Sanskrit Analysis System (SAS)

Manji Bhadra[1], Surjit Kumar Singh[2], Sachin Kumar[3], Subash[4], Muktanand Agrawal[5], R.Chandrasekhar[6], Sudhir K Mishra[7], Girish Nath Jha[8]

[1, 2, 3, 5,8]Special Centre for Sanskrit Studies
Jawaharlal Nehru University
New-Delhi
[4]C-DAC Kolkata, NLP Group
[7]Visiting Scholar, Department of Classics, Brown University
[7]C-DAC Pune, AAI Group
[1]manji.bhadra@gmail.com, [2]surjit.jnu@gmail.com, [3]sachinjnu@gmail.com, [8]girishjha@gmail.com, [4]subhash.jnu@gmail.com, [5]mukta.jnu@gmail.com, [6]chandrashekhara@gmail.com, [7]sudhirkumarmishra@gmail.com

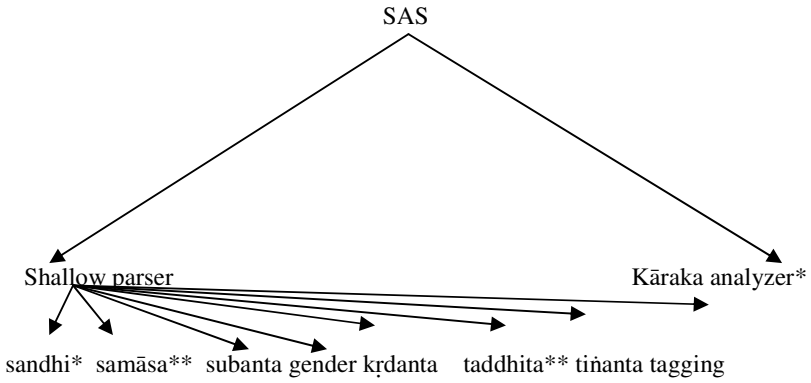
Abstract *The paper describes Sanskrit Analysis System (SAS) – a complete analysis system for Sanskrit. Some modules of this system have already been developed. The system accepts full text inputs in Devanāgarī Unicode (UTF-8). The sandhi module does the segmenting for complex tokens and then hands over the text for detailed processing. Currently, the SAS modules have independent interfaces for unit testing at http://sanskrit.jnu.ac.in . The authors are working on the integration process of these modules. The SAS has two major components - the shallow parser and the kāraka analyzer. The shallow parser has separate modules, some of them are partially implemented, and some of them are in the process of being implemented. The modules have been developed as java servlet in unicode using RDMBS techniques. The applications of the SAS will be many ranging from being a Sanskrit reading assistant to machine translation system from Sanskrit to other languages.*

Keywords: sandhi, subanta, tiṅanta, kṛdanta, samāsa, taddhita, strī pratyaya, kāraka, vibhakti, prātipadika, dhātu, sup, tiṅ, avyaya, sūtra, vārttika, ākāṅkṣā, yogyatā, vivakṣā, liṅga, upadhā, gaṇa, pada, lakāra, vacana, vikāra, upasarga, vṛddhi

1 Introduction

Developing an NLP system which analyzes a natural language is a difficult task. Sanskrit language has Pāṇini's grammar which is explicitly generative, while the task in analysis systems is to apply rules for processing a generated string or utterance. The authors in this process have tried to use Pāṇinian rules in reverse with appropriate lexical interfacing for analyzing Sanskrit. But in many cases, Pāṇinian rules demand deep semantic analysis, especially in the case of *kāraka* rules. The SAS has two major components - one is the shallow parser and the other is the *kāraka* analyzer. Shallow

parser contains *sandhi*[1] analyzer, *samāsa* analyzer, *subanta*[2] analyzer, gender analyzer[3], *kṛdanta* analyzer[4], *taddhita* analyzer, *tiṅanta*[5] analyzer and the POS tagger[6]. Among them the *sandhi* analyzer is partially implemented and *samāsa* and *taddhita* analyzers are yet to be implemented. After getting the input, the system first analyzes the *sandhi* and *samāsa* by these modules wrapped into a separate system. Then the system analyzes the nominal inflected words for separating base and *vibhakti* and stores the PNG features of the tokens. Primary derived nouns are analyzed by the *kṛdanta* analyzer and secondary derived nouns are supposed to be analyzed by the *taddhita* analyzer. The *tiṅanta* analyzer analyzes verbs into affixes, number and person. After morphological analysis of each word in the sentence, the POS tagger assigns them appropriate POS category with the help of lexicon, corpus and other Pāṇini based formulations. The *kāraka*[7] analyzer takes over from here and analyzes the syntactico-semantic relations at the sentence level. A brief modular outline of the SAS is given below –



---

1 Kumar Sachin,2007, '*Sandhi Splitter and Analyzer for Sanskrit (with reference to ac Sandhi)*', M.Phil dissertation, Special Center for Sanskrit Studies, JNU

2 Chandra Subash, 2006, '*Machine recognition and Morphological Analysis of Subanta-padas*', M.Phil dissertation, SCSS, JNU

3 Bhadra Manji, 2007, '*Computational analysis of Gender in Sanskrit Noun Phrases for Machine Translation*', M.Mhil dissertation, Special Center for Sanskrit Studies, JNU

4 Singh Surjit Kumar, 2008,'*Kṛdanta Recognition and Processing for Sanskrit*', M.Mhil dissertation, Special Center for Sanskrit Studies, JNU

5 Agrawal Muktanada, 2007, '*Computational Identification and Analysis of Sanskrit Verb Forms of bhvādigaṇa*,' M.Phil dissertation, Special Center for Sanskrit Studies, JNU

6 Chandrasekhar R, 2007, ' *Part of Speech Tagging for Sanskrit*', Ph.D. thesis, Special Center for Sanskrit Studies, JNU

7 Jha Girish Nath, Misra Sudhir, '*Semantic processing in Pāṇini's karaka system*' Presented in second International Sanskrit Computational Linguistics Symposium at Brown University,2008

