Analysis of Sanskrit text : parsing and semantic relations

Pawan Goyal, Vipul Arora, Laxmidhar Behera

ANALYSIS OF SANSKRIT TEXT: PARSING AND SEMANTIC RELATIONS

Pawan Goyal
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
pawangee@iitk.ac.in

Vipul Arora
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
vipular@iitk.ac.in

Laxmidhar Behera
Electrical Engineering,
IIT Kanpur,
208016, UP,
India
lbehera@iitk.ac.in

Abstract

In this paper, we are presenting our work towards building a dependency parser for Sanskrit language that uses deterministic finite automata(DFA) for morphological analysis and 'utsarga apavaada' approach for relation analysis. A computational grammar based on the framework of Panini is being developed. A linguistic generalization for Verbal and Nominal database has been made and declensions are given the form of DFA. Verbal database for all the class of verbs have been completed for this part. Given a Sanskrit text, the parser identifies the root words and gives the dependency relations based on semantic constraints. The proposed Sanskrit parser is able to create semantic nets for many classes of Sanskrit paragraphs(अनुच्छेद). The parser is taking care of both external and internal sandhi in the Sanskrit words.

1 INTRODUCTION

Parsing is the "de-linearization" of linguistic input; that is, the use of grammatical rules and other knowledge sources to determine the functions of words in the input sentence. Getting an efficient and unambiguous parse of natural languages has been a subject of wide interest in the field of artificial intelligence over past 50 years. Instead of providing substantial amount of information manually, there has been a shift towards using Machine Learning algorithms in every possible NLP task. Among the most important elements in this toolkit are state machines, formal rule systems, logic, as well as probability theory and other machine learning tools. These models, in turn, lend themselves to a small number of algorithms from well-known computational paradigms. Among the most important of these are state space search algorithms, (Bonet, 2001) and dynamic programming algorithms (Ferro, 1998). The need for unambiguous representation has lead to a great effort in stochastic parsing (Ivanov, 2000).

Most of the research work has been done for English sentences but to transmit the ideas with great precision and mathematical rigor, we need a language that incorporates the features of artificial intelligence. Briggs (Briggs,1985) demonstrated in his article the salient features of Sanskrit language that can make it serve as an Artificial language. Although computational processing of Sanskrit language has been reported in the literature (Huet, 2005) with some computational toolkits (Huet, 2002), and there is work going on towards developing mathematical model and dependency grammar of Sanskrit(Huet, 2006), the proposed Sanskrit parser is being developed for using Sanskrit language as Indian networking language (INL). The utility of advanced techniques such as stochastic parsing and machine learning in designing a Sanskrit parser need to be verified.

We have used deterministic finite automata for morphological analysis. We have identified the basic linguistic framework which shall facilitate the effective emergence of Sanskrit as INL. To achieve this goal, a computational grammar has been developed for the processing of Sanskrit language. Sanskrit has a rich system of inflectional endings (vibhakti). The computational grammar described here takes the concept of vibhakti and

karaka relations from Panini framework and uses them to get an efficient parse for Sanskrit Text. The grammar is written in 'utsarga apavaada' approach i.e rules are arranged in several layers each layer forming the exception of previous one. We are working towards encoding Paninian grammar to get a robust analysis of Sanskrit sentence. The paninian framework has been successfully applied to Indian languages for dependency grammars (Sangal, 1993), where constraint based parsing is used and mapping between karaka and vibhakti is via a TAM (tense, aspect, modality) tabel. We have made rules from Panini grammar for the mapping. Also, finite state automata is used for the analysis instead of finite state transducers. The problem is that the Paninian grammar is generative and it is just not straight forward to invert the grammar to get a Sanskrit analyzer, i.e. its difficult to rely just on Panini sutras to build the analyzer. There will be lot of ambiguities (due to options given in Panini sutras, as well as a single word having multiple analysis). We need therefore a hybrid scheme which should take some statistical methods for the analysis of sentence. Probabilistic approach is currently not integrated within the parser since we don't have a Sanskrit corpus to work with, but we hope that in very near future, we will be able to apply the statistical methods.

The paper is arranged as follows. Section 2 explains in a nutshell the computational processing of any Sanskrit corpus. We have codified the Nominal and Verb forms in Sanskrit in a directly computable form by the computer. Our algorithm for processing these texts and preparing Sanskrit lexicon databases are presented in section 3. The complete parser has been described in section 4. We have discussed here how we are going to do morphological analysis and hence relation analysis. Results have been enumerated in section 5. Discussion, conclusions and future work follow in section 6.

2 A STANDARD METHOD FOR ANALYZING SANSKRIT TEXT

The basic framework for analyzing the Sanskrit corpus is discussed in this section. For every word in a given sentence, machine/computer is supposed to identify the word in following structure. $< Word >< Base >< Form >< Relation >$.
The structure contains the root word (<Base>) and its form <attributes of word> and relation with the verb/action or subject of that sentence. This analogy is done so as to completely disambiguate the meaning of word in the context.

2.1 <Word>

Given a sentence, the parser identifies a singular word and processes it using the guidelines laid out in this section. If it is a compound word, then the compound word with सन्धि has to be undone. For example: नदीमागच्छत्=नदीम्+आगच्छत्.

2.2 <Base>

The base is the original, uninflected form of the word. Finite verb forms, other simple words and compound words are each indicated differently. For Simple words: The computer activates the DFA on the ISCII code (ISCII,1999) of the Sanskrit text. For compound words: The computer shows the nesting of internal and external समास using nested parentheses. Undo सन्धि changes between the component words.

2.3 <Form>

The <Form> of a word contains the information regarding declensions for nominals and state for verbs.

- For undeclined words, just write u in this column.
- For nouns, write first.m, f or n to indicate the gender, followed by a number for the case (1 through 7, or 8 for vocative), and s, d or p to indicate singular, dual or plural.
- For adjectives and pronouns, write first a, followed by the indications, as for nouns, of gender (skipping this for pronouns unmarked for gender), case and number.
- For verbs, in one column indicate the class (गण) and voice. Show the class by a number from 1 to 11. Follow this (in the same column) by '1' for parasmaipada, '2' for ätmanepada and '3' for ubhayapada. For finite verb forms, give the root. Then (in the same column) show the tense as given in Table 3. Then show the inflection in the same column, if there is one. For finite forms, show

Table 1: Codes for <Form>

pa/	passive
ca/	causative
de/	desiderative
fr/	frequentative

Table 2: Codes for Finite Forms, showing the Person and the Number

1	प्रथम पुरुष
2	मध्यम पुरुष
3	उत्तम पुरुष
s	singular
d	dual
p	plural

Table 3: Codes for Finite verb Forms, showing the Tense

pr	present
if	imperfect
iv	imperative
op	optative
ao	aorist
pe	perfect
fu	future
f2	second future
be	benedictive
co	conditional

Table 4: Codes for <Relation>

v	main verb
vs	subordinate verb
s	subject(of the sentence or a subordinate clause)
o	object(of a verb or preposition)
g	destination(gati) of a verb of motion
a	Adjective
n	Noun modifying another in apposition
d	predicate nominative
m	other modifier
p	Preposition
c	Conjunction
u	vocative, with no syntactic connection
q	quoted sentence or phrase
r	definition of a word or phrase(in a commentary)

the person and number with the codes given in Table 2. For participles, show the case and number as for nouns.

