

  2 go puṃ eka gauṇakarma 4

  3 dugdha napuṃ eka mukhyakarma 4

  4 duh2 karmaṇi vartamānaḥ

  output: gopālena gauḥ dugdham duhyate.

karaṇam

- **input**

  1 bāla puṃ eka kartā 4

  2 hasta puṃ eka karaṇam 4

  3 anna napuṃ eka karma 4

  4 khād1 kartari vartamānaḥ

  output: bālaḥ hastena annaṃ khādati.

sampradānam

- **input**

  1 devadatta puṃ eka kartā 4

  2 brāhmaṇa puṃ eka sampradānam 4

  3 go puṃ eka karma 4

  4 dā3 kartari vartamānaḥ

  output: devadattaḥ brāhmaṇāya gāṃ dadāti.

- **input**

  1 upādhyāya puṃ eka kartā 4

  2 śiṣya puṃ eka sampradānam 4

  3 capeṭā strī eka karma 4

  4 dā3 kartari vartamānaḥ

  output: upādhyāyaḥ śiṣyāya capeṭāṃ dadāti.

adhikaraṇam

- **input**

  1 vānara puṃ eka kartā 3

  2 vṛkṣa puṃ eka adhikaraṇam 3

  3 vas1 kartari vartamānaḥ

  output: vānaraḥ vṛkṣe vasati.

pūrvakālaḥ

- **input**

  1 rāma puṃ eka kartā 5

  2 dugdha napuṃ eka karma 3

  3 pā1 pūrvakālaḥ 5

  4 śālā strī eka karma 5

  5 gam1 kartari vartamānaḥ

  output: rāmaḥ dugdhaṃ pītvā śālāṃ gacchati.

vartamānasamānakālaḥ

- **input**

  1 bālaka puṃ eka kartā 4

  2 jala napuṃ eka karma 3

  3 pā1 vartamānasamānakālaḥ puṃ eka vartamānasamānakālaḥ 4

  4 gam1 kartari vartamānaḥ

  output: bālakaḥ jalaṃ piban gacchati.

sambodhyaḥ

- **input**

  1 bho sambodhanasūcakam 2

  2 rāma puṃ eka sambodhyaḥ 4

3 asmad sarva eka karma 4

4 ut+dhṛ1 kartari ājñāprārthanādiṣu

output: bho rāma mām uddhara.

hetuḥ

- **input**

  1 vidyārthin puṃ eka kartā 4

  2 adhi+i2 bhāvaḥ napuṃ eka hetuḥ 4

  3 vidyālaya puṃ eka adhikaraṇam 4

  4 vas1 kartari vartamānaḥ

  output: vidyārthī adhyayanena vidyālaye vasati.

prayojanam

- **input**

  1 asmad sarva eka kartā 5

  2 yoga-śāstra puṃ eka karma 3

  3 paṭh1 prayojanam 5

  4 vidyālaya puṃ eka karma 5

  5 gam1 kartari vartamānaḥ

  output: ahaṃ yoga-śāstraṃ paṭhitum vidyālayaṃ gacchāmi.

- **input**

  1 asmad sarva eka kartā 6

  2 bhavat sarva puṃ eka karma 5

  3 asmad sarva eka ṣaṣṭhīsambandhaḥ 4

  4 gṛha napuṃ eka adhikaraṇam 5

  5 dṛś1 karma 6

  6 iṣ2 kartari vartamānaḥ

  output: ahaṃ bhavantaṃ mama gṛhe draṣṭuṃ icchāmi.

kartṛsamānādhikaraṇam

- **input**

  1 rāma puṃ eka kartā 3

  2 śūra puṃ eka kartṛsamānādhikaraṇam 3

  3 as2 kartari vartamānaḥ

  output: rāmaḥ śūraḥ asti.

karmasamānādhikaraṇam

- **input**

  1 asmad sarva eka kartā 4

  2 yuṣmad sarva eka karma 4

  3 paṇḍita puṃ eka karmasamānādhikaraṇam 4

  4 man1 kartari vartamānaḥ

  output: ahaṃ tvāṃ paṇḍitaṃ manve.

pratiṣedhaḥ

- **input**

  1 ghaṭa puṃ eka kartā 3

  2 na pratiṣedhaḥ 3

  3 as2 kartari vartamānaḥ

  output: ghaṭaḥ na asti.

ṣaṣṭhīsambandhaḥ

- **input**

  1 adhyāpaka puṃ eka ṣaṣṭhīsambandhaḥ 2

  2 pustaka napuṃ eka karma 4

  3 chātra puṃ bahu kartā 4

  4 paṭh1 kartari vartamānaḥ

  output: adhyāpakasya pustakaṃ chātrāḥ paṭhanti.

viśeṣaṇam

- **input**

  1 dāśarathi viśeṣaṇam 2

  2 rāma puṃ eka kartā 4

  3 vana napuṃ eka karma 4

  4 gam1 kartari vartamānaḥ

  output: dāśarathiḥ rāmaḥ vanaṃ gacchati.

sambodhanasūcakam

- **input**

  1 bho sambodhanasūcakam 2

  2 rāma puṃ eka sambodhyaḥ 4

  3 asmad sarva eka karma 4

  4 ut+dhṛ1 kartari ājñāprārthanādiṣu

  output: bho rāma mām uddhara.

abhedaḥ

- **input**

  1 daśaratha napuṃ eka ṣaṣṭhīsambandhaḥ 3

  2 putra abhedaḥ 3

  3 rāma puṃ eka kartā 4

  4 as2 kartari vartamānaḥ

  output: daśarathasya putraḥ rāmaḥ asti.

atyantasaṃyogaḥ

- **input**

  1 rāma puṃ eka kartā 3

  2 māsa puṃ eka karma 3

3 adhi+i2 kartari vartamānaḥ

output: rāmaḥ māsaṃ adhīte.

sandarbhabinduḥ

- **input**

  1 grāma puṃ eka sandarbha_binduḥ 2

  2 paritaḥ deśādhikaraṇam 4

  3 vṛkṣa puṃ bahu kartā 4

  4 as2 kartari vartamānaḥ

  output: grāmaṃ paritaḥ vṛkṣāḥ santi.

tulanābinduḥ

- **input**

  1 śyāma puṃ eka tulanā_binduḥ 2

  2 tulya puṃ eka kartā 4

  3 rāma puṃ eka kartṛsamānādhikaraṇam 4

  4 as2 kartari vartamānaḥ

  output: śyāmena tulyaḥ rāmaḥ asti.