For example –

INPUT : भोजनान्तरं * धीमती किन्तु चञ्चला बालिका भ्रमित्वा धातुपाठं पठति ।
PREPROCESSING: भोजनान्तरं धीमती किन्तु चञ्चला बालिका भ्रमित्वा धातुपाठं पठति ।
*SANDHI : भोजन अन्तरं
धीमती किन्तु चञ्चला बालिका भ्रमित्वा धातुपाठं पठति ।
**SAMASA: भोजन अन्तरं
धीमती किन्तु चञ्चला बालिका भ्रमित्वा
धातुणाम् पाठम्
पठति ।
SUBANTA: भोजन अन्तरं
धीमती् सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला सु प्रथमा एकवचन
बालिका सु प्रथमा एकवचन
भ्रमित्वा
धातु ङस् षष्ठी वहुवचन पाठ अम् द्वितीया एकवचन
पठति[Verb]
GENDER: भोजन अन्तरं
धीमती[sf:hf] सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला[sf:hf][टाप्]सु प्रथमा एकवचन
बालिका[sf:hf] [टाप्] सु प्रथमा एकवचन
भ्रमित्वा
धातु [sm:hm]ङस् षष्ठी वहुवचन पाठ[sm:hm]˙ अम् द्वितीया एकवचन
पठति[Verb]> feminine
KṚDANTA: भोजन अन्तरं
धीमती [sf:hf] सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला[sf:hf][टाप्]सु प्रथमा एकवचन
बालिका[sf:hf] [टाप्] सु प्रथमा एकवचन
भ्रमित्वा[भ्रम् त्वाच्]
धातु [sm:hm]ङस् षष्ठी वहुवचन पाठ[sm:hm]˙ अम् द्वितीया एकवचन
पठति[Verb]> feminine
**TADDHITA: भोजन अन्तरं
धीमती [धीमत् मतुप्] [sf:hf] सु प्रथमा एकवचन
किन्तु[AV] चञ्चला[sf:hf][टाप्]सु प्रथमा एकवचन
बालिका[sf:hf] [टाप्] सु प्रथमा एकवचन

भ्रमित्वा[भ्रम् त्वाच्]
धातु [sm:hm]ङस् षष्ठी वहुवचन पाठ[sm:hm]ं अम् द्वितीया एकवचन
पठति[Verb]> feminine
TIṄANTA: भोजन अन्तरं
धीमती [धीमत् मतुप्] [sf:hf] सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला[sf:hf][टाप्]सु प्रथमा एकवचन
बालिका[sf:hf] [टाप्] सु प्रथमा एकवचन
भ्रमित्वा[भ्रम् त्वाच्]
धातु [sm:hm]ङस् षष्ठी वहुवचन पाठ[sm:hm]ं अम् द्वितीया एकवचन
पठति { (कर्तृवाच्य) पठ ([भ्वादिगण] [सेट्] [सकर्मक]) ([लट्]) तिप् ([परस्मै] [प्रथम–पुरुष] [एकवचन]) } > feminine
POS TAGGER: भोजन अन्तरं[N]
धीमती[Adj][धीमत् मतुप्] [sf:hf] सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला[sf:hf][Adj][टाप्]सु प्रथमा एकवचन
बालिका[N][sf:hf][टाप्]सु प्रथमा एकवचन
भ्रमित्वा[भ्रम् त्वाच्]
धातु[N][sm:hm]ङस् षष्ठी वहुवचन पाठ[N][sm:hm]ं अम् द्वितीया एकवचन
पठति { (कर्तृवाच्य) पठ ([भ्वादिगण] [सेट्] [सकर्मक]) ([लट्]) तिप् ([परस्मै] [प्रथम–पुरुष] [एकवचन]) } > feminine
KĀRAKA: भोजन अन्तरं](कर्म)[N]
धीमती(कर्ता)[Adj][धीमत् मतुप्] [sf:hf] सु प्रथमा एकवचन
किन्तु[AV]
चञ्चला(कर्ता)[Adj][sf:hf][टाप्]सु प्रथमा एकवचन
बालिका(कर्ता)[N][sf:hf] [टाप्] सु प्रथमा एकवचन
भ्रमित्वा[भ्रम् त्वाच्]
धातु[N][sm:hm]ङस् षष्ठी वहुवचन पाठ(कर्म)[N][sm:hm]ं अम् द्वितीया एकवचन
पठति { (कर्तृवाच्य) पठ ([भ्वादिगण] [सेट्] [सकर्मक]) ([लट्]) तिप् ([परस्मै] [प्रथम–पुरुष] [एकवचन]) } > feminine

2 Description of each module

2.1 *Sandhi* module

The analysis procedure of the *sandhi* analysis system uses lexical lookup method as well as rule base method. Before *sandhi* analysis process, pre-processing, lexical search of *sandhi* strings in *sandhi* example base and *subanta*-analysis takes place respectively. The pre- processing will mark the punctuation in the input. After that, the program checks the *sandhi* example base. This example base contains words of *sandhi*-exceptions (*vārttika* list) and commonly-occurring *sandhi* strings (example list) with their split forms. These words are checked first to get their split forms without parsing each word for processing. After lexical search, *subanta* analyzer gets the case terminations (*vibhakti*) separated from the base word (*prātipadika*). *Subanta analyzer* also has a function to look into lexicon for verb and *avyaya* words to exclude them from *subanta* and *sandhi* processing.

2.1.1 *Sandhi* rule base

The rule base has been built up in the following format:

input Sanskrit text
↓
pre-processing
↓
subanta processing
↓
rule base
↓
result generator
↓
lexical lookup
↓
subanta processing
↓
lexical lookup
↓
output

Rules for vowel sandhi are in format of

एऽ=ए+अ:(पूर्वरूपसन्धिः,एङःपदान्तादति);ओऽ=ओ+अ:(पूर्वरूपसन्धिः,एङःपदान्तादति);आय्s=ऐ+s:(अयादिसन्धि,एचोऽयवायावः);अय्s=ए+s:(अयादिसन्धि,एचोऽयवायावः);आव्s=औ+s:(अयादिसन्धि,एचोऽयवायावः);अव्s=ओ+s:(अयादिसन्धि,एचोऽयवायावः);?य्s=?ई+s:(यण् सन्धि इको

यणचि); ?य्s=?इ+s:(यण् सन्धि इको यणचि);

In these rules, the Roman character 's' stands for *svara* or vowel. This rule applies on the string after phoneme splitting. When phonemes are split, there are only vowels, consonants, *avagraha*, *visarga* and *anusvara*. For example the rule *आय्s=ऐ+s* means when in the phonemic string a sequence of characters appears as 'आ' followed by 'य्' then '*s*' (or any '*svara*'), then replace it by the right hand side of the '=' sign of the rule. In the RHS of the rule, '*s*' means that *svara* (not any *svara*) which is in LHS of the rule. The case of variable '*s*' is the same as in the rules of *ayādi sandhi*. Some of vowel *sandhi* rules make changes depending upon consonants. Operations depend upon consonants in following ways - on voiced consonants, unvoiced consonants and also as semivowels.