2.4 <Relation>

The relation between the different words in a sentence is worked out using the information obtained from the analysis done using the guidelines laid out in the previous subsections. First write down a period in this column followed by a number indicating the order of the word in the sentence. The words in each sentence should be numbered sequentially, even when a sentence ends before the end of a text or extends over more than one text. Then, in the same column, indicate the kind of connection the word has to the sentence, using the codes given in table 4.

Then, in the same column, give the number of the other word in the sentence to which this word is connected as modifier or otherwise. The relation set given above is not exhaustive. All the 6 karakas are defined as in relation to the verb.

3 ALGORITHM FOR SANSKRIT RULEBASE

In the section to follow in this paper, we shall explain two of the procedures/algorithms that we have developed for the computational analysis of Sanskrit. Combined with these algorithms, we have arrived at the skeletal base upon which many different modules for Sanskrit linguistic analysis such as: relations, सन्धि, समास can be worked out.

3.1 Sanskrit Rule Database

Every natural language must have a representation, which is directly computable. To achieve this we have encoded the grammatical rules and designed the syntactic structure for both the nominal and verbal words in Sanskrit. Let us illustrate this structure for both the nouns and the verbs with an example each .

Noun:-Any noun has three genders: Masculine,Feminine and Neuter. So also the noun has three numbers: Singular, Dual and Plural. Again there exists eight classification in each number: Nominative, Accusative, Imperative, Dative, Ablative, Genitive, Locative and Vocative. Interestingly these express nearly all the relations between words in a sentence .

In Sanskrit language, every noun is deflected following a general rule based on the ending alphabet such as अकारान्त. For example, राम is in class अकारान्त which ends with अ(a). Such classifications are given in Table 5. Each of these have different inflections depending upon which gender they correspond to. Thus अकारान्त has different masculine and neuter declensions, आकारान्त has masculine and feminine declensions, इकारान्त has

masculine, feminine and neuter declensions. We have then encoded each of the declensions into ISCII code, so that it can be easily computable in the computer using the algorithm that we have developed for the linguistic analysis of any word .

Table 5: attributes of the declension for noun

$Class^{*}$		$Case^{\eta}$	$Gender^{\zeta}$
अकारान्त(1)	दकारान्त(14)	कर्त्ता(1)	पुल्लिङ(1)
आकारान्त(2)	धकारान्त(15)	कर्म(2)	स्त्रीलिङ(2)
इकारान्त(3)	नकारान्त(16)	करण(3)	नपुंसकलिङ(3)
ईकारान्त(4)	नकारान्त(17^{1})	सम्प्रदान(4)	$Number^{@}$
उकारान्त(5)	पकारान्त(18)	अपादान(5)	एकवचन(1)
ऊकारान्त(6)	भकारान्त(19)	सम्बन्ध(6)	द्विवचन(2)
ऋकारान्त(7)	रकारान्त(20)	अधिकरण(7)	बहुवचन(3)
ऐकारान्त(8)	वकारान्त(21)	सम्बोधन(8)	
ओकारान्त(9)	शकारान्त(22)		
औकारान्त(10)	षकारान्त(23)		
चकारान्त(11)	सकारान्त(24)		
जकारान्त(12)	हकारान्त(25)		
तकारान्त(13)			

Let us illustrate this structure for the noun with an example . For अकारान्त, masculine, nominative, singular declension:

This is encoded in the following syntax: $(163\{1^{*}, 1^{\eta}, 1^{\zeta}, 1^{@}\})$.

Where 163 is the ISCII code of the declension (Table 6). The four 1's in the curly brackets represent Class, Case, Gender and Number respectively (Table 5) .

Table 6: Noun example

अ Masculine	Singular(एकवचन)	
	Endings	ISCII Code
Nominative	:	163

Pronouns:-According to Paninian grammar and Kale, (Kale) Sanskrit has 35 pronouns which are: सर्व, विश्व, उभ, उभय, इतर, इतम, अन्य, अन्यतर, इतर, त्वत्, त्व, नेम, यम, यिम, पूर्व, पर, अवर, दक्षिण, उपर, अधर, स्व, अन्तर, त्यद्, एतद्, इतद्, अदस्, एक, द्वि, युष्मद, भवत् and किम् .

We have classified each of these pronouns into 9 classes: Personal, Demonstrative, Relative, Indefinitive, Correlative, Reciprocal and Possessive. Each of these pronouns have different inflectional forms arising from different declensions of the masculine and feminine form. We have codified the pronouns in a form similar to that of nouns .

Adjectives:- Adjectives are dealt in the same manner as nouns. The repetition of the linguistic morphology is avoided .

Verbs:- A Verb in a sentence in Sanskrit expresses an action that is enhanced by a set of auxiliaries"; these auxiliaries being the nominals that have been discussed previously .

The meaning of the verb is said to be both *vyapara* (action, activity, cause), and *phala* (fruit, result, effect). Syntactically, its meaning is invariably linked with the meaning of the verb "to do". In our analysis of Verbs, we have found that they are classified into 11 classes(गण, Table 7). While coding the endings, each class is subdivided according to "इट्" knowledge, सेट्, अनिट् and वेट्; each of which is again sub-classified as into 3 sub-classes as आत्मनेपद, परस्मैपद andउभयपद, which we have denoted as pada. Each verb subclass again has 10 *lakaaras*, which is used to express the tense of the action. Again, depending upon the form of the sentence, again a division of form as कर्त्तवाच्य, कर्मवाच्य and भाववाच्य has been done. This classification has been referred to as voice. This structure has been explained in Table 7.