viṣayādhikaraṇam

- **input**

  1 tad sarva puṃ eka kartā 3

  2 paṭh1 bhāvaḥ napuṃ eka viṣayādhikaraṇam 3

  3 lasj1 bhūtakartā puṃ eka kartā 4

  4 as2 kartari vartamānaḥ

  output: saḥ paṭhane lagnaḥ asti .

prayojanam

- **input**

  1 rāma puṃ eka prayojanam 2

  2 kuśala napuṃ eka kartā 3

  3 bhū1 kartari āśīḥ

  output: rāmāya kuśalaṃ bhūyāt .

saha-arthaḥ

- **input**

  1 rāma puṃ eka sahārthaḥ 3

  3 sītā strī eka kartā 5

  4 vana napuṃ eka karma 5

  5 gam1 kartari vartamānaḥ

  output: rāmeṇa saha sītā vanaṃ gacchati .

5.2 Relation Labels' Suitability

The relation labels used for the generation are the same as those used for the analysis. In order to ensure that the relation labels are both necessary and sufficient for both these inverse processes of analysis and generation, each sentence is parsed with the available parser and the parsed output, which is the same as the meaning representation or the semantic input for the generation is manually verified. This semantic representation is given to the generator as an input. Again the output from the sentence generator is verified manually[1] with the sentence given to the parser. Thus the evaluation takes place in two places namely in the parser and generator. Hence it is a mutual feedback mechanism.[2] The term mutual feedback mechanism is inspired by the term 'Dual learning mechanism' [He et al. (2016)]. The fact that any machine transla-

[1] Here also human assistance becomes necessary to verify or to select correct input from the multiple analysis from the parser.

[2] The present research was carried out at the same time with the parser development, which helped a lot to evaluate the adequacy of information given as input to the sentence generator.

tion task has a dual task,e.g., here, Sanskrit Sentence analysis and Sanskrit Sentence Generator forms the basis of Dual Learning Mechanism.

Does the analysis produced by the parser carry sufficient information for the generator to generate the desired sentence?. As noted earlier, it makes sense to see natural language generation and natural language understanding as the two halves of the puzzle of natural language processing. The dual learning or mutual feedback mechanism or infusing NLG and NLU as system as a whole helps in the improvement of both the systems. See figures 5.2. This helps to improve the efficiency of the the generator and it also helps the performance of the parser.

5.3 Problems of Evaluation

The main problem in testing the accuracy of the generated sentence is that the generated sentence need not be unique or there is no 'the right' answer. The multiple answers may be due to variations in spelling such as *pītvā*, *pīttvā* or variations in form such as *mama,me* or even choice of lexical items. Owing to the free order in Sanskrit, the generated sentence need not have the same word order as that of the original analysed sentence. Hence we used the mutual feedback mechanism for testing and evaluation.

There were a few challenges in the evaluation. In the absence of a taddhita (secondary derivatives) word generator, we provide the nominal stem formed by affixing the taddhita suffix. For example, we directly provide the stem śaktimat instead of śakti + matup. Similarly, in the absence of a handler for feminine suffix, we provide the stem formed after the addition of feminine suffix as in anarthā (which is formed by adding a feminine suffix to anartha). In order to handle the out of vocabulary words, we developed a morphological analyser that assigns the default paradigm for the generation of such words.

5.4 Sanskrit Sentence Generator: Interface

For both the evaluations, an interface is used to verify the input. The Graphical User Interface (GUI) of the Sanskrit Sentence Generator facilitates a user to provide the

required input in a prescribed form. Figure 5.1 shows the generator interface for the following input.

word index	stem and features	relation
1	dṛś1	pūrvakālaḥ 11
2	tu	sambandhaḥ 1
3	pāṇḍava-ānīka {puṃ eka}	karma 1
4	vi+vah1 {bhūtakarma}	viśeṣaṇam 3
5	duryodhana {puṃ eka}	kartā 11
6	tadā	kālādhikaraṇam 11
7	ācārya {puṃ eka}	karma 8
8	upa_sam+gam1	pūrvakālaḥ 11
9	rājan	abhedaḥ 5
10	vacana {napuṃ eka}	karma 11
11	brū1 {anadyatanabhūtaḥ}	kartari

Table 5.1: Input for the generator

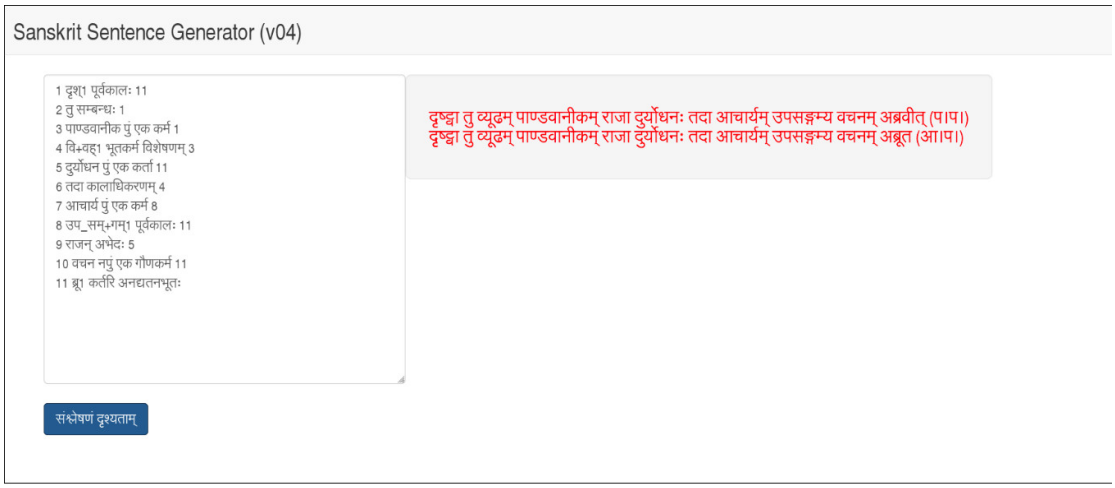

Figure 5.1: Generation of Bhagavadgīā 1.2 śloka from its analysis

We have also provided another interface. This interface takes the input from the Sanskrit parser. It allows us to test the completeness of both parser as well as the generator at the sentence level. This interface takes the machine internal representation of the parser's output (which is the same as shown in the Table 2.1) and feeds it to the generator. The overall architecture of our generator (and parser) is as shown in Figure 5.2.

