KRIDANTA ANALYSIS FOR SANSKRIT

N. Murali[1], Dr. R.J. Ramasreee[2] and Dr. K.V.R.K. Acharyulu[3]

[1]Department of Computer Science, S.V. Vedic University, Tirupati
[2]Department of Computer Science, R.S. Vidyapeetha, Tirupati
[3]Professor of Vyakarana (Retd.), R.S. Vidyapeetha, Tirupati

ABSTRACT

Kridantas play a vital role in understanding Sanskrit language. Kridantas includes nouns, adjectives and indeclinable words called avyayas. Kridantas are formed with root and certain suffixes called Krits. Some times Kridantas may occur with certain prefixes. Many morphological analyzers are lacking the complete analysis of Kridantas. This paper describes a novel approach to deal completely with Kridantas.

KEYWORDS

Avyaya, Kridanta, Morphological Analyzer, Natural Language Processing, upapada, upasarga

1. INTRODUCTION

Word is considered as the most basic unit of the linguistic structure. Word is a sequence of characters delimited by space. Word consists of a complex set of more primitive parts. The study of morphology is concerned with the construction of words from more basic meaningful units called morphemes. The process of analyzing the given word to extract the information encoded in the word is called as morphological analysis. Morphological analysis deals with the segregation of word into morphemes. By identifying the morphemes of a given word, the form (syntax) and meaning (semantics) of the word can be understood. Morphological Analyzer (MA) is a tool which analyzes the given word into its root, affixes and feature values of the grammar like number, gender, person etc.

A. MAs for Sanskrit

A thorough survey has been conducted on existing MAs and the following four MAs for Sanskrit language have been selected for the present study.

1. The Sanskrit Reader Companion developed at INRIA
2. MA developed at HU
3. MA developed at JNU
4. MA developed at UoH

The stories of Balakanda from Ramayana which were published in the children's fortnightly magazine called Chandamama has been collected and used as a test data to test the above mentioned MAs. The test data contains a total number of 28,713 words, out of which 8,981 words are unique words. The rich linguistic features of Sanskrit like Sandhi, Samasa, kridantas, taddithas etc., have great impact on the performance of the above mentioned MAs. Out of these, kridantas is one of the very interesting features of Sanskrit which includes nouns, adjectives and indeclinable words / avyayas.

2. KRIDANTAS

In Sanskrit, the words can be broadly classified into 2 categories viz. Declinable and Indeclinable [2]. The following figure illustrates this [16].

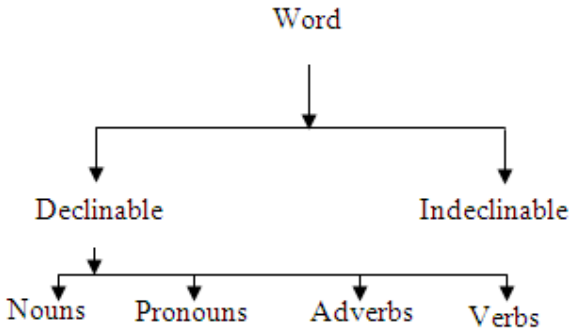

Figure 1. Classification of Words

Declinable or inflectional words are those words whose base form can be changed or inflected. For example, the base form or *pratipadikam* "*Rama*" can be inflected in 8 *vibhaktis* i.e., cases and three *vacanas* i.e., numbers. These declinable/inflectional words can again be categorized as Nouns, Pronouns, Adverbs and Verbs. In Sanskrit, indeclinable words are called *avyayas*.
Nouns and adjectives can be derived from verbs and this process is called nominalization. The term *krit* represents a process of nominalization in Sanskrit in which certain affixes can be coalesced with roots to generate nouns or adjectives or indeclinable words i.e., *avyayas*. The words which have been so generated are called as *kridantas*. These affixes are called *krit* suffixes. All these suffixes were described in the text *siddhantakaumudi* under a separate chapter called *Kridanta prakaranam*. *Krit* affixes are used to indicate *karakas*. The activity element possessed by the root lies inactive in the verbal derivative nouns and adjectives. In case of avyayas the root acts as an infinitive and directly related to the main verb in the sentence. All *krit* suffixes may be used in the senses of different *karakas*. Panini has mentioned that, if no other sense is assigned to a *krit* affix, then it should be understood that *karta* or agent of the verbal activity [1]. Certain *upapadas* and *upasrgas* may change the meaning of a *kridanta* completely. The following examples illustrate *kridantas* and shows how a root can be inflected in different ways with *upapada* or *upasarga* combinations.

Example 1: रामः (*Ramah*) - means who makes every one happy and is derived from

Verbal Root रम् (*ram*) and

Krit Suffix घञ् (*Ghay*)

Example 2: विरामः (Viramah) - means recession / rest / relaxing and is derived from

Prefix वि (*vi*)

Verbal Root रम् (*ram*) and

Krit Suffix घञ् (*Ghay*)

Example 3: आरामः (*Aramah*) - means abode and is derived from

Prefix आ (A)

Verbal Root रम (*ram*) and

Krit Suffix घञ् (*Ghay*)

Kridantas may be nouns, adjectives and *avyayas*. The *Kridantas* nouns and adjectives fall in the category of declinable words and *avyaya* falls in the category of indeclinable words. The nouns and adjectives are treated similarly. Hence the process of identification of *Kridantas* is divided into two stages

Identification of *Avyaya* and
Identification of *Kridantas*

Avyaya, types of *avyayas*, the process of recognizing *avyayas* and other *kridantas* have been presented in the following sections.

3. AVYAYAS

The following *sloka* gives a beautiful and simple definition for *avyayas*

सदृशं त्रिषु लिङ्गेषु सर्वासु च विभक्तिषु ।
वचनेषु च सर्वेषु यन्न व्येति तदव्ययम् ।।
sadrusam trishu lingeshu srvasu ca vibhaktishu.
vacaneshu ca sarveshu yanna vyeti tadavyayam.