Table 7: attributes of the declension for verb

$Class^{*}$	it^{γ}	$pada^{\eta}$	$Tense^{\zeta}$
भ्वादिगण(1)	सेट्(1)	आत्मनेपद(1)	लट्(1)
अदादिगण(2)	अनिट्(2)	परस्मैपद(2)	लङ(2)
दिवादिगण(3)	वेट्(3)	उभयपद(3)	लृट(3)
स्वादिगण(4)			लोट्(4)
तुदादिगण(5)			विधिलिङ(5)
रुढादिगण(6)			आर्शीलिङ(6)
तनादिगण(7)			लिट्(7)
क्रह्यादिगण(8)			लुट्(8)
चुरादिगण(9)			लुङ(9)
जुहोत्यादिगण(10)			लृङ(10)
कण्ड्वादिगण(11)			

$Voice^{\lambda}$	$Person^{@}$	$Number^{\delta}$
कर्त्तवाच्य(1)	प्रथम पुरुष(1)	एकवचन(1)
कर्मवाच्य(2)	मध्यम पुरुष(2)	द्विवचन(2)
भाववाच्य(3)	उत्तम पुरुष(3)	बहुवचन(3)

.

Let us express the structure via an example for भ्वादिगण, परस्मैपद, Present Tense, First person,

Singular. This is encoded in the following syntax: $(219(194\{1^{*}, 1^{\gamma}, 2^{\eta}, 1^{\varsigma}, 1^{\lambda}, 1^{@}, 1^{\delta}\}))$.
Where 219194 is the ISCII code of the endings (Table 8). The numbers in curly brackets represent class, "it", pada, tense, voice, person and number respectively (Table 7).

Table 8: Verb example

PRESENT	Singular(एकवचन)	
	Endings	ISCII Code
First	ति	219194

Separate database files for nominals and verbs have been maintained, which can be populated as more and more Sanskrit corpsuses are mined for data. The Sanskrit rule base is prepared using the "Sanskrit Database Maker" developed during this work.

3.2 Deterministic Finite Automata: Sanskrit Rule Base

We have used deterministic finite automata (DFA) (Hopcraft, 2002) to compute the Sanskrit rule base, which we developed as described in section III A. Before we explain the DFA, let us define it. A deterministic finite automaton consists of:

1. A finite set of states, often denoted Q.
2. A finite set of input symbols, often denoted S.
3. A transition function that takes as arguments a state and input symbol and returns a state, often commonly denoted d.
4. A start state, one of the states in Q, denoted q0.
5. A set of *final* or *accepting states* F. The set F is a subset of Q.

Thus, we can define a DFA in this "five-tuple" notation: $A = (Q, S, d, q0, F)$. With this short discussion of the DFA, we shall proceed to the DFA structure for our Sanskrit Rule Base. Since we are representing any word by ISCII codes that range from 161 to 234, we have effectively 74 input states. In the notation given below, we are representing the character set by $\{C0, C1, \ldots, C73\}$, where Ci is the character corresponding to the ISCII code $161 + i$. Thus, if we define a DFA = $M(Q, S, d, q0, F)$ for our Sanskrit Rule Database, each of the DFA entities are as follows:

- Q = $\{q_0, q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{0, 1\}$. 0 represents that the state is not a final state and 1 tells that the state is a final state.
- $\sum = \{C0, C1, \ldots, C73\}$
- $\delta((q_x, a), Y) = \delta(q_Y, a)$or $\delta(q_Y, b)$ a,b $\epsilon\{0, 1\}$
- $q_0 =< q_0, 0 >$
- $F \subset \{q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{1\}$

In this work, we have made our DFA in a matrix form with each row representing the behavior of a particular state. In a given row, there are 74 columns and entries in a particular column of the corresponding row store the state we will finally move to on receiving the particular input corresponding to the column. In addition, each row carries the information whether or not it is a final state.
For example: $D[32][5] = 36$ conveys that in the DFA matrix D[i][j], in 32nd state, if input is $C5$, we will move to state no. 36 .(To be noted: $C5$ is the character corresponding to the ISCII code 166.).

In the graph below, we are giving an example how the DFA will look as a tree structure. The particular graph is constructed for the verb declensions for the class भ्वादिगण. The pada is परस्मैपद and the tense is present tense. The search in this DFA will be as follows:- If the first ending of the input corresponds to one of the state 163, 195 or 219, we will move ahead in the DFA otherwise the input is not found in this tree. On getting a match, the search will continue in the matched branch. .

In general, the search in the DFA is done as follows (We take the example of searching for भवथः in the DFA tree constructed above:-

- Firstly, an input word is given as the input to the user interface in Devanagari format .
- The word is changed to its equivalent ISCII code (203212195163 in this case).
- The automaton reads the forms in the reverse order to lemmatize them. In our DFA, we

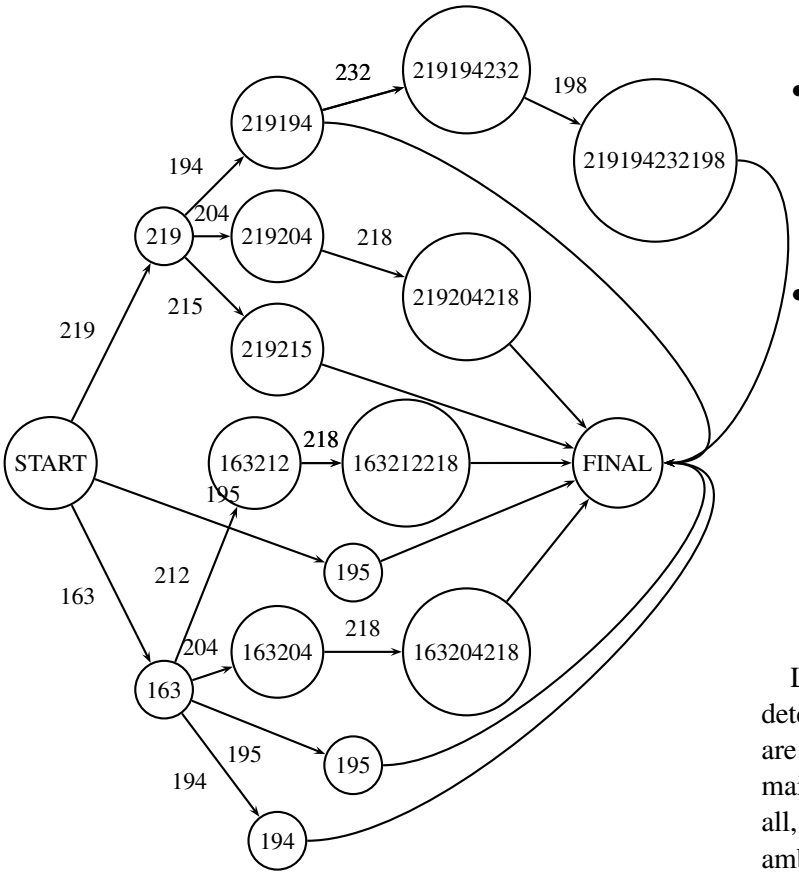

Figure 1: DFA tree obtained for भ्वादिगण परस्मैपद present tense.

give one by one the last three digits of the ISCII code till the matching is there.