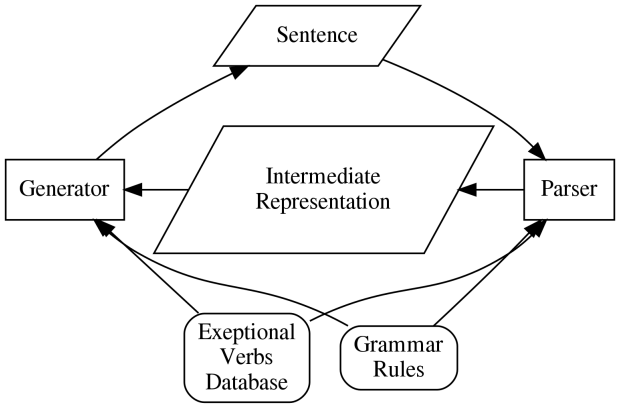

Figure 5.2: Parser-generator: inverse operations

Conclusion

Successive sentence generation depends on evaluation of 'what to say' (see input, section 2.3.1) and evaluation of 'how to say' (see kāraka to vibhakti mapping chapter 3). Problems we faced will be useful in developing an interface in future. It will help to know 'what do people know about their language, what processes do they employ that enables them to be'.

In the present version of Sanskrit Sentence Generator, compound words to be manually splitted, and unsandhied words to be given as input are its limitations. Hence the user should have a basic knowledge of grammar to handle the input. Most of the relation labels are semantic in nature, one may need some initial training for the proper use of some relational tags. In specifying the use of conjuncts and disjuncts since the current implementation is dominated by the syntax of Sanskrit more research is needed to arrive at a uniform treatment of the conjuncts across languages [Panchal and Kulkarni]. Other disadvantage of this generator is the amount of information one has to provide to the generator in a particular format. Finally the sentence generator completely depends on the efficiency of the morphological generator and it is to be improved.

Designing an user interface that hides the language and grammar specific details from the user and allows him to provide the input purely in semantic form are to be incorporated. Question answer or selection mode, check for input constistency,

constrain rules and global checking are the other features of interface to be added. Sentences covering different relations mentioned in the tagset to be implemented.

Chapter 6

Conclusion

Pāṇini' grammar provides a grammatical framework for generation. While the complexity of Sanskrit generation lies at the word level, the sentence generation is pretty straightforward. Only challenge in designing the generator was in deciding the granularity of the semantic relations appropriate for both analysis and generation. We wanted to make sure that the grammatical relations used are universal in nature, without carrying any baggage of the language idiosynchrosy. Having confirmed that the tagset is appropriate for both generation and analysis [Kulkarni (2019)], we can now open it for other languages as well; to start with, the Indian languages.

6.1 Utility

SSG will be useful for the following applications—

As a module for MT involving Sanskrit as target language: One language to another Language translation helps not only in communicaton, to enrich the language itself. There are tools available for languages other than Sanskrit to have a translation from a source language to a target language and vice versa. Already available fulfledged grammar like Aṣṭādhyāyī, Sanskrit requires a Sentence Generator, which can be extended to be a model for other Indian languages to overcome the language barriers.

Voice-converter: With a single keystroke, one can generate passive constructs which are predominantly found in Sanskrit literature, with which a non-native speaker may not be at ease with.

Mutual Improvement of Sanskrit Parser and Generator: Analysis and generation of a sentence is mutual, hence the functioning of the parser and the generator is also

mutual. Analysis from the parser taken as an input to the generator, produces the same sentence again which is analysed by the parser is an advantage.

As a module in Question-Answering and information Retrieval systems.

As an assistance for writing/composing Sanskrit texts: It will act as an

- Useful aid to the non-native speakers of Sanskrit to write in Sanskrit effectively guranteeing grammatically correct sentences
- One need not memorize the word forms and the gender of the nominal stems
- No need to remember all the special rules assigning case suffix to a noun representing specific kāraka role.
- The generator does not dictate any word order. So one may generate a sentence in any word order as one desires. In future it should also be possible to provide a generator that will help the user to render the text in a chosen prosodic meter.

The major contribution of the development of this module was in identifying some morpho-syntactic relation labels such as those due to upapadas [Kulkarni (2019)].

Sanskrit Sentence Generator is not domain specific. The generator can take up the intermediate representation of the QA system and generate valid Sanskrit sentences which can be used to communicate with the user. Similarly information retrieval system may extract the information from the database/corpus which might be in machine understandable form but not readable/understandable by the user. This generator can then be used to present the information in plain Sanskrit sentences that are understandable by the users.

6.2 Limitations and Future work

Following special constructs in Sanskrit are not yet implemented.

- Sentences involving karmapravacanīyas
- Compound and complex sentences involving conjuncts, anaphora, multiple clauses, etc.

- Not handled śeṣasambandha which include the relation denoting svasvāmibhāva, avayava-*avayavī* etc.,
- Present morphological generator which we are using to generate a sentence does not have the capacity to deal with femine suffixes (strīpratyayāḥ).

Bibliography

Abhyankar, K. V. (1961). *A Dictionary of Sanskrit Grammar*. Oriental Institute Baroda.

Apte, V. S. (1917). *The Students Guide to Sanskrit Composition*. The Oriental Publishing Company, Bombay.

Becker, T. (2006). *Natural Language Generation with Fully Specified Templates*, pages 401–410. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bharati, A., Chaitanya, V., and Sangal, R. (1994). *Natuaral Language Perscpective - A Pāṇinian Perspective*. Prentice Hall of India.

Bharati, A. and Kulkarni, A. (2011). 'Subject' in English is abhihita. In Cardona, G., Aklujkar, A., and Ogawa, H., editors, *Studies in Sanskrit Grammars*. D.K. Printworld, New Delhi.

Cardona, G. (2007). *On the Structure of Pāṇini's System*, volume 1&2, pages 1–31. Springer.

Cawsey, A. J., Webber, B. L., and Jones, R. B. (1997). Natural Language Generation in Health Care. *Journal of the American Medical Informatics Association*, 4(6):473–482.

Dale, Robert; Reiter, E. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, U.K.