3 *Subanta* analyzer

Sanskrit is a heavily inflected language, and depends on nominal and verbal inflections for communication for meaning. A fully inflected unit is called *pada*. Inflected nouns are called *subanta pada* and inflected verbs are called *tiṅanta pada*. According to Cardona[8], a Sanskrit sentence has NPs (including *avyayas* (AVs)) and VPs. It is defined as (N-En)p…(V-En)p. After *sup* and *tiṅ* combine with *prātipadika* (PDK)[9] they are assigned *kāraka* stipulations to return complete sentence.

3.1 Sanskrit *subanta* (inflected nouns)[10]

Sanskrit nouns are inflected with seven cases in three markers. Sanskrit nouns can be further divided as primary derived forms (*kṛdanta*), secondary derived forms (*taddhita*), compounds (*samāsa*). There are 21 suffixes called *sup* (seven *vibhaktis* combined with three numbers)[11] which can be attached to the nominal bases (PDK) according to the syntactic category, gender and end-character of the base. Apart from these suffixes, there are *upasarga* (prefixes) which can attach to the PDK. But a PDK with only *upasarga* cannot be used in sentence without *vibhakti*. In Sanskrit, there are indeclinable (AVs) which are *subanta* but remain unchanged under all morphological conditions.[12]

---

8 George Cardona, 1988, *Pāṇini His Work and Tradition*, vol…I Delhi(MLBD 1988)

9 अर्थवदधातुरप्रत्ययः प्रातिपदिकम् 1.2.45, कृत्तद्धितसमासश्च 1.2.46

10 Jha Girish Nath et al., '*Inflectional Morphology Analyzer for Sanskrit*, pages 47-66 Proceedings of First International Sanskrit Computational Linguistics Symposium October, 2007

11 स्वौजसमौट्छष्टाभ्याम्भिस्ङेभ्याम्भ्यास्ङसिभ्याम्भ्यास्ङसोसांङ्योस्सुप्

12 सदृशं त्रिषु लिङ्गेषु सर्वासु च विभक्तिषु/ वचनेषु च सर्वेषु यन्न व्येति तदव्ययम् (Gopatha Brāhmaṇa)

3.2.1 Recognition of punctuation

The system recognizes punctuations and tags them with the label _PUNCT. If the input has any extraneous character, then some normalization takes place. For example - UÉ/&^%@#म्:, बा,''':–=लक: ➔ रामः, बालकः .

The Devanāgarī Sanskrit input text is then sent to the analyzer.

3.2.2 Recognition of *avyaya*

The system takes help of *avyaya* database for recognizing AVs. If an input word is found in the AVs database, it is labeled AV, and excluded from the *subanta* analysis. Around 524 *avyayas* are stored in the database.

3.2.3 Recognition of verbs

System takes the help of verb database for verb recognition. If an input is found in the verb database, it is labeled VERB and not sent for further processing. Since storing all Sanskrit verb forms is not a good option for computational reasons (there are 2000 verb roots and forms generated from it would be in the millions. Besides, there are innumerable *nāmdhātu*s as well and a regular verb form can be conjugated as *sannata*, *ṇijanata* etc as well). The SAS has 450 commonly used verb roots and their regular forms plus mechanisms to recognize unseen verbs (in the *tiṅanta* module) as well.

3.2.4 Recognition of *subanta*

Thus a process of exclusion identifies the nouns in a Sanskrit text. After the punctuation, *avyayas* and verbs are identified, the remaining words in the text are labeled as SUBANTA.

3.3 Analysis of *subanta*

System does analysis of inflected nouns with the help of two relational databases – examples and rules. Brief description of these databases follows-

3.3.1 Example database

All complicated forms including those of some pronouns which cannot be easily analyzed according to rules are stored in the database. For example: अहम्=अस्मद्+सु प्रथमा एकवचन;अहं=अस्मद्+सु प्रथमा एकवचन; आवाम्=अस्मद्+औ प्रथमा द्विवचन; आवां=अस्मद्+औ प्रथमा द्विवचन;वयम्=अस्मद्+जस् प्रथमा वहुवचन;वयं=अस्मद्+जस् प्रथमा वहुवचन;

3.3.2 Rule database

The *subanta* patterns are stored in this database. This database analyzes those nouns which match a particular pattern from the rule base. For example, रामः ,नदी, रमा, पुस्तकम् etc. First, the system recognizes *vibhakti* as the end character of nouns. For example, **‘:’** is found in nominative singular like- रामः श्यामः सर्वः भरतः एकः **.** The system isolates ‘:’ and searches for analysis in the *sup* rule base. In the case of nominative and accusative dual, PDK forms will be ending in ‘ौ’ . For example, रामौ, श्यामौ, सर्वौ, एकौ. The system isolates ‘ौ’ and searches for analysis by matching the rule database. The sample data is as follows-

ा=ा+सु प्रथमा एकवचन;ाभ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ाभ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ेभ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;भ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;

4 Gender analyzer

After *subanta* analyzer one can get information about Sanskrit nouns. But still gender information is not fully analyzed by *subanta* analyzer. In Sanskrit, there is gender agreement between adjectives and noun. Though there is no gender agreement between verb and the agent like Hindi, but *kṛdanta* forms agree with agent in terms of gender in a sentence. If machine has to understand Sanskrit language then it needs to understand the gender also like any other grammatical category. In the absence of a correct gender analysis of Sanskrit NPs, the target language translations may be wrong.

4.1 Description of the gender analyzer

The gender analyzer gets each sentence as a token. Then it sends the token for pre processing. After pre processing, it finds the verb and *avyayas* using database and excludes them for further processing. If in the text there are multiple NPs with conjunct or comma, then it gets separated NP chunks separated by conjunct or comma. After that, the system takes help of *subanta* analyzer to obtain PDK. After obtaining PDK, the system takes the help of lexical resources to get gender information of nouns. If enough information about gender is not found then the system looks for rules. At the end, it suggests the collocational gender of a sentence with respect to a target Hindi sentence.