- Start state: 000
- input to DFA: 163, i.e character $C2$.
- In the DFA matrix we will check the entry D[0][2]. If it is zero, no match is there for this entry and hence no match either for the word. Else we will move to the state specified by the entry.
- In this case, we get the entry corresponding to state "163". That means it is either an intermediate state or a final state. From the graph, it is visible that the tree accepts 163 just after the start state. Also, it is not a final state. Now we will have 195 (i.e. $C34$) as next input and 34^{th} column of the row corresponding to state 232 will be checked and the search continues till no match.
- Final match will be 163195.

- The final match will be checked for being eligible for a final state which is true in this case. We can verify it from the graph given.
- Remaining part of the word is sent to database engine of program to verify and to get attributes. The word corresponding to the stem, Devanagari equivalent of 203212, that is भव) will be sent to database.
- If both criteria are fulfilled (final state match and stem match through database), we will get the root word and its category (verb in this case). The attributes such as tense, form, voice, class, pada, person, number are coded in the final state itself according to the notations given in table 5 and 7. All the possible attributes are stored and it is left up to the final algorithm to come up with the most appropriate solution.

Let me just explain how we have obtained the deterministic finite automata. Clearly, the states are obtained via input symbols. Ambiguity remains in $\{0, 1\}$. If the state is not a final state at all, it is declared as intermediate state without any ambiguity to be considered for non-deterministic. When the state is a final state, for example consider गच्छति and गमिष्यति. When we encounter ति in गच्छति, we get the root as गच्छ. (Of course, we have to add the हलन्त and go through धात्वादेश getting गम् as root verb.) But in गमिष्यति, ति is not a final state. It seems at this point that we could have obtained a non-deterministic finite automaton. We have resolved the problem by accepting the following facts:

1. Final state can be intermediate state too but not the other way round.

2. Our algorithm doesn't stop just as it gets to a final state, it goes to the highest possible match, checks it for being final state and, in case it isn't, it backtracks and stops at the optimal match which satisfies the two criteria as told in the algorithm (final state match and stem match through database).

There might be another ambiguity too, for example, in रामाभ्याम्, आभ्याम् is a final state but it refers to करण, सम्प्रदान, and अपादान karaka. This seems to be non-deterministic. We have avoided this problem by suitably defining the states. Final state represents all possibilities merged in a single state. It is up to the algorithm to come up with the unique solution. There could be situation

where longest match is not the right assignment. To deal with this, all other possible solutions are also stacked and are substituted (if needed) when we go for relation analysis. For example, let us take the word सनाभ्याम्. We assume that सना, सनाभि and सनाभ्या are valid root words. Our algorithm will choose सना as root word along with the attributes (3 possibilities here). But the other solutions are also stacked in decreasing order of the match found. It is discussed in the relation analysis, how we deal with this situation.

4 ALGORITHM FOR SANSKRIT PARSER

The parser takes as input a Sanskrit sentence and using the Sanskrit Rule base from the DFA Analyzer, analyzes each word of the sentence and returns the base form of each word along with their attributes. This information is analyzed to get relations among the words in the sentence using If-Then rules and then output a complete dependency parse. The parser incorporates Panini framework of dependency structure. Due to rich case endings of Sanskrit words, we are using morphological analyzer. To demonstrate the Morphological Analyzer that we have designed for subsequent Sanskrit sentence parsing, the following resources are built:

- Nominals rule database (contains entries for nouns and pronouns declensions)
- Verb rule database (contains entries for 10 classes of verbs)
- Particle database (contains word entries)

Now using these resources, the morphological analyzer, which parses the complete sentences of the text is designed.

4.1 Morphological Analysis

In this step, the Sanskrit sentence is taken as input in Devanagari format and converted into ISCII format. Each word is then analyzed using the DFA Tree that is returned by the above block. Following along any path from start to final of this DFA tree returns us the root word of the word that we wish to analyze, along with its attributes. While evaluating the Sanskrit words in the sentence, we have followed these steps for computation:

1. First, a left-right parsing to separate out the words in the sentence is done.
2. Second, each word is checked against the Sanskrit rules base represented by the DFA trees in the following precedence order: Each word is checked first against the avavya database, next in pronoun, then verb and lastly in the noun tree.

The reason for such a precedence ordering is primarily due to the fact that *avavya* and *pronouns* are limited in number compared to the verbs, and verbs are in-turn limited compared to the infinite number of nouns that exist in Sanskrit.

4.1.1 Sandhi Module

In the analysis, we have done, the main problem was with words having external sandhi. Unless we are able to decompose the word into its constituents, we are unable to get the morph of the word. So, a rulebase sandhi analyzer is developed which works on the following principles.

- Given an input word, it checks at each junction for the possibility of sandhi.
- If it finds the junction, it breaks the word into possible parts and sends the first part in the DFA.
  - If it finds a match, it sends the second part in DFA.
    - If no match, it recursively calls the sandhi module (For the possibility of multiple sandhi in a single word).
    - If match is found, terminates and returns the words.
  - If no match, it goes to the next junction.

The rules for decomposing the words are taken from Panini grammar. The search proceeds entirely backwards on the syllabic string. Emphasis is given on minimum possible breaks of the string, avoiding overgeneration.
Panini grammar has separate sections for vowel sandhi as well as consonant sandhi. Also, there is specification of visarga sandhi. Below, we are describing the simplified rules for undoing sandhi.

Vowel Sandhi:- We have considered दीर्घ संधि, वृद्धि संधि, गुण संधि, यण संधि and अयादि संधि in vowels. (पररूप, पूर्वरूप and प्रकृतिभाव are not taken into account yet.)

1. दीर्घ संधि:- If the junction is the मात्रा corresponding to आ, ई, ऊ or ॠ, it is a candidate

for दीर्घ संधि. The algorithm for an example word भानूदय is explained.

- We assume that we don't get any match at the junction आ after भ.
- The junction ऊ is a candidate for दीर्घ संधि. So the following breaks are made: 1. भानु+ उदय, 2. भानु+ ऊदय, 3. भानू+उदय, 4. भानू+ ऊदय. For each break, the left hand word is first sent to DFA and only if it is a valid word, right word will be sent. In this case, first solution comes to be the correct one.

2. वृद्धि संधि:- In this case, the junction is ऐ, औ. The corresponding break-ups are:
   - ऐ:- (अ or आ) + (ए or ऐ).
   - औ:- (अ or आ) + (ओ or औ).

   The algorithm remains the same as told in previous case.