De Smedt, K., Horacek, H., and Zock, M. (1996). Architectures for natural language generation: Problems and perspectives. In Adorni, G. and Zock, M., editors, *Trends in Natural Language Generation An Artificial Intelligence Perspective*, pages 17–46, Berlin, Heidelberg. Springer Berlin Heidelberg.

Galitsky, B. (2013). A Web Mining Tool for Assistance with Creative Writing.

Goyal, P. and Kulkarni, A. (2014). Converting phrase structures to dependency structures in sanskrit. In *Proceedings of COLING 2014, the 25th International Conference*

on Computational Linguistics: Technical Papers, pages 1834–1843, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Goyal, P., Kulkarni, A., and Behera, L. (2009). *Computer Simulation of* Aṣṭādhyāyī *: Some insights.* Springer Verlag LNCS 5402.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016). Dual learning for machine translation. *30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.*

Houben, J. E. (2009). Pāṇini's grammar and its computerization: A construction grammar approach. *Sanskrit Computational Linguistic*, 3:6–21.

Hämäläinen, M. (2018). Poem machine - a co-creative nlg web application for poem writing. In *Proceedings of The 11th International Natural Language Generation Conference*, pages 195–196, Noida, India. Association for Computational Linguistics.

Iwama, K. and Kano, Y. (2018). Japanese advertising slogan generator using case frame and word vector. In *Proceedings of The 11th International Natural Language Generation Conference, Japan*, pages 197–198, Japan. Association for Computational Linguistics.

Jong, R. D. and Theune, M. (2018). Going dutch: Creating simple-nlg-nl. *The 11th International Natural Language Generation Conference,Tilburg, The Netherlands,.*

Joshi, S. D. (2009). Background of the *Aṣṭādhyāyī*. *Sanskrit Computational Linguistic*, 3:1–5.

Kiparsky, P. (2009). *On the Architecture of Pāṇini's Grammar*, volume 1&2, pages 32–94. Springer, Berlin, Heidelberg.

Krishna, A. and Goyal, P. (2016). Towards automating the generation of derivative nouns in sanskrit by simulating. *16th World Sanskrit Conference, Bangkok June 28th - July 02 2016.*

Kuchibbotla, L. R. (January 1995). *Natural Language Generation: Illustration of Word Usage.* PhD thesis, Department of Computer/Information Sciences, University of Hyderabad.

Kulkarni, A. (2019). Appropriate Dependency Tagset for Sanskrit Analysis and Generation. In *Proceedings of Sanskrit in China International Conference 2019: Sanskrit on Paths*. forthcoming.

Kulkarni, A. and (eds), G. H. (2009). *Sanskrit Computational Linguistics symposium 3*. Springer Verlag LNCS 5406, Berlin, Heidelberg.

Kulkarni, A. and Pai, M. (2019). Sanskrit Sentence Generator. International Conference on Churning of Indology, Bharatiya Vidvat Parishad & Tattvasamshodhana Samsat, Udupi, 4 to 6, January, 2019.

Kulkarni, A. and Shukl, D. (2009). Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

Kulkarni, A., Shukla, P., Shukl, D., and Satuluri, P. (2016). How 'free' is Free Word Order in Sanskrit. In Scharf, P., editor, *Sanskrit Syntax*. Sanskrit Library.

Madhavan, M. and Reghuraj, P. (2011). Application of Karaka Relations in Natural Language Generation. *Govt. Engg. College Sreekrishnapuram Palakkad, India-678633*.

Manning, C. D. and Marneffe, M.-C. D. (2016). Stanford typed dependencies manual. *Stanford typed dependencies manual*.

Matilal, B. K. (1966). *Indian Theorists on the Nature of the Sentence (Vākya)*, volume 2. Springer.

Mellish, R. D. C. (1998). Towards the evaluation of natural language generation. *https://web.science.mq.edu.au/ rdale/publications/papers/1998/lrec98.pdf*.

Mishra, A. (2007). *Simulating the Pāṇinian System of Sanskrit Grammar*, volume 1&2, pages 127–138. Springer.

Mollá-Aliod, D. and Vicedo, J.-L. (2010). *Handbook of Natural Language Processing*. Chapman and Hall Book.

Nivre, J. (2005). Dependency grammar and dependency parsing. Technical report, Växjö University.

O'Donnell, M. (1997). Input specification in the wag sentence generation system. *Association for Computational Linguistics*. https://www.aclweb.org/anthology/W96-0405.

Pan, S. and Shaw, J. (2004). Segue: A hybrid case-based surface natural language generator. *INLG 2004*.

Panchal, S. and Kulkarni, A. Caśabdayuktavākyaviśleṣaṇam. In Mahulikar, G., editor, *Proceedings of NFSI*. Chinmaya Vishvavidyalaya, Veliyanad. forthcoming.

Pande, G. D. (1992). *Aṣṭādhyāyī of Pāṇini*. Chaukhamba Surabharti Prakashan, Varanasi.

Patañjali (2014). *Vyākaraṇamahābhāṣyaṃ*. Chaukhamba Sanskrit Pratishtan, Delhi.

Patel, D. and Katuri, S. (2016). Prakriyāpradarśinī - an open source subanta generator. *16th World Sanskrit Conference, Bangkok June 28th - July 02 2016*. https://www.sanskritworld.in/sanskrittool/SanskritVerb/tiGanta.html.

Potti, P. R. V. (2006). *Kārakaprakaraṇam (Malayālam)*. Sukṛtīndra Oriental Research Institue, Thammanam, Kochi, Kerala - 682035.

Ramakrishanamacharyulu, K. (2009). Annotating Sanskrit Texts Based on Śābdabodha Systems. *Sanskrit Computational Linguistics*.

Ramakrishnamacharyulu, K. V. (2017). Vibhaktyarthavicāraḥ. In *Sārasvatam, A Book of Proceedings of National Seminar, 2016-2017*. Rashtriya Sanskrit Sansthan, Guruvayoor Campus.

Rao, V. S. (1969). *The Philosophy of a Sentence and its Parts*. Munshiram Manorharlal Publishers, New Delhi.

Roodbergen, J. A. F. (2008). *Dictionary of Pāṇinian Grammatical Terminology*. Bhandarkar Oriental Research Institute.

Satuluri, P. (2015). *Sanskrit Compound Generation: With a Focus on the Order of Operations*. PhD thesis, Department of Sanskrit Studies, University of Hyderabad. PhD Thesis.

