Webology (ISSN: 1735-188X)
Volume 16, Number 2, 2019

Sanskrit To Hindi Translation Using Lexical Paring

Neha Sadana , Dhavleesh Rattan

[#]Computer Engineering Department, Punjabi University, Patiala-147004, Punjab, India.

Abstract Word is the basic unit of the Linguistic structure. It is the sequence of characters marked by space. In this paper we have used only the noun forms that end with vowel 'a'or akarant shabd. Noun can be classified into 2 forms subantapada and taddhipada, here pada is basically a syntactic unit. We have worked on subantapada i.e. words ending with sup suffix and also analyzed the given word into its root and feature value of the grammar like number, person and avyaya shabad. We have used some mapping rules and patterns to find out desired output.

Key Words: Avyaya, subanta, lexical analysis, parsing, tokenization.

1. INTRODUCTION

The Sanskrit language is called as 'Devbhashaa' which means the language of God. This language is written in Devanagari script [15]. Panini has created entire Sanskrit grammar, known as Astadhyayi with the help of fourteen distinctive sounds. Sanskrit grammar is perfect and this can be proved easily by the extensiveness of its grammatical tenses. In tenses there is one form of the present tense, three forms of the past tense and two forms of the future tense. It has three separate words i.e. ekvachan, dvi-vachan and bahu-vachan to represent singular plural forms of noun. It has three categories of the verbs i.e. atmaepadi, padasmaipadi and ubhaipadi. It has nearly 90 forms of one single verb. For example the word 'gach' root word (known as dhatu) means 'to go'. In Sanskrit it has 90 forms of the verb for example gachhti, gachhtah, gachhanti, etc. where as in English language it has three forms go, went, gone. To represent all three gender nouns each word has further 21 forms to cover all situation [14].

1.1 Overview of Devanagari Script:

The script used for writing classical Sanskrit is 'Devanagari' and its derivative is Hindi. The script is used to write many other languages such as Santhali, Newari, Jaipuri, Papa, Bihari, Marwari etc. Many other script are also assosciate with devanagari such as Tibetan script, Sinhala script of Sri Lanka and southeast Asian script, Brahmi script. It consists of 42 characters, 9 vowels, 33 consonents. This is written from left to right. Devanagari gives a sign known in Sanskrit as the virama or vowel omission sign (◌्). In Hindi it is specified as hal or halant. Normally its purpose is to cancel (or kill) the inherent vowel of the consonant to which it is applied. When a consonant loses its inherent vowel by the application of virama or halant, it is known as a dead consonant [18]. We need Unicode to support classical text. The Unicode Standard is a character coding system which is designed to support process, to interchange, and display the texts of diverse languages [23]. It provides a unique code/binary number for each character irrespective of platform, program and the language. Its block ranging from 0900 to 097F is used for writing characters [16]. UTF-8 encoding system assigns 1 byte for Devanagari codes [17].

1.2 Motivation for work

a) If someone wants to analyze the text of Geeta slok, one will get information like number of persons, avayaya words and their Hindi translation.

b) Upon assessing the work done in Sanskrit, we have realized the lack of work done in the area of lexical analysis. Thus we have done lexical analysis of Sanskrit text using the subject system as the Geeta slok.

2. BACKGROUND

The Merriam-Webster dictionary defines Sanskrit as an ancient Indo-Aryan language that is the classical language of India. Sanskrit saṁskṛta, is originated, from sam i.e. together + karoti i.e he makes [24]. The papers we have studied till now include Lexical analysis, Parsing**,** Tokenization and Machine Translation.

2.1 Lexical Analysis:

Lexical Analysis is the initial phase of compilation process. It scans each character of source program and produces meaningful sequence called Tokens. It is also called as scanner. It removes blanks, tabs and comments [1]. In a language, Lexicon is a container for words; it contains mostly root word instead of complete words of the sentence. It is a data structure which contains word entries and language rules [20].

2.2 Parsing

Parsing is defined as analyzing the given text, consists of sequence of tokens, its grammatical structure with respect to formal grammar is determined. It includes the process for transformation of the given input into an internal system representation, which can be graphs, tress or some other structural form. A core component necessary for parsing is the system lexicon. It stores data which lists all words known to the system, and encodes their syntactic properties [2].Each lexicon uses lexical descriptors to identify information about each word. The use of grammatical rules and other knowledge resources to specify functions of the words in the sentence is called de-linearization of the linguistic text which is also known as parsing. The parser rules are designed with the help of rule based approach. This approach uses rules and a lexicon to give an output [20]. It syntactically processes the instruction. Syntactic parsing converts the natural language strings into either tree structures or dependency links [2].