4.1.1 Rule base for gender analyzer

The present *subanta* analyzer of Sanskrit analyses the Sanskrit words with *prātipadika* with *vibhakti* markers. Some *vibhakti* markers help to identify the gender of a word. For example, the word *narān* can be analyzed as masculine gender from the *vibhakti* marker *ān,* and the number of the word would be plural. If the word appears in the input with this particular *vibhakti,* then the gender recognition of the word would be easy. The problem with this method is that the particular word has to arrive in the input with this particular *vibhakti.* As a consequence of this step, there would be huge numbers of words whose gender would be unrecognized by the system.

4.1.2 Rules of *Liṅgānuśāsana*

Gender can be recognized from the last but one syllable of the word. The technical name of this category is *upadhā* (penultimate).[13]

upadhā	**clause**	**Gender**	**Example**	**Exception**
k, ṭ, ṇ, ṭh, n, p, bh, ma, y, r, ṣ,s	if the word ends in *a*	Masculine	*stabaka, ghaṭa,dīpa, bhanu* etc	*chibuka*, *lalāṭa,pāpa, ratna* etc
L	if the word ends in *a*	Neuter	*phala*	*tūla, upala* etc

Among these words, if some words are used as proper names then it would follow the gender of a person if the name is a mythical and famous one, for example *ambarīṣa.* After this step, the gender of a large number of words would remain unrecognized. To handle this problem, another rule from *Liṅgānuśāsana* is implemented for gender analyzer. The rule depends on the last *varṇa* of the word. For example, if the word ends in *ṛ* (*pitṛ, bhrātṛ*), generally the gender of the word would be masculine. But there are exceptions to this rule, like the words *mātṛ, nanāndṛ* etc in the feminine gender.

After the application of the Pāṇinian rules, there are still a large number of Sanskrit words whose gender recognition is very difficult. For these kinds of words, the gender can be recognized from the last syllable of the word apart from the Pāṇinian rule. For example, if the ending syllable is *a* then the gender of the word would be masculine. If the ending syllable is *ā* then generally the gender of that word would be feminine. But there are exceptions to this rule as *viśapā, dārā, hāhā* etc.

---

[13] अलोऽन्त्यात्पूर्व उपधा

5 *Kṛdanta* analysis

All the verbal suffixes besides *tiṅ* are called *kṛt*. *kṛt* is a technical term of Pāṇinian grammar that covers a vast field, both structurally as well as semantically.[14] The primary nominal derivatives from the verb roots are *kṛdanta*. The *kṛt* suffixes are added to roots or their modified forms, to form nouns, adjectives and indeclinables, for example *kṛ - kāra, krtṛ, karaṇa, kurvat, kariṣyat, cakṛvas, kṛtvā, kartum*. These are called *kṛdantas* or primary derived nominal bases.[15]

5.1 *Kṛdanta* identification and analysis mechanisms

The process of *kṛdanta* analysis mechanism is divided into two sections - recognition and analysis.

5.1.1 *Kṛdanta* identification mechanism

The *kṛdanta* recognition starts by an exclusion process. The verb forms, *avyayas* and punctuations are excluded by running POS tagger by checking the verb, *avyaya* and pronoun databases and punctuation lists. The nominal bases are obtained by the *subanta* analyzer which is a part of the POS tagger. These nominal bases are then checked in fixed lists by the POS tagger. This may result in some of the *subantas* being marked for *kṛdanta*. The remaining *subantas* are sent to the *kṛdanta* recognizer and analyzer system for recognition and analysis using following steps –

- check the *kṛdanta* database, annotated corpus and *kṛdanta*-tagged Monier Williams Sanskrit Digital Dictionary (MWSDD).
- the *subantas* still untagged for *kṛdanta* are sent to the rule base for kṛdanta checking.
- the rule base applies *Pāṇinian* rule base in reverse for marking *kṛdantas*.
- it is possible that even after these systematic identification procedures, there may remain an untagged *kṛdanta subanta*. This will count as failure of the system.

14 Sharma, Dipti, *Structure and Meaning*,1982 Nag Publishers New-Delhi

15 Kale, M.R., *A Higher Sanskrit Grammar*

5.1.2 *Kṛdanta* analysis mechanism

The system is divided into two parts- lexical database and rule-base. Lexical database of examples has been created for analyzing those forms which would be otherwise very complex to analyze if passed through the rule base. Lexical database has three major parts- a lexical *kṛdanta* database with complicated *kṛdanta* forms and their lexical information, Monier Williams Sanskrit Digital Dictionary and corpus of the current Sanskrit prose with *kṛdanta* words tagged with *kṛdanta* information.

The rule-base is for analyzing more regular forms. It consists of mainly three tables, namely, *upasargavikāra* table, *dhātuvikāra* table and *pratyayavikāra* table. To restrict *dhātuvikāra* and *pratyayavikāra* from inconsistent combinations, both are bound with a unique id.

For example, पाठकः[(पठ+ण्वुल्/पठ+णिच्+ण्वुल्)प्रथमा–एकवचन]

6 *Tiṅanta* analysis

Verbs have been of central importance to Sanskrit grammarians. Yāska insisted so much on them that he propounded that all the nominal words are derived from verb roots[16]. Like noun *padas,* verb *padas* (*tiṅanta*) have to undergo certain inflectional process in which various verbal affixes are added to verb roots or *dhātus*. These *dhatus* are encoded with the core meaning of the verb. These can be primitive[17] or derived[18]. Primitive verb-roots, which are around 2000 in number, have been listed in a lexicon named *dhatupātha*. They are divided in 10 groups/classes called *gaṇas*. All the verb-roots of a group undergo somewhat similar inflectional process. Derived verb-roots may be derived from primitive verb-roots or from nominal forms. Prefixes also play an important role as they can change the meaning of a verb root. These roots then have to undergo various inflectional suffixes that represent different paradigms. In this process, the base or root also gets changed.