3. गुण संधि:- In this case, the junction is ए, ओ, अर्, अल्. The corresponding break-ups are:
   - ए:- (अ or आ) + (इ or ई).
   - ओ:- (अ or आ) + (उ or ऊ).
   - अर्:- (अ or आ) + (ऋ or ॠ).
   - अल्:- (अ or आ) + (ऌ or ॡ).

   The algorithm follows the same guidelines.

4. यण संधि:- In this case, the junction is a halanta followed by य, व, र, ल. The corresponding break-ups are:
   - halanta + य:- (इ or ई) + अ.
   - halanta + व:- (उ or ऊ) + अ.
   - halanta + र:- (ऌ or ॡ) + अ.
   - halanta + ल:- (ऋ or ॠ) + अ.

   The algorithm follows the same guidelines.

5. अयादि संधि:- In this case, the junction is अय्, आय्, अव्, आव् followed by any vowel. The corresponding break-ups are:
   - अय् + vowel:- ए + vowel. (same vowel is retained.)
   - आय् + vowel:- ऐ + vowel.
   - अव् + vowel:- ओ + vowel.
   - आव् + vowel:- औ + vowel.

   The algorithm follows the same guidelines.

Consonant Sandhi:- For dealing with consonant sandhi, we have defined some groups taking clue from panini grammar such as कु, चु, टु, तु, पु each of which have 5 consonants which are similar in the sense of place of pronunciation. Also, there is a specific significance of first, second, third etc. letter of a specific string. The following ruleset is made:

- Define string s1, with first five entries of टु and 6th entry as ष. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,
  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c \epsilon$ s1, $b \epsilon$ s2 and the position of a and b are same in the respective strings.
  For example, in the word रामष्षष्ठः, the junction is ष + halanta + ष. The break-up will be, स +halanta and ष. Hence we get रामस् + षष्ठः.

- Define string s1, with first five entries of चु and 6th entry as श. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,
  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c \epsilon$ s1, $b \epsilon$ s2 and the position of a and b are same in the respective strings.
  For example, in the word सज्जन, the junction is ज + halanta + ज. ज is the third character of string $s1$. The break-up will be, द +halanta and ज. Hence we get सद् + जन.

- We have defined strings as घोष and अघोष with अघोष containing first two characters of all the five strings कु, चु, टु, तु, पु as well as श, ष, स. घोष contains all other consonants and all the vowels. The rule says, if we get a junction with $a + halanta + c$, where a,c ϵ घोष, a will be changed to corresponding अघोष while undoing the sandhi. Similarly, other rules are made.

- The vowels are categorized into ह्रस्व and दीर्घ categories. ह्रस्व contains अ, इ, उ, ऋ and दीर्घ contains आ, ई, ऊ, ॠ. If the junction is a + न + halanta + न, where $a \epsilon$ ह्रस्व, the break-up will be: a + न + halanta and ϕ, where ϕ denotes null, i.e. other न is removed. For example, तस्मिन्नरण्ये breaks up into तस्मिन् and अरण्ये.

Visarga Sandhi:- We have looked at visarga sandhi in a single word. The rules made are as follows:

- The junction is श + halanta + a ϵ चु. The break-up will be ः and a.
- The junction is ष + halanta + a ϵ टु. The break-up will be ः and a.
- The junction is स + halanta + a ϵ तु. The break-up will be ः and a.
- The junction is र + halanta + a ϵ consonant. The break-up will be ः and a.
- The junction is र + halanta + a ϵ vowel. The break-up will be ः and a.

4.2 Relation Analysis

With the root words and the attributes for each word in hand for the previous step, we shall now endeavor to compute the relations among the words in the sentence. Using these relation values we can determine the structure of each of the sentences and thus derive the semantic net, which is the ultimate representation of the meaning of the sentence.

For computing the relations, we have employed a case-based approach i.e., nominals were classified as subject, object, instrument, recipient (beneficiary), point of separation (apaadaana) and location, to the verb based on the value of the case attribute of the word, as explained under noun example in Section 3.1.

The Sanskrit language has a dependency grammar. Hence the karaka based approach is used to obtain a dependency parse tree. There are reasons for going for dependency parse:

1. Sanskrit is free phrase order language. Hence, we need the same parse for a sentence irrespective of phrase order.
2. Once the karaka relations are obtained, it is very easy to get the actual thematic roles of the words in the sentence.

The problem comes when we have many possible karakas for a given word. We need to disambiguate between them. We have developed some If-Then rules for classifying the nouns, pronouns, verb, sub-verbs and adjectives in the sentence. The rules are as follows: First we are looking at the sentences having at least one main verb. Nominal sentences are to be dealt in the similar manner but the description will be given later.

1. If there is a single verb in the sentence, declare it as the main verb.
2. If there are more than one verb,
   (a) The verbs having suffix क्त्वा, ल्यप्, तुमुन् are declared subverbs of the nearest verb in the sentence having no such affix.
   (b) All other verbs are main verbs of the sentence and relations for all other words are given in regard to the first main verb.
3. For the nouns and pronouns, one state may have many possibilities of the cases. These ambiguities are to be resolved. The hand written rules for determining these ambiguities are as follows (Rules are written for nouns. Adjective precede nouns (May not precede too due to free word order nature.) and hence get the same case as nouns. For pronouns, rules are same as that for nouns.):
   (a) Nominative case: The assumption is that there is only one main subject in an active voice sentence. We proceed as follows:
      - All the nouns having nominal case as one of the attributes are listed. (For example, फलम् has both possibilities of being nominative or accusative case.)
      - All those connected by च are grouped together and others are kept separate. We now match each group along the following lines:
        – The number matches with that of the verb(Singular/dual/plural).
        – The root word matches with the person of the verb(i.e root word "अस्मद्" for 3rd person, "युष्मद्" for 2nd person).

      If ambiguity still remains, the one having masculine/feminine as gender is preferred for being in कर्ता karaka and declared as subject of the main verb.

      In passive voice,

- Nominative case is related to main verb as an object. After grouping and going through the match, the noun is declared as object of main verb.

(b) Accusative case: Assuming that the disambiguation for nominative case works well, there is no disambiguation left for this case. All those left with accusative case indeed belong to that. The noun is declared as object to nearest sub-verb or main verb.

(c) Instrumental case: If the sentence is in passive voice, the noun is declared as subject of the main verb.
For active voice, ambiguity remains if the number is dual. The folowing rules are used:
- We seek if the indeclinable such as सह, साकम्, सार्धम्, समम् follow the noun. In that case, noun is declared as instrument.
- If the noun is preceded by time or distance measure or is itself one of these, it is declared as instrument. For example द्वाभ्याम् दिनाभ्याम् नीरोगः जातः, here द्वाभ्याम् is the disambiguating feature.
- If बधिरः, काणः are following noun, the noun is declared as instrumental.