Satuluri, P. and Kulkarni, A. (2013). Generation of Sanskrit Compounds. In *Proceedings of ICON 2013, the 10th International Conference on NLP*, pages 77–86, Noida, India.

Scharf, P. M. (2007). *Modeling Pāṇinian Grammar*, volume 1&2, pages 95–126. Springer.

Shailaja, N. (2014). *Comparison of Paṇinian Dhātuvṛttis.* PhD thesis, Department of Sanskrit Studies, University of Hyderabad.

Shastri, B. (2000). *Laghusiddhāntakaumudī-Bhaimīvyākhyā.* Bhaimi Prakashan, New Delhi.

Shukla, P., Shukl, D., and Kulkarni, A. (2010). Vibhakti divergence between sanskrit and hindi. In Jha, G. N., editor, *Sanskrit Computational Linguistics*, pages 198–208, Berlin, Heidelberg. Springer.

Sripada, S. G., Reiter, E., and Robertson, R. (2003). Acquiring Correct Knowledge for Natural Language Generation. *AI Access Foundation and Morgan Kaufmann Publishers.*

Subbanna, S. and Varakhedi, S. (2007). *Computational Structure of the Aṣṭādhyāyī and Conflict Resolution Techniques*, volume 1&2, pages 56–65. Springer.

Subrahmanya, K. (*2012*). Mahāvākyārthavicāraḥ. *Śrīlakṣmaṇārya Bhavyasmṛti Granthamālā, Bhāgyanagaram.*

Valls, J. and Ontañón, S. (2012). Natural language generation through case-based text modification. In Agudo, B. D. and Watson, I., editors, *Case-Based Reasoning Research and Development*, pages 443–457, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vamana and Jayaditya (1969). *Kāśikā.* Ramlal Kapur Trast, Sonipat, Hariyana.

Vicente, M., Barros, C., and Lloret, E. (2017). A study on flexibility in natural language generation through a statistical approach to story generation. In Frasincar, F., Ittoo, A., Nguyen, L. M., and Métais, E., editors, *Natural Language Processing and Information Systems*, pages 492–498, Cham. Springer International Publishing.

Appendices

.1 Default Kāraka Vibhakti Mapping

dhātu type	kartā	mukhya-karma	guaṇa-karma	udāharaṇa
akarmaka	1	-		Rāmaḥ tiṣṭhati.
sakarmaka	1	2		Rāmaḥ grāmaṃ gacchati.
dvikarmaka	1	2	2	Rāmaḥ ajāṃ grāmaṃ nayati.

Table 1: Kartari prayogaḥ

dhātu type	kartā	mukhya-karma	guaṇa-karma	udāharaṇa
akarmaka	3	-		Rāmeṇa sthīyate.
sakarmaka	3	1		Rāmeṇa grāmaḥ gamyate.
dvikarmaka-type1	3	2	1	Rāmeṇa gauḥ dugdhaṃ duhyate.
dvikarmaka-type2	3	1	2	Rāmeṇa ajā grāmaṃ nīyate.

Table 2: Karmaṇi prayogaḥ

dhātu type	prayojaka-kartā	prayojya-kartā	karma	udāharaṇa
akarmaka	1	2	-	Rāmaḥ Lakṣmaṇaṃ śāyayati.
sakarmaka-type1	1	2	2	Rāmaḥ Lakṣmaṇaṃ vedaṃ pāṭhayati.
sakarmaka	1	3	2	Rāmaḥ Lakṣmaṇena annaṃ pācayati.
dvikarmaka	1	3	2	Rāmaḥ Lakṣmaṇena ajāṃ grāmaṃ nāyayati.

Table 3: Ṇijantakartari prayogaḥ

dhātu type	prayojaka-kartā	prayojya-kartā	karma	udāharaṇa
akarmaka	3	1	-	Rāmeṇa Lakṣmaṇaḥ śīyate.
sakarmaka-type1.1	3	1	2	Rāmeṇa Lakṣmaṇaḥ grāmaṃ gamyate.
sakarmaka-type1	3	1(2)	2(1)	Rāmeṇa Lakṣmaṇaḥ vedaṃ pāṭhyate. / Rāmeṇa Lakṣmaṇaṃ vedaḥ pāṭhyate.
sakarmaka	3	3	1	Rāmeṇa Lakṣmaṇena odanaṃ pācyate.
dvikarmaka-type1	3	3	1(mukhya) 2(gauṇa)	Rāmeṇa Lakṣmaṇena gauḥ dugdhaṃ duhyate.
dvikarmaka-type2	3	3	1(mukhya) 2(gauṇa	Rāmeṇa Lakṣmaṇena ajā grāmaṃ nīyate.

Table 4: Ṇijantakarmaṇi prayogaḥ

dvikarmaka-type1 = duh, yāc, etc.

dvikarmaka-type2 = nī, hṛ, kṛṣ, and vah

sakarmaka-type1.1 = only gatyarthaka dhātus.

sakarmaka-type1 = gati, buddhi, pratyavasānārtha, and śabdakarma dhātus.

.2 Tagset of Dependency Relations

- **Kāraka-sambandhāḥ**
- kartā
  - prayojaka-kartā
  - prayojya-kartā
- karma
  - mukhya-karma
  - gauṇa-karma
  - vākya-karma
- karaṇam
- sampradānam
- apādānam
- adhikaraṇam
  - kāla-adhikaraṇam
  - deśa-adhikaraṇam
  - viṣaya-adhikaraṇam
- **Kāraketara-sambandhāḥ**
  - **Kriyā-kriyā-sambandhāḥ**
    - pūrva-kālaḥ
    - vartamāna-samāna-kālaḥ
    - bhaviṣyat-samāna-kālaḥ
    - bhāvalakṣaṇa-pūrva-kālaḥ
    - bhāvalakṣaṇa-vartamāna-samāna-kālaḥ
    - bhāvalakṣaṇa-anantara-kālaḥ
    - sahāyaka-kriyā
  - **Kriyā-sambandhāḥ**
    - sambodhyaḥ
    - hetuḥ
    - prayojanam
    - kartṛ-samānādhikaraṇam
    - karma-samānādhikaraṇam
    - kriyāviśeṣaṇam
    - pratiṣedhaḥ
  - **Nāma-nāma-sambandhāḥ**
    - śaṣṭhī-sambandhaḥ
    - aṅgavikāraḥ
    - vīpsā
    - viśeṣaṇam
    - sambodhana-sūcakam
    - vibhaktam
    - avadhiḥ
    - abhedaḥ
    - lyapkarmādhikaranam
    - nirdhāraṇam
    - atyanta-saṃyogaḥ
    - apavarga-sambandhaḥ
    - vakyakarmadyotakaḥ
- **Upapada-sambandhāḥ**
  - sandarbhabinduḥ
  - tulanābinduḥ
  - viśayādhikaraṇam
  - nirdhāraṇam
  - prayojanam
  - udgāravācakaḥ
  - saha-arthaḥ
  - vinā-arthaḥ
  - svāmī
  - srotaḥ
- **Vākyetarasambandhāḥ**
  - anuyogī
  - pratiyogī
  - nitya-sambandhaḥ
- **Samuccayādisambandhāḥ**
  - samuccitaḥ
  - samuccaya-dyotakaḥ
  - anyataraḥ
  - anyatara-dyotakaḥ