6.1 Process of formation of Sanskrit verb forms

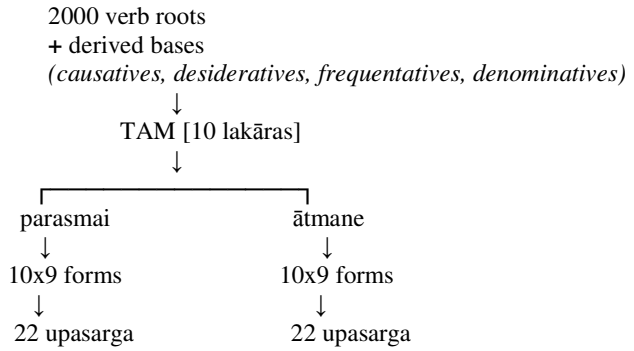
A Sanskrit verb root may take various forms. There are ten lakāras that represent Tense, Aspect and Mood. Inflectional terminations are 18 in number. These are divided in two groups – *parasmaipada* and *ātmanepada*, each having 9 affixes which is a combination of 3 persons x 3 numbers. A verb is conjugated in either *pada*,

16 *bhāvapradhānamākhyātam* (Yāska, *Nirukta*)

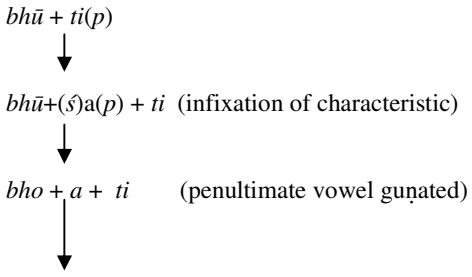
17 *bh*uv*ādayo dh*ā*tavaḥ* (*P* 1/3/1)

18 *san*ā*dyant*a *dh*ā*tava*ḥ (*P* 3/1/32)

though some of the roots are conjugated in both. For each different *lakāra*, a root is affixed with these 9 terminations. Again, there are three voices- Active, Passive and Impersonal. Transitive verbs are used in the Active and Passive voices while intransitive verbs are conjugated in the Active and Impersonal voices. Addition of one or more of 22 prefixes (*upasargas*) to verb roots can result in more variety of forms. Derivative verb roots, both derived from verb roots as well as nominal words, also follow the same process to form verb forms. There can be some specific rules and exceptions in some cases. The following chart gives a rough estimate of possible verb-forms in Sanskrit[19] . This is to suggest that Sanskrit verb forms can not be stored in the database because the derived verb forms can be potentially innumerable.



The verb roots of different *gaṇas* adapt certain terminations when *tiṅ* affixes are added to them. The *tiṇ* affixation also influences the verb root and it undergoes several morpho-phonemic changes, for example, having *guṇa* operation on the end vowel. The verb root can adopt certain operations resulting in the final verb-forms.



19 Mishra Sudhir K., Jha, Girish N., 2004, *Identifying Verb Inflections in Sanskrit morphology*, in proc.of SIMPLE 04, IIT Kharagpur, pp. 79-81.

bhav a ti (*ayādi sandhi*)

As shown in the example, when suffix *tip* is added to the verb root *bhū*, then *bhavati* form is obtained as the final verb form. This can be cited as a common analysis of most verb forms.

6.2 The Analysis of Sanskrit Verb Forms

The methodology for the analysis of Sanskrit verb form in the present work follows the analysis of Pāṇini in somewhat reverse direction. Pāṇinian analysis identifies different morphemes in any given *pada* and presents an analysis where he provides step-by-step methodology to derive a verb form from a given verb root in certain paradigms. As illustrated above, Sanskrit verb forms are a blend of multiple morphemes which contain relevant information. Analytically it can be said that the first element is the conjugational affix that remains at the end of every verb form. These affixes have encoded information of *pada* (though it is determined by root), lakāra, person and number. Thus termination can serve as the most important thing to convey about the paradigm information of any verb form. They can be a tool to identify a verb form in a given text. The terminations, as they are basically replacements of 18 original *tiṅ* affixes in different *lakāras*, differ among themselves according to *lakāras*. However in each *lakāra,* they are similar for all verb roots of various groups, leaving some expectation. So *ti* can be used to identify any verb form of present tense of *parasmaipada.* But some terminations can vary among themselves for a group of *gaṇas*. Then again, the termination may be changed due to morphophonemic environment, *tā* affix of *luṭ lakāra,* changing to *ṭā* with roots like *yaj*.

Further left, there are various morphemes of the various characteristics and increments inserted between the verb root and terminations, in the process of their formation explained above. *Bhvādigaṇa* verb forms in conjugational *lakāras*, have '*a*' as a result of *śap* characteristics; *svādi* roots have *no*, *nu* or *nv* all of them remaining morphemes of *śnu*. Some roots like that of *adādi* have no such characteristics sign infixed in them.

At the right end of the verb form, there is modified stem of the verb root. The modification can be *guṇa*, *vṛddhi* or any other. Generally a root adopts a common stem in all the forms for both *padas* in conjugational *lakāras*. So *bhav* is the stem for all *parasmaipadī* forms in the conjugational *lakāras*. But there are exceptions to it to that extent that four or five types can be found among nine forms of a single *lakāra pada*.

Here the first morpheme the *tiṇ* termination is common among all verb forms of a particular *pada-lakāra- puruṣa -saṁkhyā* combination. Second constituent, the characteristics (existing in the form of its remaining morpheme) and increments inserted in between may differ, yet being almost the same in a particular group. The

third constituent, the modified verb-root is particular in the strict sense. In the analysis, the recognition of the *tiṅ* will identify a word as a verb form and find out its *pada-lakāra- puruṣa -saṁkhyā*. The second morpheme can, in many cases, be helpful to recognize the *gaṇa* of a particular root because the characteristics in a *lakāra* are determined by the *gaṇa* that the roots belong to. Thus the core of the analytical approach is that each *tiṅanta* verb form can be analyzed to form a unique combination of verbal stem + *tiṅ* termination, and both of these constituent parts are stored in separate tables. When it is to be analyzed, its constituent morphemes are recognized and identified with the help of pre-stored structured data.