(d) Dative case: For dative case, disambiguity is with respect to ablative case in terms of dual and plural nnumbers The disambiguating feature used here is main verb. That is, there are certain verbs which prefer dative case and certain verbs prefer ablative. For example:
- The verbs preferring dative case are क्रुध्, द्रुह्, ईर्ष्य्, असूय्, स्पृह् etc.
- The verbs preferring ablative case are जुगुप्सा, विराम, प्रमाद, वार etc.

Initially, we have populated the list using अष्टाध्यायी knowledge as well as some grammar books but this has to be done statistically using corpus analysis.

(e) Ablative case: The ambiguity here is for certain nouns with the genitive case in singular person. The ambiguity resolution proceeds along the following lines:
- If the noun having ambiguity has a verb next to it, it will be taken as ablative (Noun with genitive case marker is not followed by a verb.)
- If suffixes तरप्, तमप् are used in the sentence, the noun is declared as ablative.
- If तुल्य, सदृश are following the noun, it is declared as genitive.
- Finally, we look for the disambiguating verbs as done in previous case.

(f) Genitive case: The ambiguity is there in dual with respec to locative case. We have used that by default, it will be genitive since we have not encountered any noun with locative case and dual in number.

(g) Locative case: The ambiguities are already resolved.

Only problematic case will be the situation discussed in section 3.2 with the example of सनाभ्याम्. If the algorithm is able to generate a parse taking the longest possible match, we will not go into stacked possibilities, but if the subject disagrres with the verb (blocking), or some other mismatch is found, we will have to go for stacked possibilities.
Thus, we have got the case markings. Relation for nominative and accusative case markings have already been defined. For other case markings,

- Instrumental: related as an instrument to main verb in certain cases (taken from अष्टाध्यायी).
- Dative: related as recipient to main verb in certain cases, but also denotes the purpose.
- Ablative: related as separation point.
- Genitive: this is not considered as karaka since karaka has been defined as one which takes role in getting the action done. Hence it is related to the word following it.
- Locative: related as location to the main verb.

Still, we have not given any relation to adjectives and adverbs. For each adjective, we track the noun it belongs to and give it the same attributes. It is defined as adjective to the noun. The adverbs are related to the verb it belongs as adverb.

Based on these relations, we can obtain a semantic net for the sentence with verb as the root node and the links between all the nodes are made corresponding to relations with the verb and interrelations obtained.

Sanskrit has a large number of sentences which are said to be nominal sentences, i.e. they don't take a verb. In Sanskrit, every simple sentence has a subject and a predicate. If the predicate is not a finite verb form, but a substantive agreeing with the subject, the sentence is a nominal sentence. In that case, the analysis that we have done above seems not to be used as it is. But in Sanskrit, there is a notion called आक्षेप, that is, if one of the verb or subject is present, other is obtained to a certain degree of definiteness. Take for example, the sentence अहम् कलमेन लिखामि। If instead of saying the full sentence, I say अहम् कलमेन, लिखामि is determined as verb. Similarly, if I say कलमेन लिखामि, the subject अहम् is determined. अहम् is a kind of appositive expression to the inflectional ending of the verb लिखामि. We have used this concept for analyzing the nominal sentences. That is, verb is determined from the subject. Mostly, the forms of अस् only are used and relations are defined with respect to that. Although, the analysis done is not exhaustive, some ruleset is built to deal with them. Most of the times, relations in a nominal sentence are indicated by pronouns, adjectives, genitive. For example, in the sentence सः सुन्दरः बालकः, there is आक्षेप of the verb अस्ति in the sentence by the subject बालकः. Hence बालकः is related to the verb as subject. सः is a pronoun referring to बालकः and सुन्दरः is an adjective referring to बालकः. Similarly, इदम् तस्य गृहम्। In this sentence, इदम् is a pronoun referring to गृहम् and तस्य is a genitive to गृहम्. Here again, there will be आक्षेप of the verb अस्ति and गृहम् will be related to the verb as subject.

5 RESULTS

5.1 Databases Developed

The following Sanskrit Rule Databases have been developed during the project:-

- Nominals (शब्दरूप) rule database contains entries for nouns and pronouns declensions along with their attributes.
- Verb (धातुरूप) rule database contains entries for 10 classes of verb along with their tenses.
- Particle (अव्यय) database.

Along with these databases, we have developed some user interfaces (GUI) to extract information from them. For example, if we want to get the forms of a particular verb in a particular tense, we can just open this GUI and give also obtained. the root word and tense information.

5.2 Parser Outputs

Currently, our parser is giving an efficient and accurate parse of Sanskrit text. Samples of four of the paragraphs which have correctly been parsed are given below along with snapshot of one sentences per paragraph.

अनुच्छेद 1:- पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति। गुरुकुले अनेकेछ्हराः सन्ति। केचन बालाः। केचन प्रौढाः। गुरुचरणाः प्रौढान् छात्रान् पाठयन्ति। प्रौढाः बालान् पाठयन्ति। सर्वे वेदमन्त्रान् विशुद्धरूपेण उच्चारयन्ति। गुरुचरणाः शिष्यान् वेदस्य अर्थम् अपि बोधयन्ति। प्रतिदिनम् प्रातः सायम् सर्वे शिष्याः यज्ञम् अपि कुर्वन्ति। ते वनम् गत्वा समिधः आनयन्ति। ते नियमेन व्यायामम् कुर्वन्ति। एते शिष्याः एव धर्मस्य रक्षणम् करिष्यन्ति।

S.no.	Word	Root	Group	Attribute	Relation
1	पूज्याः	पूज्य	Adjective	m8p m1p	1 adjective 2
2	गुरुचरणाः	गुरुचरण	Noun	m8p m1p	2 subject 12
3	रमणीये	रमणीय	Adjective	n8d n7s n2d n	3 adjective 4
4	गुरुकुले	गुरुकुल	Noun	n8d n7s n2d n	4 location 12
5	प्रातः	प्रातः	Avyaya		5 adverb 8
6	नियमेन	नियमेन	Avyaya		6 adverb 8
7	सन्ध्याम्	सन्ध्या	Noun	f2s	7 object 8
8	उपास्य	आस्	sub-verb	उप,क्त्वा	8 sub-verb 12
9	मेधाविनः	मेधाविन्	Adjective	m8p m6s m5s	9 adjective 10
10	शिष्यान्	शिष्य	Noun	m2p	10 object 12
11	सम्यक्	सम्यक्	Avyaya		11 adverb 12
12	अध्यापयन्ति	इंड	Verb	pr1p pr1p pr1p	12 verb 12

Figure 2: Parser output for पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति।