The *sloka* connotes that the words which cannot be changed or inflected or which remain immutable in all genders, numbers and cases; are called avyayas [2], [6]. But, there is an exception to some nouns (Nominative or Locative), which have only one declension e.g. *adau, samvat, svar, svaha, svadha* etc. Figure 2 illustrates various types of *avyaya* [16]. Some grammarians have classified avyayas into Substantive and Non-substantive. The examples for Substantive *avyayas* are *asti, adau, samvat, svar* etc. Non-substantive avyayas can be further divided into two types i.e., Conditional and Relational. *cet, yat-tat, yatha-tatha, yadi-tarhi, yatra-tatra, yavat-tavat* etc. are the examples of Conditional avyayas, while *atha, api, iti, iva, eva, evam, kila, khalu, ca, tu, manye, vat, va* etc. for relational avyayas. For the present study, avyayas have been classified into two types i.e., Primitive/ Paniniyan avyayas and Secondary / Lakshanika avyayas.

3.1. Primitive/ Paniniyan *avyayas*

These types of *avyayas* are directly named by the Sanskrit grammarian Panini as *nipatas*. These avyayas can be used as prepositions, interjections, particles and conjunctions and can be called as nipatas i.e., exceptional or irregular forms. All *nipatas* are indeclinable. All *upasargas* / prefixes can also be treated as nipatas and are indeclinable.

3.2. Secondary / *Lakshanika avyayas*

These are used as adverbs or adjectives. *Kridanta avyayas*, *taddhitanta avyayas* and adverbial, determinative (of type *pradi*, *gati*) and attributive types of compounds fall into this category.

3.2.1 *Kritpratyayanta* words as *avyayas*

Kritpratyayas are used to form nouns and avyayas from roots [2], [6]. These avyayas denote some action, which have a relation to the main verb in the sentence.

3.2.2 *Taddhitapratyayanta* words as *avyayas*

Ttaddhitapratyayas are those suffixes that are used to form nouns or *avyayas* from other nouns.

3.3.3 Compounds as *avyayas*

Some compounds such as Adverbial / *avyayibhava*, Determinative / *tatpurusha* (of type *pradi* and *gati*) and Attributive / bahuvrihi also act as avyayas.

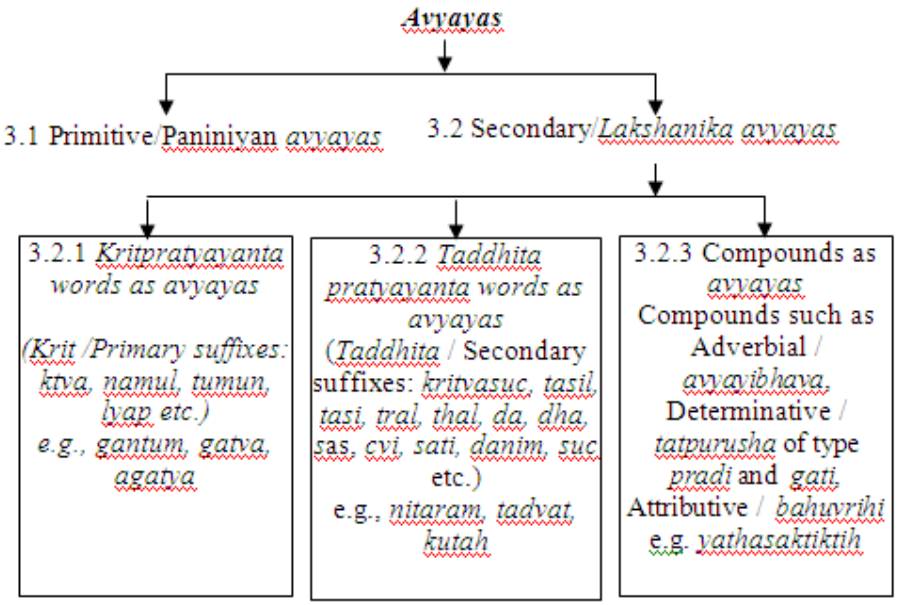

Figure 2. Classification of avyayas

4. AVYAYA ANALYZER

Avyaya Analyzer deals fully with the primitive type *avyayas* i.e., 3.1, and *Kridantas* i.e., 3.2.1 and partly with *Taddhitas* i.e., 3.2.2. Dealing with avyayas of primitive type is very simple because it has already been defined as avyayas by Panini in *nipatas*. Some of the examples of primitive type avyayas are asti, adau, atha, api, iti, iva, eva, evam, katham, kila, khalu, ca, tu, manye, vat, va etc. Here a simple lookup is enough to identify the avyaya of this category. The *avyayas* of type 3.2.2 had been encoded with information and play a vital role in understanding the sentence. Consider the following examples:

Gantum means “to go”
Agantum means “to come”
Gatva means “having gone”
Agatya means “having come”

The above examples were derived from a single root gam, prefix “a” and *krit* suffixes *tumun*, *ktva* and *lyap*. Each word indicates a different meaning because of the suffixes. All these *avyayas*

indicate incomplete action of their constituent verbal roots and have a relation to the main verb in the sentence and act as an adverb. The following simple sentences illustrates the role of *avyayas*
Sentence: *aham gantum icchami.*

POS: Noun avyaya verb
Meaning: I wish to go.
Sentence: *aham agantum icchami.*
POS: Noun avyaya verb
Meaning: I wish to come.
Sentence: *bhavan tatra gatva vadatu*
POS: Noun avyaya avyaya verb
Meaning: having gone there, you tell.
Sentence: *atra agatya upavisatu*
POS: avyaya avyaya verb
Meaning: having come, sit here.

In all the examples given above, *avyayas* act as adverbs. Both the *avyaya* and main verb are mutually dependent on each other and it is impossible to understand the meaning of the sentence without gaining a proper understanding of the avyaya.