7. POS Tagger

After getting the information about inflected nouns of Sanskrit, it is necessary to understand the role of each word in a sentence. This process of marking up the words in a text as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e., relationship with adjacent and related words in a phrase, sentence, or paragraph[20] is called POS tagging. A typical POS tagger acts as a shallow parser and is pre- requisite in several NLP related applications such as machine translation system, information retrieval word sense disambiguation etc. Sanskrit is an inflectional language and words in a sentence carry information about entities in terms of stem, endings, gender, case, number and case relation, while verbs denote activity, function reaction, mode, voice, tense, person, number etc. Extracting and organizing, i.e. annotating, these information is the first step towards understanding the language. Words in a language may occur in POS or various grammatical categories as they are also known. In Sanskrit for example

1) *gacchati* can be either a *tiṅanta* or *kṛdanta*
2) *rāmaḥ* can either be a *nāmapada* (*abhidhāna*) or *tiṅanta*
3) *āyātaḥ* can either be a *kṛdanta* or *tiṅanta*
4) *mā* can either be an *avyaya* or *namapada* or a *sarvanāman* etc

7.1 The Sanskrit Tagset

The designed tagset is classified according to the morphological structure of the categories. There are two kinds of tags in this tagset. Word class main tags and feature sub-tags. The tag as a whole is a combination of word class main tag with feature sub-

[20] http://en.wikipedia.org/wiki/Part-of-speech_tagging

tags separated by an underscore. All the tags bear Sanskrit names[21] with letter-digit acronymic in Roman script.

The process first involved evolving a stable tagset for Sanskrit text which has 65 word class tags, 43 feature sub-tags, and 25 punctuation tags and one tag *UN* to tag unknown words – a total of 134 tags. A single full tag is a combination of word class tag and feature sub-tags (indeclinable and punctuation tags do not have sub-tags). The word class tags are 8 Noun tags, 8 Pronoun tags, 3 Adjective tags, 9 Participle tags, 2 Number tags, 14 Compound tags, 11 indeclinable tags and 10 verb tags. Feature tags are three gender sub tags (p,s,n); 8x3 = 24 (Nominal)Case and Number tags (1.1 through 8.3); 4 Verb base modifying tags (Nd, Yn, Sn, Ni); 1Verbal Preposition (UPA); 2 Pada tags (P and A); 3x3 = 9 (Verbal) Person and Number tags (1.1 through 3.3).

7.2 Description POS Tagger

7.2.1 Pre-processing

After getting Unicode (UTF-8) sandhi free Devanagari Sanskrit input (or with minimal sandhi) as word, sentence or text, the system sends those input for pre processing. In this step, the system searches for punctuations in the input and tags them. In addition to tagging the punctuations, this function also removes unwanted foreign letters or punctuations from the inside of a Devanagari string.

7.2.2 Fixed-List Tagger

After initializing the input, the system goes to check in the fixed tagged lists. This database stores lists of *avyayas*, list of verbs and POS list. The POS example base consists of approximately 1 MB data. For example

अञ्चति[P_laTV_1.1]/[KV1_p_7.1]/[KV1_n_7.1];अञ्चतः[P_laTV_1.2]/[KV1_p_2.3]/[KV1_p_5.1]/[KV1_p_6.1]/KV1_n_5.1]/[KV1_n_6.1];अञ्चन्ति[P_laTV_1.3]/[KV1_n_1.3]/[KV1_n_2.3];अञ्जति[P_laTV_1.1]/[KV1_p_7.1]/[KV1_n_7.1];अञ्जतः[P_laTV_1.2]/[KV1_p_2.3]/[KV1_p_5.1]/[KV1_p_6.1]/KV1_n_5.1]/[KV1_n_6.1];अञ्जन्ति[P_laTV_1.3]/[KV1_n_1.3]/[KV1_n_2.3];नाम[N_n_1.1]/[(nAman)N_n_2.1];प्रथमम्[N_n_1.1]/[(prathama)N_n_2.1]/[N_p_2.1]/[AVKV];तन्त्रम्[N_n_1.1]/[(tantra)N_n_2.1];यस्य[SNS_n_6.1]/[(yad)SNS_p_6.1];महान्[NVI_p_1.1]/[(maha)N_p_2.3];सिंहगोवृषयोः[N_p_6.2]/[(siMhagovRuSha)N_p_7.2]/[N_s_6.2]/[(siMhagov

[21] Few names are coined in English for the purpose of clarity and to avoid confusion while marking their notions. The tags having English names are all the compound tags containing 'C' for Compound and few punctuation tags.

RuShA)N_s_7.2];वने[N_n_7.1]/[(vana)N_n_1.2]/[(vana)N_n_2.2];तत्[SNN_n_1.1]/[(tad)SNN_n_2.1];महिलारोप्यम्[N_n_1.1]/[(mahilAropya)N_n_2.1];नगरम्[N_n_1.1]/[(nagara)N_n_2.1];

If the token is found, it gets tagged with corresponding tag from the lexicon.

7.2.3 *subanta* analyzer

However, a large number of input tokens are not found in these lists as they may be marked for *subanta*. Therefore the next component of *subanta* analyzer checks untagged input in the *subanta* examples. If not found, it starts analyzing the token from the right end and checks in the lexicon after each appropriate cut. If it is found, it tags the input. If after all these steps, the input remains untagged, it gets the 'not found' tag. The resultant tagged token is sent back to the main tagger 'Post' which linearizes the results with adding color schemes for ambiguous and untagged tokens.

8 *Kāraka* analysis

After understanding words in a Sanskrit sentence, it is necessary to understand how the words are arranged in a sentence, what are the relation between other words and verbs. In Sanskrit, this relation can be understood while analyzing *kāraka* relation. Etymologically *kāraka* is the name given to the relation between a noun and a verb in a sentence. It means 'that which brings about' or 'doer'. [22]

8.1 *Kāraka* and *vibhakti* mapping

Pāṇini discusses the entire gamut of *kāraka-vibhakti* relations in three sections of *Aṣṭādhyāyī*

- *kāraka sūtra* (P. 1.4.23 – P. 1.4.55) → 33 *sūtras*
- *vibhakti sūtra* (P. 2.3.1 - P. 2.3.73) → 73 *sūtras*
- *karma-pravacanīya* (P. 1.4.82 – P. 1.4.97) →16 *sūtras*

Now the problem of implementation of all *kāraka* rules is that there are rules of *vivakṣā* dependent operations. In the example स्थाल्या पचति, स्थाली should be

[22] A detailed description of kāraka and its mapping with vibhakti is given in Jha Girish Nath, Mishra Sudhir K., '*Semantic processing in Pāṇini's kāraka system*' Presented in second International Sanskrit Computational Linguistics Symposium at Brown University,2008