Note: The bold entries are the headings and do not indicate relation labels

.3 Morpho-syntactic Classification of Upapadas

Complete list of all the *upapadas* having an expectancy of *dvitīyāvibhakti*, *tṛtīyāvibhakti* etc. are listed below. (In the entire list, we can see words with suffix and without suffix. Here the term without case marker (prātipadika) like *utsuka* shows its variation in all the forms in pulliṅga and napuṃsakaliṅga, and in all the number (vacana). To cover the strīliṅga seperate entry like *utsukā* is kept and exact words like *paritaḥ, abhyāśāt* etc., kept as it is. Terms like *adhareṇa* is an indeclinable (subantapratirūpaka-avyaya) constructed with the suffix *enap* is to be differentiated from the subanta-forms.)

Dvitīyāvibhakti

adhareṇa
adhodhaḥ
adho'dhaḥ
adhyadhi
antarā
antareṇa
abhitaḥ
uttareṇa
uparyupari
ubhayataḥ
dakṣiṇena
dhik
nānā
nikaṣā
paritaḥ
pṛthak
yāvat
vinā
samayā
sarvataḥ
hā

Tṛtīyāvibhakti

alam
utsuka
utsukā
tulya
tulyā
nānā
pṛthak
prasita
prasitā
yukta
vinā
sadṛkṣa
sadṛkṣā
sadṛkṣī
sadṛśa
sadṛśā
samam
samā
samāna
samānā
saha
sākam
sārdham

Caturthīvibhakti

arthaḥ
alam
āyuṣyam
kuśalam
cirañjīvitam
namaḥ
nirāmayam
bhadram
madram
vaṣaṭ
śam
sukham
svadhā
svasti
svāhā
hitam

Pañcamīvibhakti

antikam
antikāt
antikān
antike
antikena
anyaḥ
anyā
abhyāśaḥ

abhyāśam
abhyāśā
abhyāśāt
abhyāśān
abhyāśe
abhyāśena
avācī
avāc
ārabhya
ārāt
itara
itarā
uttaram
uttarā
uttarān
uttarāhi
uttare
udak
udac
udīcī
ṛte
dakṣiṇam
dakṣiṇā
dakṣiṇān
dakṣiṇe
dūrā
dūrāt
dūrān
dūre
dūreṇa
nānā
nikaṭam
nikaṭāt
nikaṭān
nikaṭe
nikaṭena
paścimaṃ
paścimā
paścimān
pūrva
pūrvā
pūrvān
pṛthak
pratīcī
pratyak
pratyañca
prabhṛti
prāk
prācī
prāñca
bahiḥ
bhinna
bhinnā
vinā
viprakṛṣṭam
viprakṛṣṭā
viprakṛṣṭāt
viprakṛṣṭān
viprakṛṣṭe
viprakṛṣṭena
vilakṣaṇa
vilakṣaṇā
samīpam
samīpāt
samīpān
samīpe
samīpena

Ṣaṣṭhīvibhakti

agrataḥ
adhaḥ
adharāt
adhareṇa
adhastāt
adhipati
antaḥ
antikam
antikāt
antikān
antike
antikena
abhyāśaḥ
abhyāśam
abhyāśā
abhyāśāt
abhyāśān
abhyāśe
abhyāśena
arthaḥ
arthe
avaḥ
avarataḥ
avarataḥ
avarastāt
avastāt
āyuṣyam
īśvara
uttarataḥ
uttarāt
uttareṇa
upari
upariṣṭāt
kuśalam

kṛte
cirañjīvitam
tulya
tulyā
dakṣiṇataḥ
dakṣiṇāt
dakṣiṇāhi
dakṣiṇena
dāyāda
dūram
dūrā
dūrāt
dūrān
dūre
dūreṇa
nikaṭam
nikaṭāt
nikaṭān
nikaṭe
nikaṭena
nirāmayam
parataḥ
parastāt
paścāt
puraḥ
purataḥ
purastāt
pratibhū
prasūtaḥ
prasūtā
bhadram
madram
vinā
viprakṛṣṭam
viprakṛṣṭā
viprakṛṣṭāt
viprakṛṣṭān
viprakṛṣṭe
viprakṛṣṭena
śam
sadṛkṣa
sadṛkṣā
sadṛkṣī
sadṛśa
sadṛśā
samā
samāna
samānā
samīpam
samīpāt
samīpān
samīpe
samīpena
sākṣiṇaṃ
sākṣiṇī
sākṣin
sukham
svāminī
svāmin
hitam

Saptamīvibhakti

adhipati
anurakta
anuraktā
asādhu
āyukta
āyuktā
āsakta
āsaktā
īśvara
utsuka
utsukā
kuśala
kuśalā
dāyāda
nipuṇa
nipuṇā
pratibhū
prasita
prasitā
prasūta
prasūtā
lagna
lagnā
sākṣiṇī
sākṣin
sādhu
svāminī
svāmin