अनुच्छेद 2:- मम गृहे एकम् उपवनम् वर्तते। उपवनस्य समीपे एका गोशाला वर्तते। अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि। तत्र मम गौः माम् प्रतीक्षते। मम गोः नाम धवला अस्ति। धवलायाः वत्सायाः नाम गौरी अस्ति। अहम् धवलाम् प्रेम्णा स्पृशामि। तस्याः सास्नायाम्

कण्डूये। सा प्रसन्ना भवति। गौरीम् अपि कराभ्याम् संवाहयामि। तत्पश्चात् गोशालाम् मार्जयामि। गोमयम्, भुक्तावशिष्टानि तृणानि च निरस्यामि। ततः मम धवलायै सुस्वादूनि कोमलानि तृणानि ददे। अनन्तरम् गौरीम् मुञ्चामि। सा झटिति स्वमातरम् धवलाम् धयते। धवलायाः ऊधसि क्षीरम् अवतरति। तत्पश्चात् अहम् धवलाम् दुग्धम् दुहे। एतत् मम नित्यकर्मः। गोसेवाकाले अहम् गोमहिमप्रतिपादकान् वेदमन्त्रान् मन्दम् मन्दम् मधुरस्वरेण उच्चारयामि।

S.no.	Word	Root	Group	Attribute	Relation
1	अहम्	अस्मद्	Pronoun	g1s	1 subject 11
2	प्रतिदिनम्	प्रतिदिनम्	Avyaya		2 adverb 7
3	प्रातः	प्रातः	Avyaya		3 adverb 7
4	सूर्योदयात्	सूर्योदय	Noun	m5s	4 separation 5
5	पूर्वम्	पूर्व	Pronoun	n2s m2s	5 object 7
6	शय्याम्	शय्या	Noun	f2s	6 object 7
7	त्यक्त्वा	त्यज्	sub-verb	N,क्त्वा	7 sub-verb 11
8	शौचादिकम्	शौचादिक	Noun	n2s n1s	8 object 9
9	विधाय	धा	sub-verb	वि,क्त्वा	9 sub-verb 11
10	गोशालाम्	गोशाला	Noun	f2s	10 object 11
11	गच्छामि	गम्	Verb	pr3s pr3s pr3s	11 verb 11

Figure 3: Parser output for अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि।

अनुच्छेद 3:- मम गृहे एकम् उपवनम् वर्तते। उपवनम् विशालम् अस्ति। अस्मिन् उपवने नाना फलवृक्षाः सन्ति। बहवः पुष्पतरवः अपि वर्तन्ते। अनेकाः लताः अपि विद्यन्ते। आयुर्वैदिक वनस्पतयः अपि सन्ति। अहम् प्रतिदिनम् सायंकाले उद्यानकर्म कुर्वे। अहम् हानिकराणि तृणानि निराकरोमि। अहम् वारिणा सर्वान् वनस्पतीन् सिञ्चामि। सायंकाले उपवने मम पितृचरणाः भ्रमन्ति। तैः साकम् मम भ्रातृजाः अपि भ्रमन्ति। कदाचित् गौरी अपि तत्र आयाति। सः इतस्ततः वेगेन धावति। शाद्वले हरिततृणानि चरति। मयूराः अपि तत्र आगत्य नृत्यन्ति। ते मधुरम् ध्वनिम् कुर्वते। वृक्षेषु वानराः क्रीडन्ति। विविधाः पक्षिणः वृक्षेषु कूजन्ति। मम मातृचरणाः सर्वेभ्यः रोटिकाखण्डानि धान्यकणान् च वितरन्ति। इदम् दृश्यम् अति आनन्दप्रदम् भवति।

अनुच्छेद 4:- मम देशस्य नाम भारतवर्षम्। मम देशे वेदानाम् अध्ययनम् भवति। वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति। वेदेषु अन्यतमः ऋग्वेदः। ऋग्वेदे एकम् वाक्यम् वर्तते,"हे मानव! त्वम् कृषिम् अवश्यम् कुरु।" सम्पूर्णे जगति जन्तूनाम् जीवनाधारः अन्नमेव।अन्नञ्च कृषिम् विना नैव सम्भवति। अतः भारतीयाः आचार्याः सर्वेषाम् जीवानाम् कल्याणाय कृषिकर्मशिक्षणाय बहुविधान् उपायान् अशिक्षयन्त।

S.no.	Word	Root	Group	Attribute	Relation
1	मम	अस्मद्	Pronoun	g6s	1 relation 2
2	मातृचरणाः	मातृचरणा	Noun	f8p f2p f1p	2 subject 7
3	सर्वेभ्यः	सर्व	Pronoun	n5p n4p m5p r	3 recipent 7
4	रोटिकाखण्डानि	रोटिकाखण्ड	Noun	n8p n2p n1p	4 object 7
5	धान्यकणान्	धान्यकण	Noun	m2p	5 object 7
6	च	च	Avyaya		6 adverb 7
7	वितरन्ति	तृ	Verb	pr1p pr1p pr1p	7 verb 7

Figure 4: Parser output for मम मातृचरणाः सर्वेभ्यः रोटिकाखन्डानि धान्यकणान् च वितरन्ति

तेषु प्रधानः उपायः गोवंशसेवा। गोः वंशे धेनवः, बलीवर्दाः वृषभाः, वत्साः, वृद्धाः धेनवः सर्वे गोवंशजाः अन्तर्भवन्ति। भूमेः बलवर्धनाय गोमूत्रम् गोमयम् च अत्यन्तम् उपकुरूतः। बलदाः भूमिम् कर्षितुम् उपयुज्यन्ते। शस्यानाम् नयनानयनाभ्याम् गोमयजन्यउर्वरकाणाम् इतरपदार्थानाम् इतस्ततः प्रापणाय शकटानाम् प्रयोगः आवश्यकः। शकटाश्च वृषभैरेव उह्यन्ते। अतः गोवंशस्य कृषेश्च अविच्छेद्य कश्चन अपूर्वः सम्बन्धः वर्तते। अतएव गोः उपमा दातुम् न शक्यते। भूमेः एकम् नाम गौः इत्यपि विद्यते। संस्कृते गौः इति पदेन स्त्री गौ पुङ्गवश्च उभावपि गृह्यते। कृषिवर्धनाय शस्यनाशकानाम् कीटानाम् नाशार्थम् केचन घातकद्रव्यानाम् प्रयोगम् कुर्वते। सामवेदस्य प्रथमे आर्चिके कृषिविषये गम्भीरा चर्चा प्राप्यते। अस्माकम् गुरवः महर्षयः सम्पूर्णं जगतः कल्याणाय कृषिविद्याम् सम्यक् प्रचारयन्। कृषिकर्मणे जनान् प्रेरितवन्तः। वेदेषु भूमिः माता इति कथ्यते। यथा माता अस्मान् पालयति तथा पृथ्वी सर्वान् जीवान् अन्नेन फलैः नानाविधपदार्थैः पालयति। यन्त्रैः, तैलेन्धनेन, रसायनद्रव्यैः कीटनाशकैः च या कृषि क्रियते तेन सर्वेषाम् जीवानाम् नाश एव भवति। अनेन कर्मणा सर्वेषाम् प्राणिनाम् मनुष्याणाञ्च महती हानि जायते।.