2.3 Tokenization

Tokenization is defined as splitting the character stream or text into words. A Tokenizer is a scanner which divides the text of an input document into words. The scanner is a piece of software that matches a regular expression with input stream of symbol. The set of rules are specified as regular expression. The longest of match is accepted in case of multiple overlapping substrings of the input matches. The output of first phase of compiler is stream of tokens of the given natural language [21].

2.4 Machine Translation

Natural language processing has emerged from Artificial Intelligence in order to provide linguistic definition to machine so that it can interpret the sentences which are given as input in natural language [14]. Machine Translation is defined as a process which utilizes computer software to translate one language to another and in this process it involves grammatical structure of each language and uses rules to transfer the source language (SL) to target language (TL) [19]. Various approaches have been made for Machine translation as Rule based Machine Translation (RBMT), statistical based machine translation (SBMT); Example based Machine Translation (EBMT) etc. We have used these rules for lexical analysis, followed by parsing and tokenization of text.

3. RELATED WORK

Farhanaaz and Sanju [1] explained lexical process in detail. They described lexical process as first phase of compilation process dealing with input language and also discussed multi-core environment. Their study shows Token as the logical unit, single character or sequence of characters. A finite automaton implements a regular definition to recognize Token. A Finite State Automaton is a recognizer or acceptor of Language. It is a restricted model of computers i.e. similar to CPU whose memory is absent .They have used Generalized approach to build a lexical model i.e. initially discuss the problem domain (Here problem domain is language whose lexical output has to be generated). In this lexical program reads each character from the document and analyze the character stream to distinguish words in document. It also describes Tokenization as the process in which tokens have logical relationship and collective meaning .This process needs to be paralleled to adapt multi-core environment.

Tapasawi et al. [2] discussed a set of instruction using formulation of Lexical Functional Grammar for parsing Sanskrit sentences. They described Lexical-Functional Grammar as a constraint based theory of language, whose c-structure (constituent structure) and f-structure (functional structure) is basic architecture that distinguishes two levels of syntactic representation: c-structures as a tree representation, and f-structures as an attribute value matrix. The lexical functional grammar is used to parse natural languages like Sanskrit. This paper is useful for the one who is familiar or unfamiliar with the formal structure of the grammar for construction of rules or lexical constructs of any language and they also discussed two reasons for using the LFG framework.

Tapaswi and Jain [3] described Morphological and Lexical analysis of the sentences that evolved the languages like Sanskrit which are morphologically rich Language. They developed a model that can be used for identification of the words as well as spellchecker. In this paper they discussed sandhi and algorithm for sandhi-wichad in detail and also explained lexical analyzer that is used for information retrieval which identifies the root word and its meaning.

Jha et al. [4] in this paper describes Sanskrit morphological analyzer that identifies and analyzes noun and verb form in a given Sanskrit text. It also describes the subanta, tinata, avyaya analysis of given text and also label them for POS categories. It includes the analysis of derived verb roots. The system using reverse Pāinian approach for analysis. The method to develop system creates database tables to store various morphological components of Sanskrit verbs. It has been developed as java servlet with Sanskrit data as Unicode text.

Mishra [5] has developed a model using ANN model and used rule based approach for English to Sanskrit machine translation .The ANN model gives matching of Sanskrit word equivalent of English word Also, selection of Sanskrit word like noun, verb, object, adjective is done by ANN model .This model uses rules that generate nouns and verbs to translate English sentences to Sanskrit sentences. According to author the rule based approaches make use of hand written transfer rules for translation from source language to target language. The author proposed rules for handling of non-finite verbs (gerunds, infinitives and participles of English) in system and used morphological markers to identify subject, object etc in a sentences.

Vishali and Parkash [6] have used rule based approach to develop English to Sanskrit machine translation. The system goes through many different modules such as lexical parser, semantic mapper, itranslator and composer. They compares English with Sanskrit grammar and then Constructs algorithm for semantic mapper.

Vipul [7] presented two methods for automatic segmentation of Sanskrit sentence into its valid constituents. He has used approach of Finite state transducer to analyse Sanskrit text and traverse it into a word and morph analyzer approach to generate all possible segmentations and validate it. The input to segmentizer is either a Unicode or roman transliterated string. He has used scoring matrix followed by baseline

system to prioritize various analyses. He discussed two algorithms that are to split a string into sub-strings and to insert sandhi rules in the finite state transducer.