Hence the structure of *avyaya* may be as follows:

$$Avyaya \rightarrow (Upapada^*)\ (Upasarga^*)\ \text{root suffix}$$

Upapada and *upasarga* are optional. The Avyaya Analyzer will give the derivational morphological information of an *avyaya* i.e., the *upapada*, *upasarga*, root and suffix information.

4.1 Representation of linguistic information

Avyaya Analyzer relates the root and linguistic features to the surface form through a set of transformations. 2000 verbal roots, 120 *krit* suffixes (out of which four suffixes are related to *avyayas*), 22 *upasargas* and nearly 500 *upapadas* were collected. With the combination of all these *upapadas, upasargas*, roots and suffixes, we can generate a large number of *avyayas*. The *upapada, upasarga*, root and suffixes acquire different forms due to euphonic transformations between them. Hence, each and every possible phonetic change of roots, suffixes, *upasargas,* and *upapadas* were written manually along with the necessary linguistic features. Morphological dictionaries for *upapadas, upasargas*, roots and suffixes were created. An example of the possible phonetic changes in *upapada, upasarga*, root and suffix are given below. For computational purpose, the data is presented in WX transliteration scheme. Converters are available to convert the text from WX transliteration to Devanagari Unicode UTF-8 and vice-versa.

Upapada:

sva/sv,sva
IRax/IRan/IRac/IRaj/IRal/IRaw,IRaw
iram/iraM,ira
jala/janaM/janam/jala/jana/jal/jana, jana

*U*pasrga:
nir/niH/nil,nir
vi/vy/v,vi
anu/Anu/anv/Anv/onu/onv,anu
apa/ap/opa/op/Apa/Ap,apa

Root:

raMram/riraM/ram/raM/rAm/ran/rem/raw/ra,ram_1_A_853
iR/IR/eR/ER,iR_4_pa_1127
iR/IR/eR/ER,iR_6_pa_1351

krit suffix:

anIya/aNIya/nIya/NIya,anIyar
Davaw/iwavaw/Iwavaw/navaw/Navaw/tavaw/wavaw/ Xavaw,kwavawu

Contents of *upapada, upasarga* and *krit* suffix are separated with comma. The second field indicates the surface form of *upapada* / *upasarga* and the first field indicates all possible phonetic changes of the second field in context of various euphonic transformations. The second field in the Root gives the information about the root, class, type, and root number as given in siddhantakaumudi.

4.2 Identification of avyayas

AA checks whether the word is avyaya or not by noticing whether the word is *nipata* or by comprehending its suffix. If the word is *nipata*, then it will be treated as *avyaya*. If the word ends with any one of the *krit* suffixes related to avyaya, then it return suffix info and strips the suffix and checks for *upapada* [11]. As *upapada* and *upasarga* are optional, they may or may not be present in the given word. Hence, if *upapada* is present, then returns the *upapada* info and strips it. Then the presence of *upasarga* will be tested. If *upasarga* is found, then returns *upasarga* info and strips it. Then checks whether the remaining part of the word is the root or not. If it is, then return the root info; or else control will be transferred to the next stage. The following figure illustrates the process of analyzing the avyaya.

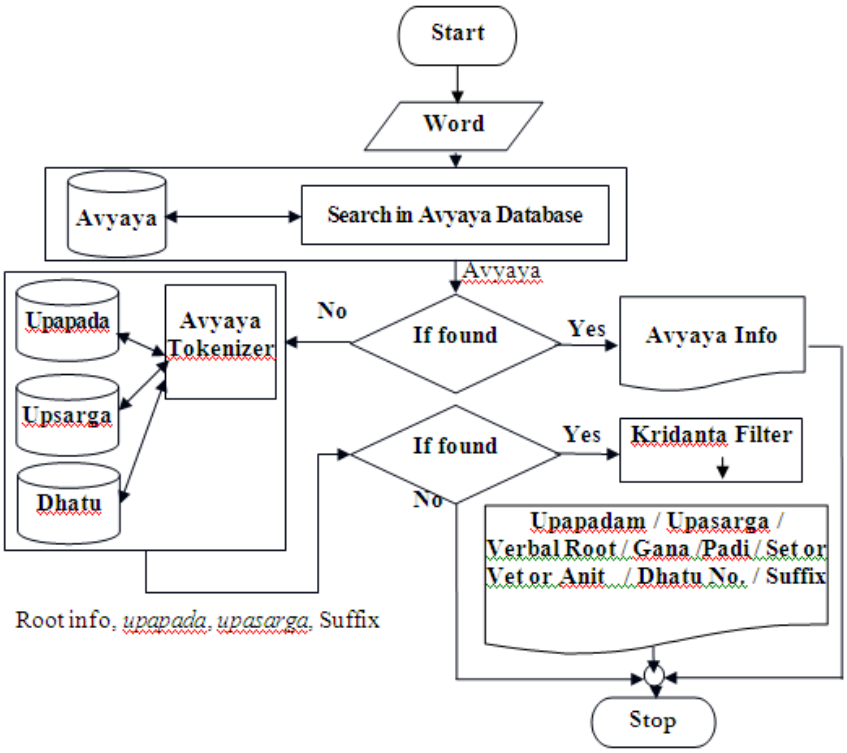

Figure 3. Avyaya Analyzer

The sample output of the Avyaya Analyzer in WX transliteration scheme is presented below. The output is rewritten for convenience.