S.no.	Word	Root	Group	Attribute	Relation
1	वेदेषु	वेद	Noun	m7p	1 location 9
2	प्राणिनाम्	प्राणिन्	Noun	m6p	2 relation 5
3	मनुष्याणाम्	मनुष्य	Noun	m6p	3 relation 5
4	च	च	Avyaya		4 adverb 9
5	जीवनाय	जीवन	Noun	m4s	5 recipient 7
6	आवश्यकाः	आवश्यक	Adjective	m8p m1p	6 adjective 7
7	नियमाः	नियम	Noun	m8p m1p	7 subject 9
8	वर्णिताः	वर्णित	Adjective	m8p m1p	8 adjective 7
9	सन्ति	अस्	Verb	pr1p pr1p pr1p	9 verb 9

Figure 5: Parser output for वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति।

The parse results pave the way for represent-

ing the sentence in the form of a Semantic Net. We here give the semantic net for the parse output given in Figure 2.The Semantic Net is shown in Figure 6.

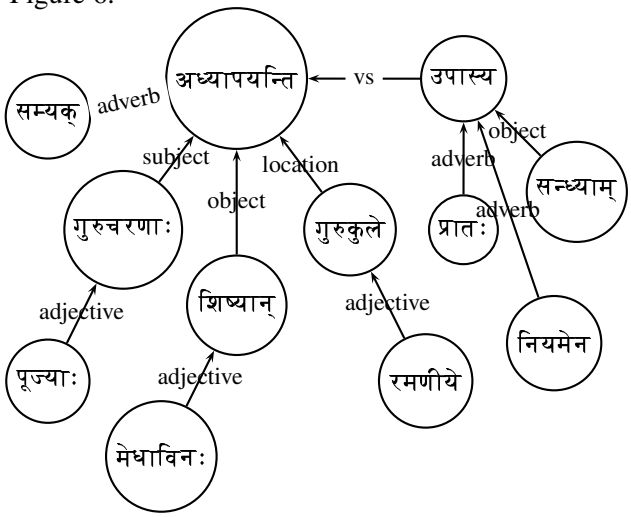

Figure 6: Semantic net representation of the sentence [पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति ॥]

6 CONCLUSIONS AND FUTURE WORK

Our parser has three parts. First part takes care of the morphology. For each word in the input sentence, a dictionary or a lexicon is to be looked up, and associated grammatical inforation is retrieved. One of the criterion to judge a morphological analyzer is its speed. We have made a linguistic generalization and declensions are given the form of DFA, thereby increasing the speed of parser. Second part of the parser deals with making "Local Word Groups". As noted by Patanjali, any practical and comprehensive grammar should be written in 'utsarga apavaada' approach. In this approach rules are arranged in several layers each forming an exception of the previous layer. We have used the 'utsarga apavaada' approach such that conflicts are potentially taken care of by declaring exceptions. Finally, words are grouped together yielding a complete parse. The significant aspect of our approach is that we do not try to get the full semantics immediately, rather it is extracted in stages depending on when it is most appropriate to do so. The results we have got are quite encouraging and we hope to analyze any Sanskrit text unambiguously.

To this end, we have successfully demonstrated the parsing of a Sanskrit Corpus employing techniques designed and developed in section 2 and 3. Our analysis of the Sanskrit sentences in the form of morphological analysis and relation analysis is based on sentences as shown in the four paragraphs in previous section. The algorithm for analyzing compound words is tested separately. Hence future works in this direction include parsing of compound sentences and incorporating Stochastic parsing. We need to take into account the नामधातु as well. We are trying to come up with a good enough lexicon so that we can work in the direction of समास विच्छेद in Sanskrit sentences. Also, we are working on giving all the rules of Panini the shape of multiple layers. In fact, many of the rules are unimplementable because they deal with intentions, desires etc. For that, we need to build an ontology schema. The Sandhi analysis is not complete and some exceptional rules are not coded. Also, not all the derivational morphology is taken care of. We have left out many प्रत्यय. Reason behind not incorporating the प्रत्यय was that it is difficult to come up with a general DFA tree for any of the प्रत्यय because of the wide number of rules applicable. For that, we need to encode the Panini grammar first.

Acknowledgment

We humbly acknowledge our gratitude to revered Aacharya Sanskritananda Hari, founder and director of Kaushalya pitham Gurukulam, Vadodara for educating us in all aspects of Sanskrit language.

References

Blai Bonet and Hctor Geffner 2001. *Planning as heuristic search*. Artificial Intelligence 129.

Ferro, M.V., Souto, D.C., Pardo, M.A.A.. 1998. *Dynamic programming as frame for efficient parsing*. Computer science, 1998.

Ivanov, Y.A., Bobick, A.F. 2000. *Recognition of visual activities and interactions by stochastic parsing*. Volume 22, Issue 8, Aug. 2000 Page(s):852 - 872. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Briggs, Rick. 1985. *Knowledge Representation in Sanskrit and artificial Intelligence*, pp 33-39. The AI Magazine.

G. Huet. 2002. *The Zen Computational Linguistics Toolkit*. ESSLLI 2002 Lectures, Trento, Italy.

G. Huet. 2005. *A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger*. Journal of Functional Programming 15 (4) pp. 573–614.

G. Huet. 2006. *Shallow syntax analysis in Sanskrit guided by semantic nets constraints*. International Workshop on Research Issues in Digital Libraries, Kolkata. Proceedings to appear as Springer-Verlag LNCS, 2007.

Bureau of Indian Standards. 1999. *ISCII: Indian Script Code for Information Interchange*. ISCII-91.

Akshar Bharati and Rajeev Sangal. 1993. *Parsing Free Word Order Languages in the Paninian Framework*. ACL93: Proc. of Annual Meeting of Association for Computational Linguistics. Association for Computational Linguistics, New Jersey, 1993a, pp. 105-111.

Kale, M.R. *A Higher Sanskrit Grammar*. 4th Ed,Motilal Banarasidass Publishers Pvt. Ltd.

Hopcroft, John E., Motwani, Rajeev, Ullman, Jeffrey D. 2002. *Introduction to Automata Theory, Languages and Computation*. 2^{nd} Ed, Pearson Education Pvt. Ltd., 2002.