Location as it is the आधार (आधारोऽधिकरणम्), but it is करण by rule साधकतमं करणम् because the speaker thinks it is the most instrumental (साधकतम (प्रकृष्ट उपकारक)) and therefore prefers Instrumental case. कर्मणा यमभिप्रैति स सम्प्रदानम prescribes Dative for the receiver of gift, but vārtika अशिष्टव्यवहारे दाणः प्रयोगे चतुर्थ्यर्थे तृतीया prohibits it if the gift was intended for deriving some benefit (sexual favor in this case).Vārtikas extend, limit Pāṇinian rules, for example - नी–वह्योर्न (नाययति वाहयति वा भारं भृत्येन) allows karaṇa if the verb is नी or वह . It thus limits गतिबुद्धिप्रत्यवसानार्थशब्दकर्माकर्मकाणामणि कर्ता स णौनियन्तृकर्तृकस्य वहेरनिषेधः which allows karma. Sometimes vārtikas limit themselves नियन्तृकर्तृकस्य वहेरनिषेधः (if the *kartā* is '*sārathi*' or any of its synonyms then नी–वह्योर्न does not apply → वाहयति रथं वाहान् सूतः (*karma* by Pāṇini's गतिबुद्धि… sūtra). Another problem is how to implement semantic conditions such as, स्वातंत्र्य,ईप्सित/ईप्सिततम्,साधकतम,अभिप्रैति (to be अभिमुख - approach someone for gift), प्रीयमाणः (one who gets pleased - रुच्यर्थानां प्रीयमाणः) – हरये रोचते भक्तिः,ध्रुव (fixed point) अपाय (path of separation) ध्रुवमपायेऽपादानम्,धावतः अश्वात् पतति (is अश्वात an आधार or ध्रुव ?) ,आधारोऽधिकरणम (आधारः किम् ?) etc.

A tentative model of *kāraka* analyzer is given below.

1. VERB ID
2. VERB ANALYSIS
3. NON—VERB ID
4. SUBANTA ANALYSIS
****5. ĀKĀṄKṢA CHECK**
****6. KĀRAKA RULES**
****7. SPECIAL CONDITIONS**
8. KĀRAKA ASSIGNMENT

In this model, the starred modules are under implementation. While analyzing the verb, the system will take the help of *tiṅanta* analysis. For tokenizing the *tiṅanta* , the system checks every character of the word through reverse module and matches through verb database for recognizing the *tiṅanta pada* which is used in the sentence. If it is found, then all information which is relevant in *kāraka* analysis are provided to system for further implementation. Otherwise it returns to check again if *dhātu* is used with *upasarga* and after recognizing *upasarga,* the system removes the *upasaga* from the verb, and again checks it for *dhātu* identification number and the result is sent to *dhātu* information database for getting the relevant information of the *dhātu*. After the verb analysis, the system checks for non verb words and then it takes help of *subanta* analyzer and *kāraka* assignment is implemented. In between, there are some steps like *ākāṅkṣā* checking, and special semantic conditions are not implemented yet.

9 Result analysis and limitations

Currently the modules of the SAS are not integrated. Individual modules can be tested as http://sanskrit.jnu.ac.in. The limitation of lexical resource may affect some modules. In *sandhi* analyzer, if the input is हिमालयः the output will be हिमाले अः (अयादिसन्धि एचोऽयवायावः),हिमाली अः (यण् सन्धि इको यणचि), हिमालि अः (यण् सन्धि इको यणचि), हिमा अलयः (दीर्घसन्धि अकः सवर्णे दीर्घः), हिम अलयः (दीर्घसन्धि अकः सवर्णे दीर्घः) हिम आलयः (दीर्घसन्धि अकः सवर्णे दीर्घः). Here, the system gives multiple answers with appropriate rules of Pāṇini as it finds all the parts in these results as valid words in the 200k Sanskrit dictionary. Future enhancements in this module will select the most common output based on a frequency marling in the dictionary. The *subanta* analyzer can not recognize many forms and is being currently updated. In the gender analyzer, lexical resources may hamper the result. The system cannot identify gender of those words which are used in different gender in different meaning properly, like the word *mitra*. Sometimes the system fails to check proper gender agreement as well. There are limitations of the *Kṛdanta, Tiṅanta*, POS tagger and *Kāraka* modules as well which are being improved currently.

10 Conclusion

The authors in this paper have presented an ongoing work for developing a complete SAS. Currently, the SAS has some modules partially developed and some under development. Significant future additions will be the ambiguity resolution modules like anaphora resolution. After the *kāraka* checking module, a disambiguation module is also going to be added in near future, to resolve problems like '*bhavati ! bhikṣām dehi*'. Here according to the SAS system, *bhavati* and *dehi* both get verb tags. But here *bhavati* is used as noun and in vocative. If there is proper punctuation like an exclamation mark after this word then one can say it is used in vocative. If there is no punctuation mark then the problem can be resolved by counting verbs in the sentence which in most cases can be only one. These kinds of problems are to be handled in the disambiguation module. For the testing of the system, 140 files in unicode Devanāgari have been collected. Those texts are in simple Sanskrit and collected from different sources mostly samples of current Sanskrit. Though there is no complete statistics of the results, but one of the tests in *subanta* with simple Sanskrit gave a 90% accuracy.

The table is given below.

S.No.	File	Theme	Source	Words	Time(secs)
1	Corpus-1	*rājā sagaraḥ*	*sandeśaḥ*	609	3
2	Corpus-2	*samrāṭa aśokaḥ*	*sandeśaḥ*	916	3.2
3	Corpus-3	*ekaḥ nibandhaḥ*	*sandeśaḥ*	882	3
4	Corpus-4	*cācā neharuḥ*	*sandeśaḥ*	332	1
5	Corpus-5	*sarasvatī vandanā*	*sandeśaḥ*	241	1

		and a story			
6	Corpus-6	*ādhunika praśāsanaḥ*	*sandeśaḥ*	1045	3.5
7	Corpus-7	*ekaḥ vaṇikaḥ*	*sandeśaḥ*	849	2
8	Corpus-8	*paśya me rūpāṇi*	*sandeśaḥ*	1328	4
9	Corpus-9	Sanskrit *sikśā*	*sandeśaḥ*	306	2
10	Corpus-10	*saṅghe śaktiḥ*	*sandeśaḥ*	4207	6