Input Word: BaFjayiwum (bhanjayitum)
Output:

```
murali@murali: /usr/local/scl/SHMT/MT/prog/morph
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl AvFn
l.pl BaFjayiwum
BaFj(Nic)10<vargaH:avy><kqw_prawyayaH:wumun><XAwuH:Bajiz><
gaNaH:churAxiH><level:1>/BaFj(Nic)10<vargaH:avy><kqw_prawy
ayaH:wumun><XAwuH:Bajiz><gaNaH:churAxiH><level:1>/BaFj(Nic
)10<vargaH:avy><kqw_prawyayaH:wumun><XAwuH:Bajiz><gaNaH:ch
urAxiH><level:1>
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$
```

Figure 4. Output of the Avyaya Analyzer

Input Word: *BojayiwvA* / (*bhojayitva*)
Output:

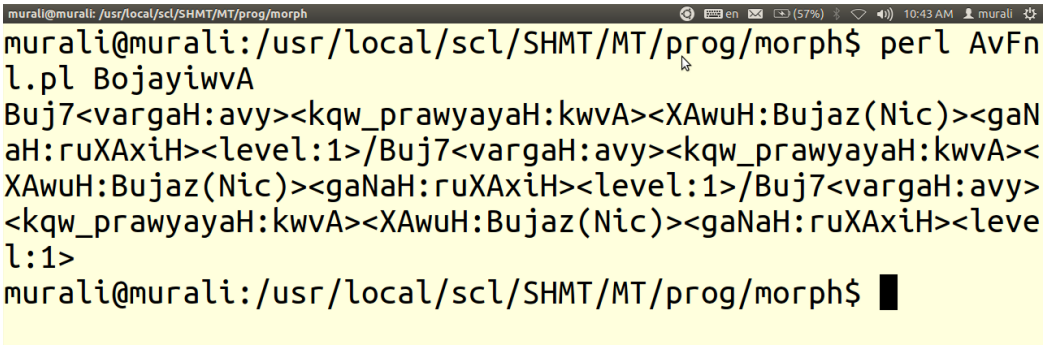

Figure 5. Output of the Avyaya Analyzer

Input Word: *lakRyIkqwya* / lakshyikritya
Output:

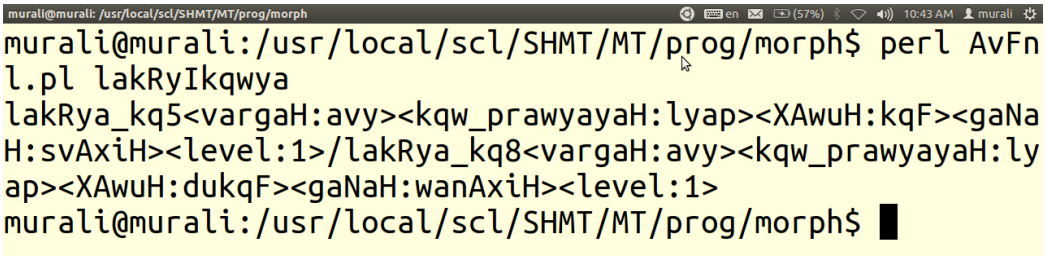

Figure 6. Output of the Avyaya Analyzer

5. KRIDANTA RECOGNITION

Kridantas have been discussed briefly in Section 2. *Kridantas* may be nouns or adjectives or *ayayas*. Some times a *kridanta* can be treated as a verb when there is no main verb in the sentence. The following example illustrates this.

Example: *nirgawaH*

The word mentioned above is in nominative case. It is derived from an *upasrga –nir*, root *–gam*, *krit* suffix *–kwa* and a *sup* suffix. This word functions as a verb. A word consisting of *krit* suffix like *kta, ktavatu* denotes the action done by the agent. In Sanskrit, adjectives will also take suffixes and can be inflected in all cases and in all numbers just like nouns. Hence, adjectives and nouns are treated in a similar way in the process of analyzing them. As it is pointed earlier, *krit* suffixes are used in the process of generating nouns, adjectives and *avyayas*. Avyayas cannot be inflected, but nouns and adjectives can be inflected by adding 21 *sup pratyayas* (suffixes) to the base form or *pratipadika* of a *kridanta*. The surface form of a *kridanta* will contain a root, *krit pratyaya* and a *sup pratyaya*. *Upapada* and *Upasarga* are optional.

kridanta → (*upapada**) (*upasarga**) Root *KritPratyaya SupPratyaya*

A *kridanta* may contain any number of *upapadas* and *upasargas*. Sometimes a *kridanta* may contain another *kridanta* as *upapada*. Hence to analyze a *kridanta* properly, it is essential to know its base form or *pratipadika*, case, gender and number. The information regarding the case, gender and number will be useful in the process of identification of phrases and understanding the semantic role of each word. The process of identification of *kridantas* is described below.

5.1. Identification of Pratipadikam

The process of reducing the inflected form to its stem is known as *stemming* [8]. For example, in English, a stemmer should identify the word "cats" to the stem "cat". In 1968, Julie Beth Lovins published the stemmer [3] and later Stemmer was written by Martin Porter [4] for English. The stemming process for Sanskrit was described here. In Sanskrit the stem is called as "*pratipadikam*". In Sanskrit the process of identifying the base form of an adjective is similar to that of a noun. The base form of a noun can be identified by removing the *sup pratyaya* from the surface form. By drawing some clues from the Porter's stemming algorithm, a stemmer for Sanskrit has been developed to retrieve the base form along with the linguistic features like gender, case and number.

Generally a student who wants to learn Sanskrit will start with learning the basic noun forms like rAma, ramA etc. The remaining nouns will be similar to these basic noun patterns. A Sanskrit basic text called *Roopchandrika* [5] consists of 206 noun types or patterns. It gives basic information regarding gender, word forms in eight cases and three numbers (Singular, Dual and Plural numbers) for each word and totally each stem has 24 inflections. In Sanskrit, all nouns and adjectives follow any one of the pattern of intonation. Based on high frequency of usage, only 23 stems have been selected among the 206 stems and are given in the table 1. The information regarding the inflections, gender, suffix, case, and number was stored in a table which will be used as *Paradigms* [9], [10]. The Suffix information was stored separately in a manner which will be useful in stripping the suffixes.