.4 Semantic Classification of Upapadas

sandarbhabinduḥ

ārāt
abhitaḥ
abhyāśā
abhyāśān
abhyāśāt
abhyāśaḥ
abhyāśam
abhyāśe
abhyāśena
adhaḥ
adharāt
adhareṇa
adhastāt
adhodhaḥ
adho'dhaḥ
adhyadhi
agrataḥ
antaḥ
antikān
antikāt
antikam
antike
antikena
avāc
avācī
avaḥ
avarastāt
avarataḥ
avastāt
bahiḥ
itarā
itara
nikaṣā
nikaṭān
nikaṭāt
nikaṭam
nikaṭe
nikaṭena
pūrvā
pūrvān
pūrva
paścāt
paścimā
paścimān
paścimaṃ
parastāt
parataḥ
paritaḥ
prāñca
prācī
pratīcī
pratyañca
pratyak
puraḥ
purastāt
puratataḥ
samīpān
samīpāt
samīpam
samīpe
samīpena
samayā
sarvataḥ
ubhayataḥ
upari
upariṣṭāt
uparyupari
uttarā
uttarāhi
uttarān
uttarāt
uttaram
uttarataḥ
uttare
uttareṇa
udīcī
udac
udak
viprakṛṣṭā
viprakṛṣṭān
viprakṛṣṭāt
viprakṛṣṭam
viprakṛṣṭe
viprakṛṣṭena
dūrā
dūrān
dūrāt
dūram
dūre
dūreṇa
dakṣiṇā
dakṣiṇāhi
dakṣiṇān
dakṣiṇāt
dakṣiṇam

dakṣiṇataḥ
dakṣiṇe
dakṣiṇena
tulanābinduḥ
bhinnā
bhinna
anyā
anyaḥ
samā
samānā
samāna
sadṛśā
sadṛśa
sadṛkṣā
sadṛkṣī
sadṛkṣa
vilakṣaṇā
vilakṣaṇa
tulyā
tulya
viṣayādhikaraṇam
āsaktā
āsakta
āyuktā
āyukta
anuraktā
anurakta
asādhu
kuśalā
kuśala
lagnā
lagna
nipuṇā
nipuṇa
prasūtaḥ
prasitā
prasita
prasita
sādhu
utsukā
utsuka
yukta
svāmī
īśvara
adhipatiḥ
pratibhū
sākṣiṇī
sākṣiṇaṃ
sākṣin
svāmin
svāminī
dāyāda
prayojanam
āyuṣyam
bhadram
śam
arthe
cirañjīvitam
hitam
kṛte
kuśalam
madram
nirāmayam
sukham
svāhā
svadhā
svasti
vaṣaṭ
sahārthaḥ
sākam
sārdham
saha
samam
vinārthaḥ
antarā
antareṇa
nānā
pṛthak
vinā
nirdhāraṇam
prasūtā
prasūta
udgāravācakaḥ
dhik
hā
kartṛsamānādhikaraṇam
alam
srotaḥ
prabhṛti
yāvat
anyasambandhaḥ0
arthaḥ
anyasambandhaḥ
alam
namaḥ
ārabhya
ṛte
nānā
prabhṛti
prāk

.5 Glossary

(For easy cross reference and their English translation which we have used in this thesis we provide this glossary. See [Abhyankar (1961) and Roodbergen (2008)]. Marked with the symbol '#' denotes the rough translation.

- *adhikāra* - governing rule consisting of a word.
- *akārānta* - ending in *a*
- *akarmakadhātu* - intransitive verbs, without any object.
- *alaukikavigraha* - the technical constituent analysis for purposed of derivation. E.g. rājapuruṣas is analysed as (rājan + ṅas) + (puruṣas +su) + su.
- *anuyogin* - relata
- *avyaya* - indeclinables
- *bhāve prayoga* - impersonal passive voice
- *caturthī vibhakti* - dative case
- *dhātu* - root
- *ṣaṣṭhī vibhakti* - genitive case
- *dvikarmakadhātu* - ditransitive verbs
- *dvitīyā vibhakti* - accusative case
- *guaṇakarma* - secondary karma#
- *kāraka* - The participants of an action are termed kārakas. There are six kārakas, they are kartā, karma, karaṇam, sampradānam, apādānam, and adhikaraṇam.
- *karmaṇi prayoga* - active voice
- *kartari prayoga* - passive voice
- *kriyāviśeṣaṇa* - adjective
- *mukhyakarma* - primary (karma)#
- *mukhyaviśeṣya* - chief qualificand
- *napuṃsakaliṅga* - nueter gender
- *padī* - there are three types of verb forms namely parasmaipadī, ātmanepadī, and ubhayapadī.#
- *pañcamī vibhakti* - ablative case
- *prathamā vibhakti* - nominative case
- *prātipadika* - stem
- *pratyaya* - suffix
- *prayojakakartā* - the agent who instigates/causes (somebody else to act)
- *prayojyakarma* - object in the form of the prompted agent.
- *prayojyakartā* - the prompted/caused agent.
- *pulliṅga* - masculine gender
- *puruṣa* - person, namely prathamapuruṣa (third person), madhyamapuruṣa (second person), and uttamapuruṣa (First person)
- *śābdabodha* - verbal import
- *sakarmakadhātu* - transitive verbs
- *sambodhanavibhakti* - vocative case
- *saptamī vibhakti* - locative case

- *strīliṅga* - femine gender
- *strīpratyaya* - femine suffixes
- *sūtra* - aphorims
- *taddhita* - secondary derivatives
- *tṛtīyā vibhakti* - instrumental case
- *lakāraḥ* - tenses
- *uddeśyavidheyabhāva* - subject and predicative
- *upapada* - A word standing near or accompanying other
- *upapadavibhakti* - the case marker which is used in connection with the upapada.
- *upasarga* - prefix
- *vacana* - number, namely ekavacana(singular), dvivacana (dual), and bahuvacana (plural)
- *vibhaktipratyaya* - case suffix
- *viśeṣaṇa* - that which qualifies, qualification
- *viśeṣaṇaviśeṣyabhāva* - qualifier and qualified#
- *viśeṣya* - that which is to be differentiated.

ಕರ್ನಾಟಕ ಸಂಸ್ಕೃತ ವಿಶ್ವವಿದ್ಯಾಲಯ
ಶ್ರೀ ಚಾಮರಾಜೇಂದ್ರ ಸಂಸ್ಕೃತ ಮಹಾವಿದ್ಯಾಲಯ ಆವರಣ
ಸಂಪಂಗಿರಾಮನಗರ ಬಸ್, ಚಾಮರಾಜಪೇಟೆ, ಬೆಂಗಳೂರು - 560018.

कर्नाटक-संस्कृत-विश्वविद्यालयः
श्री चामराजेन्द्र संस्कृत महाविद्यालयस्य परिसरः
पञ्चमहाकविमार्गः, चामराजपेटे, बेङ्गलूरु - ५६००१८.