Narmata and Suresh [8] represented modular architecture by lexical functional grammar and shown that it has strong and deep capability for parsing. They concluded that the development of ParGram grammar using LFG methodology is very powerful, versatile and effective. ParGram stands for parallel grammar. The architecture of all the ParGram grammars is same. Firstly the author split the sentence into tokens, then morphologically analyses the output of this and feed as input to the syntactic components where they interacts with syntactic rules and morphological information helps to build c-structure corresponds to tree representation i.e. hierarchical representation of words into phrases and f-structure corresponds to functional organization of the sentence representing functional relations like subjects and object as primitives, it represent dependency in the form of an attribute value matrix.

Ved et al. [9] uses Rule based approach to show knowledge representation of machine translation procedure of Sanskrit to English language. Lexemes are generated using parsing technique and these lexemes are used for translation process. Mapping rules and patterns are generated for desired output. They discussed the difference between Sanskrit and English language and also mentioned algorithm for translation process.

Narmata et al. [10] have presented architecture of the spellchecker and its algorithm using morphological rules. They applied morphological analysis to a large number of words and on the bases of analysis they developed a spell checker.

Swati et al. [11] have developed lexion parser using rule based approach for Devanagari script. It shows how a sentence is parsed into tokens and then finds the relationship between each token by using grammar and semantic relation which generates parse tree that identify grammatical meaning of the words. They implemented the Algorithm i.e. tagging and tokenization for devanagari text. An accuracy is 89.33% could be achieved on test sample of 150 sentences.

Bahadur et.al [12] discussed the structure of the Sanskrit sentences. They used Sanskrit grammar to give procedure to generate words from root words. They also compared the similarities between Sanskrit grammar and context free grammar. They discussed algorithm, database designed, developed two-way model for translation and also given technique for development of EtranS system (supports both English and Sanskrit grammar).

Rajneesh and Girish [13] present a statistical Sanskrit-Hindi Translator. The System is being developed simultaneously on the platform - the Microsoft Translator Hub (MTHub), Moses and is purposeful only for simple Sanskrit texts. The trained database set with BLEU (Bilingual Evaluation Understudy) whose score is 41 and above for 24K parallel and 25k monolingual sentences. They also discussed the errors analysis of the MTHub system and suggest possible solutions. The input and output to system is in Devnagari Unicode Script.

Mishra and Mishra [14] have presented a study of example-based English to Sanskrit machine translation. They described that example-based machine translation (EBMT) has developed as one of the most versatile, simple and accurate approaches for English to Sanskrit machine translation in comparison to rule-based machine translation (RBMT) and statistical-based machine translation (SBMT). They presented with examples the divergence between Sanskrit and English languages which can be considered to represent the divergences among the order free and subject-verb-object (SVO) classes of languages.

4. IMPLEMENTATION: LEXICAL ANALYZER FOR SANSKRIT TEXT

Sentence means a group of words which define a complete idea. Word is a basic element of sentence, having its own meaning and idea. Sometimes, the individual words cannot be taken into consideration for analyzing

the meaning of sentence, we need to analyze complete sentence in order to get the meaning. The meaning of word groups depends on the individual words and their relationship (syntactic) among words of the groups [22]. The reason is as follows:

1. Individual words of the groups (e.g. vibhakti or case in Sanskrit, preceded or succeeded with nouns).
2. For language like Hindi, verb and Tense are written as separate words बालक पढता है ।; where as in Sanskrit verb and tense will be indicated by the verb itself in a single word e.g. बालः पठति ।.

Classified view of Sanskrit words: In Sanskrit a word is encoded with following information [22]:

1. Root of the word indicating action(Dhaatu/ धातु्)
2. Tense indicating action (lakarah/ लकार्)
3. Gender, Number, Person of karta or karma (लि◌ङ्गम् / lingam)
4. Modality (active voice or passive voice) (वाचय परिवरतन)

4.1 Analysis of Sanskrit text and parsing:

- Lexical Analysis (marker, Root word)
- Syntactic Analysis (Grammar Rules)
- Context Analysis (Mapping Sanskrit to Hindi sentences)

- In the Lexical Analysis emphasis is on finding the root word. Let us consider a word which may be composed of two words and on separating these words , we get the root word e.g. परं =पर +अम्. We got its root and suffix respectively. There are Markers attached to words, to know where markers are attached in a word Let's take an example रामः (rAmaH), रामौ (ramau), रामाः (rAmAH) in these words markers are ◌ः (aH), ◌ौ (au), ◌ा◌ः (AH).
- Purpose of syntactic Analysis would be implementation of Grammar Rules. Let us consider an example where we have apply syntactic analysis on a word to find out vibhakti and person e.g. रामौ it is composed of two words राम +औ from this we can interpret it is प्रथमा/द्वितीया, द्विवचन.
- In context Analysis we find the hindi translation of Sanskrit sentence using resource sentences that are stored in text file. E.g. बालः पठति | whose mapped answer will be बालक पढता है ।

4.2 Algorithm for Sanskrit text parsing and analysis

1. Take input text
2. Tokenize input text.
3. Store all words into STRING "sanskritWords".
4. Read each word one by one from input STRING.
5. Search and compare selected word for linguistic rules.
6. If multiple rules match for the given substring, then maximum matched rule is applied.
7. Display word with associated mapped rule as an output.