Table 1: Words used in designing paradigm table

Sl. No.	Word	Gender
1.	ramA	F
2.	mawi	F
3.	naxI	F
4.	lakRmI	F
5.	mAwq	F
6.	xiS	F
7.	rAma	M
8.	hari	M
9.	SamBu	M
10.	Bavaw	M
11.	Piwq	M
12.	XAwq	M
13.	Sarva	M
14.	suhqw	M
15.	guNin	M
16.	rAjan	M
17.	rAj	M
18.	jFAna	N

19.	vAri	N
20.	XAwq	N
21.	wapas	N
22.	XanuR	N
23.	nAman	N

The 21 inflectional forms belonging to seven cases (excluding vocative case) of the above mentioned words were collected to serve as paradigms and stored in a systematic way as indicated below:

Table 2: Paradigm Table

Word	Gender	Inflected Form	Remove	Add	Case	Number
rAma	M	rAmayoH	ayoH	a	6#7	2
rAma	M	rAmAByAm	AByAm	a	3#4#5	2
rAma	M	rAmasya	asya	a	6	1
rAma	M	rAmO	O	a	1#2	2
rAma	M	rAmam	am	a	2	1
rAma	M	rAmaH	aH	a	1	1
rAma	M	rAmeRu	eRu	a	7	3
rAma	M	rAmeNa	eNa	a	3	1
rAma	M	rAmEH	EH	a	3	3
rAma	M	rAmeByaH	eByaH	a	4#5	3
rAma	M	rAme	e	a	7	1
rAma	M	rAmAya	Aya	a	4	1
rAma	M	rAmAw	Aw	a	5	1
rAma	M	rAmANAm	ANAm	a	6	3
rAma	M	rAmAn	An	a	2	3
rAma	M	rAmAH	AH	a	1	3

The first field in the above table indicates the base form or stem, the second field indicates the gender, the third field indicates the surface form or inflection, the fourth field indicates the suffix to be removed, the fifth field indicates the suffix to be added, the sixth field indicates the *vibhakti*, the seventh field indicates the *vacanam*. The contents of Paradigm table are stored in a text file. Each column is separated with a "$" symbol. The suffixes from the *Paradigm Table* have been extracted from all these inflections and recorded separately. The suffix in the surface form or the input is identified from the Suffix Info table. Then a simple lookup was done on Paradigm Table for the suffix. The suffix part from the surface form will be stripped and replaced with the 5^{th} field in the Paradigm Table to generate the stem of the given word. Simultaneously the gender, case and number information may also be obtained from 2^{nd}, 6^{th} and 7^{th} fields from the Paradigm Table. There may be a chance of getting more than one stem for the given word. In this case all the stems that were produced were considered as candidates for the second phase. Figure 7 illustrates the idea of finding the base form from the input word which is in surface form.

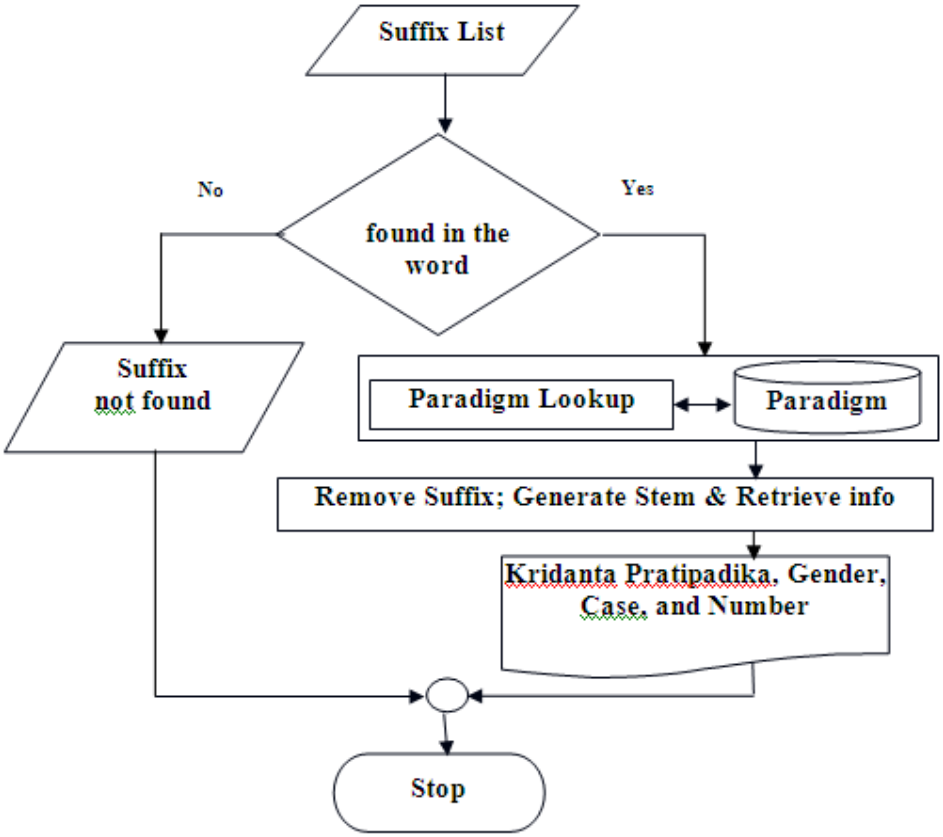

Figure 7. Stemmer and Feature Extractor

The surface form is *AgawavAn* (*Agatavan*). For this surface from the Stemmer and Feature extractor has given two possible *pratipadikas*, gender, case and number. The output is separated with "/" symbol. The first part indicates the unrecognized word. The second part gives various possible stems of the unrecognized word along with its gender, case and number. Each possible solution is separated with "%" symbol. The *pratipadikam*, gender, case and number are separated with a comma. The output of the Stemmer and Feature Extractor for Sanskrit is given below.