04 July 2019

Certificate

To whomsoever it may concern

It is certified that the article entitled "**Semantics of Morpho-syntactic Case Markers in Indian Languages: Sanskrit a Case Study**" by Madhusoodana Pai J, has been **accepted for publication** in the forthcoming issue of our journal ***"Karnatakasamskrita-adhyayanam"***, published by Karnataka Samskrit University, Bangalore.

Dr. Shivani V

Editor, Karnataka Samskritadhyayanam &

Professor and HOD,

Department of Vyakarana, KSU

ದೂರವಾಣಿ: 080 - 26701303 / 26705387; ಇ-ಮೇಲ್: karnatakasanskrituniversity@gmail.com
दूरवाणी: ०८०-२६७०१३०३; २६७०५३८७; ई-संज्ञेत: - karnatakasanskrituniversity@gmail.com

Gmail **Madhu soodan <jmadhusoodan@gmail.com>**

6th ISCLS notification for paper 4

2 messages

6th ISCLS <6thiscls@easychair.org> Mon, Jul 22, 2019 at 10:28 AM
To: Madhusoodana Pai <jmadhusoodan@gmail.com>

Dear Madhusoodana,

We are happy to inform you that based on the reviews and programme committee evaluation, your submission Sanskrit Sentence Generator is accepted for oral presentation in the Sixth International Sanskrit Computational Linguistics Symposium to be held at IIT Kharagpur, WB, India from October 23-25, 2019.

The comments from the reviewers are enclosed. The program committee has really worked hard to provide comments and suggestions to improve your submission, and we request you to make appropriate changes in your paper addressing the comments.

Bhandarkar Oriental Research Institute (BORI) has accepted the responsibility of publishing the proceedings and make it available at the time of the Symposium.

We request you to submit the final version of your manuscript over Easychair, using the style files available at URL https://iscls.github.io/subm.html. Please note that the submission deadline for this camera ready version is August 10th.

Your final submission should include the following:

1. The final pdf document
2. The source document (TeX/LaTeX/LateX2e) plus all relevant macro libraries, font descriptions, bib files and figures necessary to recreate the pdf
3. The copyright form, filled, signed, and scanned. [Please wait for further instructions regarding the copyright form].

Please register yourself for the Symposium. We will send further details soon.

We look forward to your participation in the symposium. Further information about the Symposium will be available shortly at https://iscls.github.io/.

Sincerely,
Pawan Goyal
On behalf of the Program Committee, 6th ISCLS

SUBMISSION: 4
TITLE: Sanskrit Sentence Generator

---------- REVIEW 1 ----------
SUBMISSION: 4
TITLE: Sanskrit Sentence Generator
AUTHORS: Amba Kulkarni and Madhusoodana Pai

---------- What is this paper about? ----------

47th ALL INDIA CONFERENCE OF DRAVIDIAN LINGUISTS

&

International Symposium on Language Endangerment

Organized by

Dravidian Linguistic Association of India, International School of Dravidian Linguistics

&

Central University of Karnataka, Kalaburagi

CENTRAL UNIVERSITY OF KARNATAKA

IGNORANCE IS THE CAUSE OF MISTRUST. LET US KNOW EACH OTHER

CERTIFICATE

This is to certify that Mr. / Ms. / Dr. Madhusoodana Bai J.

has participated / presided over a session / presented a paper on Semantics of Morpho-Syntactic case markers in Indian Languages: Samskrit a case study ***in the 47th All India Conference of Dravidian Linguists held at Central University of Karnataka, Kalaburagi from 20th to 22nd June 2019.***

H. M. Maheshwaraiah

PROF. H.M. MAHESHWARAIAH
Hon. Vice Chancellor,
Central University of Karnataka

PROF. NADUVATTOM GOPALAKRISHNAN
Hon. Director, ISDL & Dean, DLA

Bharatiya Vidvat Parishat and Tattwa Samshodhana Samsat (R) Udupi
(In association with Indian Council of Philosophical Research, New Delhi and Indic Academy, Hyderabad)

A THREE DAY INTERNATIONAL CONFERENCE ON

DISPASSIONATE CHURNING OF INDOLOGY

UDUPI CONVENTION - 04-06, JANUARY 2019

CERTIFICATE

This is to certify that ..... Madhusoodana . Pai . J .....

has participated/presented a paper on

..... "Sanskrit sentence generator" .....

in the Three Day International Conference on "Churning of Indology". This certificate is given in appreciation of his/her erudite participation as Resource Person/Panelist/Observer.

Prof. Veeranarayana Pandurangi
COC

Dr. Vamshi Krishna Purohit
Director, Tattwa Samshodhana Samsat

Sanskrit Sentence Generator: A Prototype

ORIGINALITY REPORT

4%	3%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	texts.sangatham.com Internet Source	1%
2	Lecture Notes in Computer Science, 2004. Publication	<1%
3	"Sanskrit Computational Linguistics", Springer Nature, 2010 Publication	<1%
4	link.springer.com Internet Source	<1%
5	aclweb.org Internet Source	<1%
6	sanskrit1.ccv.brown.edu Internet Source	<1%
7	David Martinez, Michelle R. Ananda-Rajah, Hanna Suominen, Monica A. Slavin, Karin A. Thursky, Lawrence Cavedon. "Automatic detection of patients with invasive fungal disease from free-text computed tomography (CT) scans", Journal of Biomedical Informatics,	<1%

	2015 Publication	
8	www01.us.archive.org Internet Source	<1%
9	core.ac.uk Internet Source	<1%
10	Madhurima Dutta, Ajit Kumar Das, Chirantana Mallick, Apurba Sarkar, Asit K. Das. "Chapter 16 A Graph Based Approach on Extractive Summarization", Springer Nature America, Inc, 2019 Publication	<1%
11	Iakovos Gurulian, Raja Naeem Akram, Konstantinos Markantonakis, Keith Mayes. "Preventing relay attacks in mobile transactions using infrared light", Proceedings of the Symposium on Applied Computing - SAC '17, 2017 Publication	<1%
12	Submitted to Institute of Technology, Nirma University Student Paper	<1%
13	baadalsg.inflibnet.ac.in Internet Source	<1%
14	epdf.tips Internet Source	<1%

15	www.ling.mq.edu.au Internet Source	<1%
16	semantics.dcs.kcl.ac.uk Internet Source	<1%
17	en.wikipedia.org Internet Source	<1%
18	people.mokk.bme.hu Internet Source	<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		