4.3 Architecture of the system

In this System reads the source text i.e. given in the form of Sanskrit text followed by Tokenization, Where a Tokenizer divides the text of an input text into words. The output of first phase is parsed and given to Lexical Analyzer where it scans each character of source program and produces meaningful output by analyzing those nouns which match a particular pattern from the rule file. In Rule File it checks whether the word is avyaya

or has some defined rule and gives the output for the same. Fig. 1 shows block diagram describes the architecture of the system.

4.4 Sanskrit subanta (inflected nouns)

Sanskrit is inflected language. It has noun which is represented with seven case markers in three numbers. It can be declined in all three genders. Derived nouns are as follows:

- Primary forms known as kridanta where a suffix when attached to a verb, turns into a noun e.g. तत्र गमनं प्रचलित (there is movement happening), गम् (verbal root)+अन(suffix)=गमन (noun),अन is कत pratiya (कृत् suffix) कृत्
- Secondary forms known as taddhitānta where a suffix when attached to a noun, turns into an adjective e.g. धन+वत् = धनवत् (meaning wealthy) वत् is a taddhit suffix.
- Compound form (samāsa) where compound words become pratipadika e.g. instead of राम: लक्ष्मण: च गच्छति write रामलक्ष्मणौ गच्छत: , रामलक्ष्मणौ (dual form) , रामलक्ष्मण is a compound word and is pratipadika here.
- According to Painini, there are 21 case suffix called sup (i.e. seven vibhaktis combined with three numbers)Which can attach to base (pratipadika) according to syntactic structure, end character of the base e.g. **सुबन्त = प्रातिपदिक + सुप्** |
- It has set of three as follows shown in table 1 [4]:

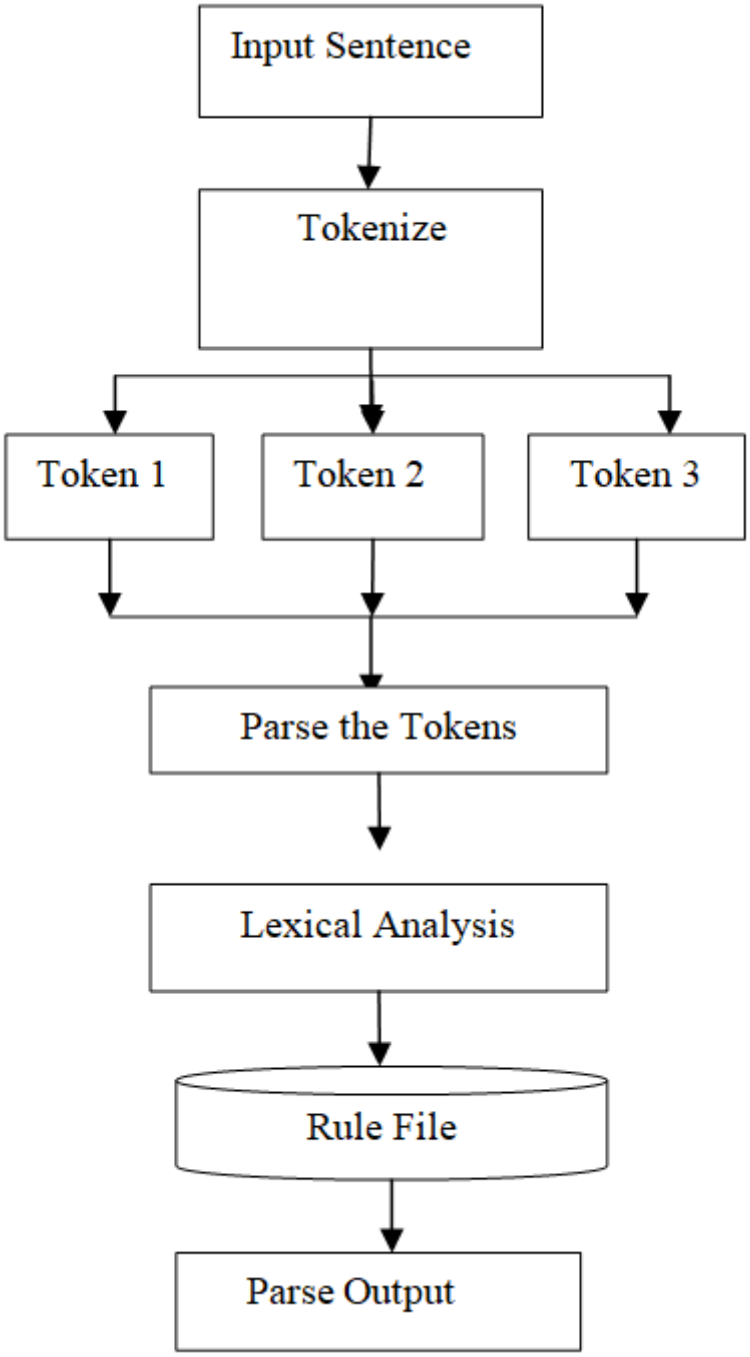

Fig.1 – Architecture of the System

Table 1: Three Forms

	एकवचनम्	द्विवचनम्	बहुवचनम्
प्रथमा	सु	औ	जश्
द्वितीया	अम्	औट	शस्
तृतीया	टा	भ्याम्	भिस्
चतुर्थी	ङे	भ्याम्	भ्यस्
पंचमी	ङसि	भ्याम्	भ्यस्
षष्ठी	ङस	ओस	आम्
सप्तमी	ङि	ओस	सुप्

4.4.1. Analysis of subanta

System does analysis of nouns using text file in a project i.e. Rule file. Brief description of this is as follows**.**

4.4.1.2 Rule File:

The subanta patterns are stored in this rule File. This File analyzes those nouns which match a particular pattern from the rule file. Initially the system recognizes vibhakti as the end characters of word For Example ':' is found in nominative singular case (1-1) like रामः श्यामः. The system isolates ':' and search for analysis in rule file. In the case of nominative and accusative dual, pratipadika forms will be 'ौ' ending for example रामौ,श्यामौ the system isolates 'ौ' and search for analysis.

The sample data is as follows:

+भ्याम् = तृतीया/चतुर्थी/पञ्चमी, द्विवचन, +ौ= प्रथमा/द्वितीया, द्विवचन, ेभ्यः=चतुर्थी पञ्चमी बहुवचन

4.5 Avyaya subanta (indeclinable nouns)

Avyaya subanta, remain unchanged under all morphological conditions [4].

4.5.1 Analysis of Avyayas

System reads the source text and takes the help of avyaya text file for analysis . If an input word is found in the Avyaya text file, it is labeled अव्यय, and excluded from the Lexical analysis as Avyaya do not change forms after subanta affixation. We have stored most Avyaya in the avyaya file.

4.6 Identification of Avyaya and subant:

In this Block diagram, it checks whether the word is avyaya or has some defined rule. If the word is avyaya then it will be treated as avyaya. If the word ends with any one of the suffix related to subanta, then it returns suffix info i.e. subanta, case and number if no suffix rule found then it checks the skiplist and control is transferred to next word. The following figure shows the process of analyzing the word.