References

1. Acharya Vamdeva, 1990. 'Liṅga-Parijnānam', Shabdatattva Prakāshan Varanasi
2. Agrwal Muktanada, 2007, 'Computational Identification and Analysis of Sanskrit Verb Forms of bhvādigaṇa,' submitted for Mphil degree at SCSS, JNU
3. Bhadra Manji, 2007, 'Computational analysis of Gender in Sanskrit Noun Phrases for Machine Translation', submitted for Mphil degree at SCSS, JNU
4. Bharati, Akshar, Vineet Chaitanya & Rajeev Sangal, 1991, A Computational Grammar for Indian Languages Processing, Indian Linguistics Journal, pp 52, 91-103
5. Bharati, Akshar, Vineet Chaitanya & Rajeev Sangal, 1995, Natural Language Processing: A Pan Perspective, Prentice-Hall of India, New Delhi.
6. Cardona George, 1967, Pāṇini's syntactic categories, Journal of Oriental Institute Baroda 16:201-15
7. Cardona George, 1988, Pāṇini His Work and Tradition vol…I Delhi(MLBD 1988)
8. Cardona George, 2004, Some Questions on Pāṇini's Derivational System, procs of Splash, iSTRANS, Tata Macgraw-Hill, New Delhi, pp 3
9. Chandrasekhar R, 2007, ' Part of Speech Tagging for Sanskrit' , submitted for Phd degree at SCSS, JNU
10. Jurafsky Daniel and James Martin, 2000, Speech and Language Processing, Prentice-Hall of India, New Delhi.
11. Edgren A.H 1885, On the verbal roots of the Sanskrit language and of the Sanskrit grammarians, Journal of American Oriental Society 11:1-5
12. Huet, G'erard, 2003, Towards Computational Processing of Sanskrit, RecentnAdvances in Natural Language Processing, Proceedings of the International Conference ICON, Mysore, India
13. Jha, Girish N. et al., 2006, Towards a Computational analysis system for Sanskrit,Proc. of first National symposium on Modeling and Shallow parsing of Indian Languages at Indian Institute of Technology Bombay, pp 25-34
14. Jha, Girish N, 2003 A Prolog Analyzer/Generator for Sanskrit Noun phrase Padas,Language in India, volume-3,
15. Jha, Girish N, 2004, Generating nominal inflectional morphology in Sanskrit, SIMPLE 04, IIT-Kharagpur Lecture Compendium, Shyama Printing Works, Kharagpur, WB. Page no. 20-23.
16. Jha, Girish N., 1993, Morphology of Sanskrit Case Affixes: A computational analysis, M.Phil dissertation submitted to J.N.U., New Delhi
17. Jha, Girish N., 2004, The system of P, Language in India, volume4:2,
18. Jha Girish Nath, Misra Sudhir, 'Semantic processing in Pāṇini's karaka system' Presented in second International Sanskrit Computational Linguistics Symposium at Brown University,2008

19. Joshi, S. D., 1962, Verbs and nouns in Sanskrit, Indian linguistics 32 : 60- 63.
20. Kale, M.R.,1995, A Higher Sanskrit Grammar, MLBD, New-Delhi
21. Kapoor, Kapil, 1985, Semantic Structures and the Verb: a propositional analysis,Intellectual Publications, New Delhi
22. Kumar Sachin,2007, 'Sandhi Splitter and Analyzer for Sanskrit (with reference to ac Sandhi)', submitted for Mphil degree at SCSS, JNU
23. Mishra Sudhir K., Jha, Girish N., 2004, Identifying Verb Inflections in Sanskrit morphology, In proc. of SIMPLE 04, IIT Kharagpur, pp.79-81
24. Mishra, Sudhir K & Jha, Girish N, 2004, Sanskrit Karaka Analyzer for Machine Translation, In SPLASH proc. of iSTRANS, Tata McGraw-Hill, New Delhi, pp. 224-225.
25. Mitkov Ruslan, The Oxford Handbook of Computational Linguistics, OxfordUniversity Press.
26. Narayan Mishra, 1996, (ed). Kashika of Pt.Vamana and Jayaditya ,Chaukhamba Sanskrit sansthan, Varanasi
27. Nooten, B. A. Van, Pāṇini's replacement technique and the active finite verb,University of California, Berkeley.
28. Sharma, Rama NATH, 2003, The Aṣtādhyayi of Pāṇini, Munshiram Manoharlal Publishers Pvt. Ltd., Delhi.
29. Shastri, Bheemsen, Laghusiddhantakaumudi , Bhaimee Prakashan, 537, Lajapatrai Market, New Delhi
30. Singh Surjit Kumar, 2008,'Kṛdanta Recognition and Processing for Sanskrit', submitted for Mphil degree at SCSS, JNU
31. Shastri, Swami Dwarikadas, 2000, 'The Madhaviya Dhatuvṛtti by Saya_acarya', Tara Book Agency, Varanasi.
32. Sharma, Dipti, 1982 , Structure and Meaning, Nag Publishers New-Delhi
33. Subash & Jha, Girish N., 2005, Morphological analysis of nominal inflections in Sanskrit, presented at Platinum Jubilee International Conference, L.S.I. at Hyderabad University, Hyderabad, pp-34.
34. Subash, 2006, Machine recognition and morphological analysis of Subanta-padas, M.Phil dissertation submitted to J.N.U., New Delhi.
35. Upadhye, P.V., 1927, Dhaturupacandrika, Gopal Narayen & Co, Bombay.
36. Whitney, W.D., 2002, History of Sanskrit Grammar, Sanjay Prakashan, Delhi.

Web References:

- IIIT,Hyderabad,http://www.iiit.net/ltrc/Publications/Techreports/tr010/anu00kbcs.txt
- Peter M. Scharf and Malcolm D. Hyman, http://sanskritlibrary.org/morph/
- Huet's site http://sanskrit.inria.fr/
- Prajna system, ASR Melcote, http://www.sanskritacademy.org/Achievements.htm
- Aiba, Verb Analyzer for classical Sanskrit,
- http://wwwasia.human.is.tohoku.ac.jp/demo/vasia/html/
- Desika, TDIL, Govt. of India, http://tdil.mit.gov.in/download/Desika.htm
- RCILTS, JNU, http://rcilts.jnu.ac.in
- Shabdabodha, ASR, Melcote, http://vedavid.org/ASR/#anchor2
- http://en.wikipedia.org/wiki/Part-of-speech_tagging