Input word: *AgawavAn*
Output:

```
murali@murali: /usr/local/scl/SHMT/MT/prog/morph
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl 12dfw.pl AgawavAn | xargs perl stem.pl
AgawavAn/Agawava,M,2,3%Agawavaw,M,1,1
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$
```

Figure 8. Output of the Stemmer and Feature Extractor

The information produced in this first phase was passed to the second phase to ensure whether the given word is *kridanta* or not. The base form along with the other features are passed to the Kridanta Analyzer for the analysis in order to find the encoded information regarding root, *krit*

suffix, *upapada* and *upasarga* for all possible base forms listed by the Stemmer and Feature Extractor.

5.2. Kridanta Analyzer

Kridanta Analyzer deals fully with the *kridantas* except *u¸¡di* suffixes and suffixes related to *Vedic svar¡s.* It is mentioned earlier that *kridantas* may be nouns or *avyayas* or adjectives. Except *tumun, ktv¡, ¸amul* and *lyap* the other *k¤it pratyayas* are used to derive nouns from verbs. These four suffixes are used to form *avyayas* or indeclinable forms which were discussed in the previous section. Kridanta Analyzer will give the derivational morphological information of a *k¤idanta* i.e., the *upapada, upasarga*, root and suffix information. Linguistic information for *upapada, upasarga, k¤it* has been represented in the format as mentioned in Section IV - A. The following figure illustrates the functioning of Kridanta Analyzer.

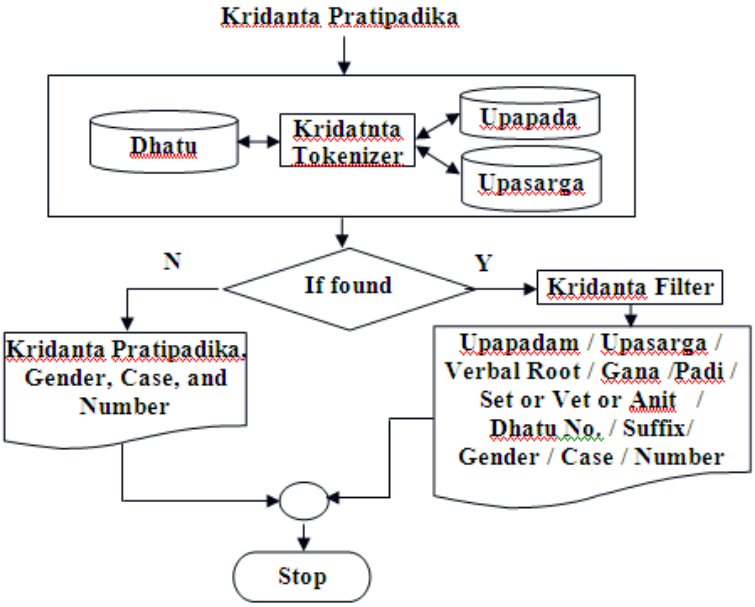

Figure 9. Kridanta Analyze

5.2.1 Identification of *kridantas*

To check whether the given word is *kridanta* or not, identify the pratipadika (stem) of the given word as mentioned in the previous section and this *pratipadikam* will be the input for the Kridanta Analyzer. First, it checks whether the given word is a *kridanta* and if it does not contain any *upapada* or *upasagra*, then it gives the root and suffix information. If it fails to analyze the *kridanta*, it means that the given word may contain any *upapada* or *upasarga* or both along with the root and suffix or the word may not be a *kridanta*. As *upapada* and *upasarga* are optional, they may or may not present in the given word. Hence, if *upapada* exists then it returns the upapada info and strips it, or else it checks whether *upasarga* exists or not. If *upasarga* is found, then returns *upasarga* info and then strips it. Finally it checks the root and suffix and returns the same. This process will continue for all possible candidate solutions generated in the first phase. The sample output of the Kridanta Analyzer is given below.

Input: *aBijFAwam* (*abhijnatam*)
Output:

```
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl 12df
w.pl aBijFAwam | xargs perl stem.pl | xargs perl 27octkqw.
pl
<Word: aBijFAwam><vargaH: KQ><lifgam: M><viBakwiH: 2><vaca
nam: 1><level: 1><upapaxam: xxx><upasargaH: aBi><prAwipaxi
kam: aBijFAwam/aBijFAwa><XAwuH: jFA9_jFA><prawyaya: kwa#><
gaNaH: 9><paxI: pa>
<lifgam: M><viBakwiH: 2><vacanam: 1><level: 1><upapaxam: n
aF><upasargaH: xxx><prAwipaxikam: aBijFAwam/BijFAwa><XAwuH
: jFA9_jFA><prawyaya: kwa#><gaNaH: 9><paxI: pa>
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$
```

Figure 10. Output of the Kridanta Analyzer

Input: *aBijFAwavAn* (*abhijnatavan*)
Output:

```
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl 12df
w.pl aBijFAwavAn | xargs perl stem.pl | xargs perl 27octkq
w.pl
<Word: aBijFAwavAn><vargaH: KQ><lifgam: M><viBakwiH: 1><va
canam: 1><level: 1><upapaxam: xxx><upasargaH: aBi><prAwipa
xikam: aBijFAwavAn/aBijFAwavaw><XAwuH: jFA9_jFA><prawyaya:
 kwavawu#><gaNaH: 9><paxI: pa>
<lifgam: M><viBakwiH: 1><vacanam: 1><level: 1><upapaxam: n
aF><upasargaH: xxx><prAwipaxikam: aBijFAwavAn/BijFAwavaw><
XAwuH: jFA9_jFA><prawyaya: kwavawu#><gaNaH: 9><paxI: pa>
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$
```