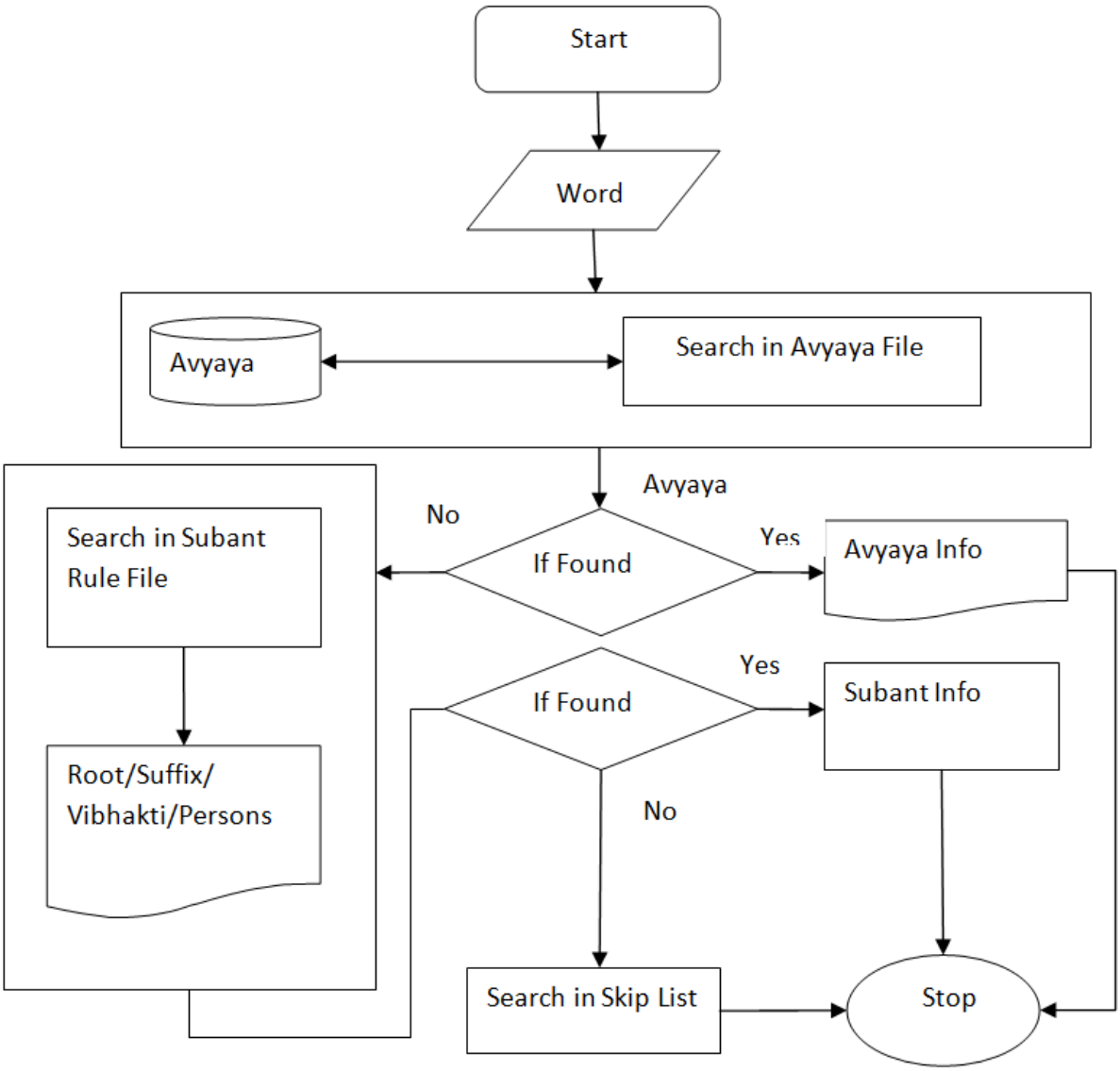

Fig. 2: Methodology

4.7 Identification of words:

In table shown below, the first field in the above table shows the root word, the second field represents inflection form, the third field indicates the suffix to be removed, and the fourth field indicates the suffix, fifth field show vibhakti, the sixth field indicates vacanam. This table represents how the rules are constructed as per the inflectional forms of word. The rules are stored in a text file. The suffix that is represented in table is extracted from the word of Sanskrit text and parse, to find desired output. Further, case, subanta and number are also obtained from a words suffix, in this table it is represented in column fourth, fifth and sixth.

The 21 inflectional forms of seven cases for word rama is shown in a systematic way in a table 2 below:

Table 2: Inflectional Forms

Word	Inflected Form	Remove	suffix	Case	Number
राम	रामः	◌ः	सु	प्रथमा (1)	एकवचन (1)
राम	रामौ	◌ौ	औ	प्रथमा/द्वितीया (1#2)	द्विवचन (2)
राम	रामाः	◌ा◌ः	जश्	प्रथमा (1)	बहुवचन (3)
राम	रामम्	म्	अम्	द्वितीया (2)	एकवचन (1)
राम	रामान्	◌ान्	शस्	द्वितीया (2)	बहुवचन (3)
राम	रामेण	◌ेण	टा	तृतीया (3)	एकवचन (1)
राम	रामाय	◌ाय	ङे॰	चतुर्थी (4)	एकवचन (1)
राम	रामात्	◌ात्	ङ॰ सि	पञ्चमी (5)	एकवचन (1)
राम	रामाभ्याम्	भ्याम्	भ्याम्	तृतीया/चतुर्थी/पञ्चमी (3#4#5)	द्विवचन (2)
राम	रामैः	◌ैः	भिस्	तृतीया (3)	बहुवचन (3)
राम	रामेभ्यः	◌ेभ्यः	भ्यस्	चतुर्थी/पञ्चमी (4#5)	बहुवचन (3)
राम	रामस्य	स्य	ङ॰ स	षष्ठी (6)	एकवचन (1)
राम	रामयोः	योः	ओस	षष्ठी/सप्तमी(6#7)	द्विवचन (2)
राम	रामाणाम्	◌ाणाम्	आम्	षष्ठी (6)	बहुवचन (3)
राम	रामे	◌े	डि॰	सप्तमी (7)	एकवचन (1)
राम	रामेषु	◌ेषु	सुप्	सप्तमी (7)	बहुवचन (3)