Figure 11. Output of the Kridanta Analyzer

Input: *aBinayanwI* (*abhinayanti*)
Output

```
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl 12df
w.pl aBinayanwI | xargs perl stem.pl | xargs perl 27octkqw
.pl
<Word: aBinayanwI><vargaH: KQ><lifgam: F><viBakwiH: 1><vac
anam: 1><level: 1><upapaxam: xxx><upasargaH: aBi><prAwipax
ikam: aBinayanwI/aBinayanwI><XAwuH: NIf1_nI><prawyaya: Saw
q#Sawq><gaNaH: 1><paxI: u>
<lifgam: F><viBakwiH: 1><vacanam: 1><level: 1><upapaxam: n
aF><upasargaH: xxx><prAwipaxikam: aBinayanwI/BinayanwI><XA
wuH: NIf1_nI><prawyaya: Sawq#Sawq><gaNaH: 1><paxI: u>
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$
```

Figure 12. Output of the Kridanta Analyzer

Input: *AcaranwI* (A*caranti*)
Output:

```
murali@murali:/usr/local/scl/SHMT/MT/prog/morph$ perl 12df
w.pl AcaranwI | xargs perl stem.pl | xargs perl 27octkqw.p
l
<Word: AcaranwI><vargaH: KQ><lifgam: F><viBakwiH: 1><vacan
am: 1><level: 1><upapaxam: xxx><upasargaH: Af><prAwipaxika
m: AcaranwI/Acaranw><XAwuH: caraz1_car><prawyaya: Sawq#Saw
q><gaNaH: 1><paxI: pa>
<Word: AcaranwI><vargaH: kQ><lifgam: F><viBakwiH: 1><vacan
am: 1><level: 2><upapaxam: xxx><upasargaH: Af><prAwipaxika
m: AcaranwI/caranw><XAwuH: caraz1_car><prawyaya: Sawq#Sawq
><gaNaH: 1><paxI: pa>
```

Figure 13. Output of the Kridanta Analyzer

Failure of recognizing a *kridanta* may occur in two cases:

If the *upapada* / *upasargas* not found in the table
If the given word is not a *kridanta*

In first case, Kridatna Analyzer has an efficient mechanism to deal with such type of words. First the word will be matched from the right hand side, letter by letter till the maximum match is found. Then the first part which is not recognized by the Kridanta Analyzer will be treated as "*pUrvapaxam*" (*purvapadam*) and the analysis for the remaining portion of the word that was recognized by the Kridatna Analyzer will be presented as specified above. Consider the following example.

Input Word: AgamanakAraNam (*Agamanakaranam*)

In the first iteration Kridanta Analyzer cannot directly analyze the above mentioned word. The reason is the word "Agamana" is not available in the *upapda* table. Hence, the Kridanta Analyzer will start analyzing the word letter by letter from the end of the word towards the beginning of the word. Now Kridanta Analyzer will take the maximum matching part as the final solution. From the given input word, the unmatched portion will be treated as *purvapadam* and the analysis for the maximum matched portion will be considered as the candidate solution. The analysis for the given input word "AgamanakAraNam" is given below:

Input Word: AgamanakAraNam (¡gamanak¡ra¸am)
Output:

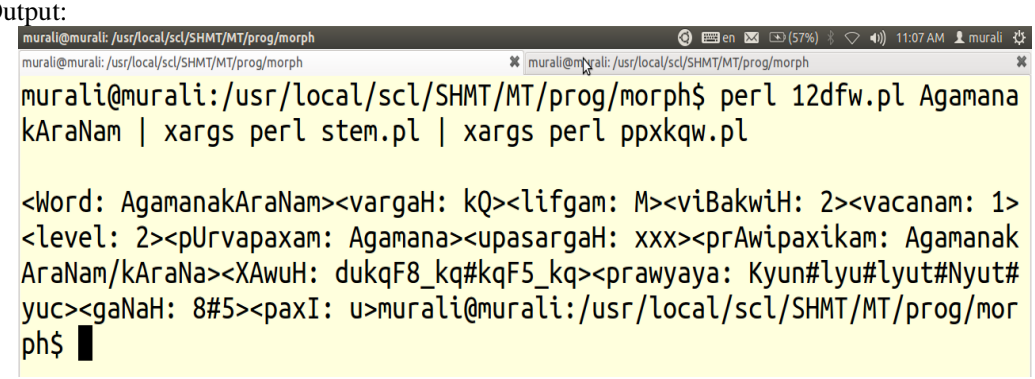

Figure 14. Output of the Kridanta Analyzer

Input Word: *aKAlamaraNam* (*akalamaranam*)
Output:

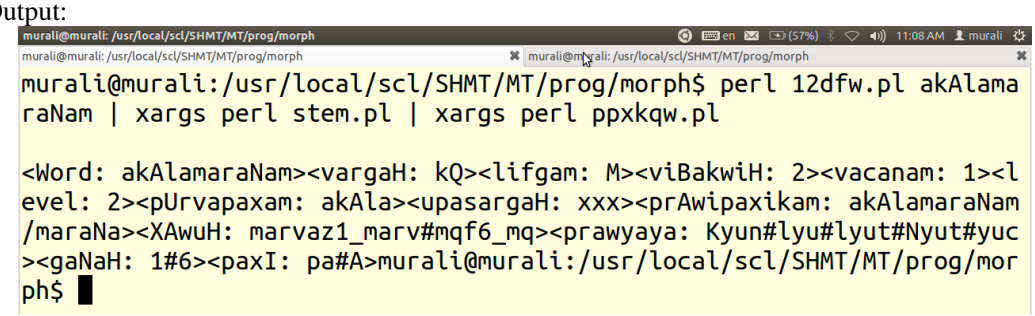

Figure 15. Output of the Kridanta Analyzer

In the second case, the basic information generated by the Stemmer and Feature Extractor regarding gender, case, number will be returned by the Kridanta Analyzer.

6. PERFORMANCE EVALUATION OF AVYAYA ANALYZER AND KRIDANTA ANALYZER

Precision and Recall are the two measures that are widely used to evaluate NLP systems [7]. The Avyaya Analyzer and Kridanta Analyzers have been tested with another test data collected from Sanskrit Consortium, University of Hyderabad. Total number of words is 4573, out of these 534 words are classified as *kridantas*, 99 words are classified as *avyayas*. The precision and recall of both the tools are given in table 3.

Table 3: Precision and Recall of the tools developed

Tool	Total No. of words Tested	Correctly Recognized Words (A)	Unrecognized Words (B)	Wrongly Recognized Words (C)
Avyaya Analyzer	99	73	18	08
Kridanta Analyzer	534	474	23	37

Precision = (A/(A+C)) X 100
Recall = (A/(A+B)) X 100

Precision and Recall for Avyaya Analyzer is measured to 90.123% and 80.219%. Precision and Recall for Kridanta Analyzer is measured to 92.75% and 95.37%.

7. CONCLUSION

In Sanskrit both *kridantas* and *avyayas* play an important role in a given sentence. At elementary level of Sanskrit and daily conversation one can find that *kridanta* and *avyaya* have been used very frequently. Hence without the analysis of *kridanta* and *avyaya,* it is impossible to understand a sentence properly. The advantages of Avyaya Analyzer and Kridanta Analyzers are given below.

- The Kridanta Analyzer and Avyaya Analyzer have been developed using PERL. Hence, these tools can be integrated with any MA with slight modifications for the analysis of *avyayas* and *kridantas*.
- Stemmer and Feature Extractor, Avyaya Analyzer and Kridanta Analyzer can process a single word or a file. All these three tools are integrated through a shell script. If all these three tools are improved further these tools can be used as an independent MA.

There are certain limitations with both the analyzers.

- The language filter must be improved to filter the solutions which are not required.
- In some cases the Kridanta Aanlyzer may give wrong gender information.
- Both the analyzers may fail in analyzing certain *Kridantas* which belongs to n*ic*
- Complete *Taddhita* Analysis is yet to be done.

ACKNOWLEDGMENT

We sincerely thank Dr. Amba P. Kulkarni, Department of Sanskrit Studies, University of Hyderabad who has given consent to study the tools developed by her. We extend our gratitude to Dr. Ankil Kumar, IIIT, Hyderabad and Mr. Narayana, R.S.V.P., Tirupati who helped us in testing the tools developed by us.

REFERENCES

[1] K.V. Abhyankar. (1961). A Dictionary of Sanskrit Grammar. Oriental Institute of Baroda.

[2] Chakradhar Nautiyaalhansa Shastri. (1966). Brihadanuvadachandrika. Motilal Banarsidas, ND.

[3] Lovins, J. B. (1968). Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics, vol.11, nos.1 and 2.

[4] Porter, M. F. (1980). An algorithm for suffix stripping. Program: electronic library and information systems, 14(3), 130-137.

[5] Jha, Ramachandra. (Eds.). (2007). Rupachandrika. Chaukhambha Sanskrit Series Office. Varanasi.

[6] Dr. R.V.R. Krishna Sastri. (1997). Samskritavyakaranam. Krishnanada Mutt, Hyderabad

[7] Melamed, I. D., Green, R., & Turian, J. P. (2003, May). Precision And Recall Of Machine Translation. In Proceedings Of The 2003 Conference Of The North American Chapter Of The Association For Computational Linguistics On Human Language Technology: Companion Volume Of The Proceedings Of HLT-NAACL 2003, Short Papers, Vol. 2, pp. 61-63. Association For Computational Linguistics.

[8] Daniel Jurafsky And James H. Martin. (2004). Speech And Language Processing. Pearson Education. New Delhi.

[9] Jayan, J. P., Rajeev, R. R., & Rajendran, S. (2009). Morphological Analyser For Malayalam-A Comparison Of Different Approaches. IJCSIT, 2(2), pp. 155-160.

[10] Menaka, S., & Sobha, L. (2009). Optimizing The Tamil Morphological Analyzer.

[11] Parakh, M., & Rajesha, N. (2011). Developing Morphological Analyzers For Four Indian Languages Using A Rule Based Affix Stripping Approach. In Proceedings Of Linguistic Data Consortium For Indian Languages, CIIL, Mysore.

[12] Murali N., Ramasree RJ. (April, 2011). Kridanta Analyzer. In proceedings of Annual International Conference on Emerging Research Areas. Organized by Amal Jyothi College of Engineering, Kerala. pp. 63-66

[13] Murali, N, Ramasree, R. J., & Acharyulu, K. V. R. K. (2012). Avyaya Analyzer: Analysis of Indeclinables using Finite State Transducers. International Journal of Computer Applications, 38(6), 7-11.

[14] Murali N, Ramasree RJ. (November, 2013). Rule-based Extraction of Multi-Word Expressions for Elementary Sanskrit Texts. International Journal of Advanced Research in Computer Science and Software Engineering - Volume 3, Issue 11, pp. 661-667. ISSN: 2277 128X

[15] Murali N., Ramasree RJ. (April, 2011). Kridanta Analyzer. In proceedings of Annual International Conference on Emerging Research Areas. Organized by Amal Jyothi College of Engineering, Kerala. pp. 63-66

[16] Murali, N, Ramasree, R. J., & Acharyulu, K. V. R. K. (2012). Avyaya Analyzer: Analysis of Indeclinables using Finite State Transducers. International Journal of Computer Applications, 38(6), 7-11.

Authors

N. Murali
Department of Computer Science,
Sri Venkateswara Vedic University, Tirupati – 517 502
email: murali.nandi@gmail.com
Website: www.pp16-17.org

Prof. R.J. Ramasree
Department of Computer Sicence,
Rashtriya Sanskrit Vidypaeetha, Tirupati – 517507
email: rjramasree@yahoo.com
Blog: http://rjramasree.blogspot.in/p/research-publications.html

Prof. K.V. Ramakrishnamacharyulu
Professor of Vyakarana,
Department of Vyakarana (Retired),
Rashtriya Sanskrit Vidypaeetha, Tirupati – 517507
email: kvrkus@gmail.com
Website: www.pp16-17.org