4.8 The results from the Lexical analyzer for the given input text fragment:

"न रूपम् अस्य इह तथा उपलभ्यते न अन्तः न च आदिः न च सम्प्रतिष्ठा । अश्वत्थम् एनम् सुविरूढमूलम् असङ्गशस्त्रेण दृढेन छित्त्वा ॥"

[न :{अव्यय } {रूपम् [रूप +(अम्, द्वितीया, एकवचन)]} {अस्य [अ +(ङ॰ स, अम्, षष्ठी, एकवचन)]} [इह :{अव्यय } [तथा :{अव्यय } {उपलभ्यते [उपलभ्यत +(डि॰ , सप्तमी, अ. नपुं प्रथमा, द्वितीया, एकवचन)]} [न :{अव्यय } {अन्तः [अन्त +(सु, प्रथमा, एकवचन)]} [न :{अव्यय } [च :{अव्यय } {आदिः [आदि +(सु, प्रथमा, एकवचन)]} [न :{अव्यय } [च :{अव्यय } {सम्प्रतिष्ठा [सम्प्रतिष्ठ +(सु, प्रथमा, एकवचन)]} {अश्वत्थम् [अश्वत्थ +(अम्, द्वितीया, एकवचन)]} {एनम् [एन +(अम्, द्वितीया, एकवचन)]} {सुविरूढमूलम् [सुविरूढमूल +(अम्, द्वितीया, एकवचन)]} {असङ्गशस्त्रेण [असङ्गशस्त्र +(टा, तृतीया, एकवचन)]} {दृढेन [दृढ +(तृतीया, एकवचन)]} [छित्त्वा :{अव्यय }

5. PERFORMANCE EVALUATION OF LEXICAL ANALYZER

In this study, we measure the efficiency of lexical parser system for Sanskrit text. In this, Geeta sloks are taken as Sanskrit text for testing purpose. When more sentences are tested, we need to add more rules then its efficiency can be increased. The system reads Sanskrit text and compares each word of sentence with avyaya

list or defined rules if suffix of word in sentence matches with defined rule the information will be displayed otherwise system will not display information.

5.1 Precision and Recall

Precision and Recall are the two measures that are widely used to evaluate machine translation system. Precision is the measure of exactness and recall is the measure of completeness. Precision describes how accurate the system is and recall tells how complete the system is.

Precision (P) = C/(C+W)

Where C = Number of Correct analysis, W = Number of Wrong analysis.

Recall (R) = C/(C+M)

C = Number of Correct analysis, M = Number of Unrecognized or missed analysis.

5.2 F-measure

It is weighted harmonic mean between precision and recall. It is weighted because in some applications one can care more about precision and recall. It is harmonic because it is conservative that is lower than arithmetic and geometric mean.

F= 2PR/ (P+R) which is equal to 2/ (1/R+1/P)

Sanskrit parser	Total No. of words Tested	Correctly Recognized Words (C)	Unrecognized words (M)	Wrongly Recognized words (W)
Lexical Analyzer	213	169	26	18

Precision=169/169+18=0.903 or 90.3%

Recall=169/169+26= 0.861 or 86.1%

F-measure= 2/ (1/0.903+1/0.861)=0.88 or 88%

Screenshot of Sample Sentences

अधः च ऊर्ध्वम् प्रसृताः तस्य शाखाः गुणप्रवृद्धाः विषयप्रवालाः । अधः च मूलानि अनुसन्ततानि कर्मानुबन्धीनि मनुष्यलोके ॥

न रूपम् अस्य इह तथा उपलभ्यते न अन्तः न च आदिः न च सम्प्रतिष्ठा । अश्वत्थम् एनम् सुविरूढमूलम् असङ्गशस्त्रेण दृढेन छित्त्वा ॥

ततः पदं तत परिमार्गितव्यम् यस्मिन् गताः न निवर्तन्ति भूयः । तम् एव च आद्यम् पुरुषम् प्रपद्ये यतः प्रवृत्तिः प्रसृता पुराणी ॥

निर्मानमोहाः जितसङ्गदोषाः अध्यात्मनित्याः विनिवृत्तकामाः । द्वन्द्वैः विमुक्ताः सुखदुःखसञ्ज्ञैः गच्छन्ति अमूढाः पदम् अव्ययम् तत् ॥

न तद् भासयते सूर्यः न शशाङ्कः न पावकः । यद् गत्वा न निवर्तन्ते धाम् परमं मम ॥

अहम् वैश्वानरः भूत्वा प्राणिनाम् देहम् आश्रितः । प्राणापानसमायुक्तः पचामि अन्नम् चतुर्विधम् ॥

सर्वस्य च अहम् हृदि सन्निविष्टः मत्तः स्मृतिः ज्ञानम् अपोहनम् च । वेदैः च सर्वैः अहम् एव वेद्यः वेदान्तकृत् वेदवित् एव च अहम् ॥

द्वौ इमौ पुरुषौ लोके क्षरः अक्षरः एव च । क्षरः सर्वाणि भूतानि कूटस्थः अक्षरः उच्यते ॥

उत्तमः पुरुषः तु अन्यः परमात्मा इति उदाहृतः । यः लोकत्रयम् आविश्य बिभर्ति अव्ययः ईश्वरः ॥

सत्त्वम् रजः तमः इति गुणाः प्रकृतिसम्भवाः । निबध्नन्ति महाबाहो देहे देहिनम् अव्ययम् ॥

तत्र सत्त्वम् निर्मलत्वात् प्रकाशकम् अनामयम् । सुखसङ्गेन तन बध्नाति ज्ञानसङ्गेन च अनघ ॥

रजः रागात्मकम् विद्धि तृष्णासङ्गासमुद्भवम् । तद् निबध्नाति कौन्तेय कर्मसङ्गेन गन देहिनम् ॥

तमः तु अज्ञानजम् विद्धि मोहनम् सर्वदेहिनाम् । प्रमादालस्यनिद्राभिः तत् निबध्नाति भारत ॥

सत्त्वम् सुखे सञ्जयति रजः कर्मणि भारत । ज्ञानम् आवृत्य तु तमः प्रमादे सञ्जयत्यति उत ॥

रामः रामौ रामाः रामं रामौ रामान् रामेण रामाभ्यां रामैः रामाय रामाभ्यां रामेभ्यः रामेभ्यः रामस्य रामयोः रामाणां रामे रामयोः रामेषु रामात्

Screenshot of Sanskrit parse sentences

6. CONCLUSION

- In this paper we analyzed and parse Sanskrit text, it shows how Sanskrit text is parsed into tokens and then finds its rule which define its subanta, case, number of person by using rule file and avyaya file. Algorithm for lexical analysis and parsing of Sanskrit text has been developed and implemented. We have also discussed block diagram of the architecture of the system and for identification of avyaya and subant. The accuracy of 79.3 was achieved.
- The lexical analyzer can be used for reading purpose to simplify the Sanskrit text. Any further processing for Sanskrit text can be done only once subanta has been analyzed. Further processing means for example kridanta and taddhita analysis, gender recognition. We research and formalized subanta rules. We studied online subanta recognition and analysis system.
- The files consisting of rules are independent of each other, it is easy to add new data or change the existing data without changing the original files. This tool is developed in eclipse (Java), JSP.

7. FUTURE SCOPE

- At present system does not give gender information for words.
- The system's accuracy can be increased by adding more rules to file.

REFERENCES:

1. Farhanaaz and V. Sanju, An Exploration on Lexical Analysis, in: Proceedings of International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016, Chennai, pp. 253-258.
2. N. Tapaswi, S. Jain, V. Chourey, Parsing Sanskrit Sentences Using Lexical Functional Grammar, in: Proceedings of International Conference on Systems and Informatics (lCSAI 2012), 2012, pp.2636-2640.
3. N. Tapswi and S. Jain, Morphological and Lexical Analysis of the Sanskrit Sentences, International Journal of Computer Science & Information Technology, 1(1) (2011) 28-31.
4. G.N Jha, M. Agrawal, Subash, S. K. Mishra, D. Mani, D. Mishra, M. Bhadra, S. K. Singh, Inflectional Morphology Analyzer for Sanskrit, in: Proceedings of Conference Sanskrit Computational Linguistics,2009,pp:219-238.

5. V.Mishra, Handling of Non-Finite Verbs in English to Sanskrit Machine Translation Nikasa, 1, (1), on 2012 pp 89-106.
6. V. Barkade and P.Devale, English to Sanskrit Machine Translation, in: Proceedings of International Journal of Engineering Science and Technology, 2(10), 2010, pp. 5313-5318.
7. V. Mittal, Automatic Sanskrit Segmentizer Using Finite State Transducers, in: Proceedings of ACL 2010 Student Research Workshop, 2010, pp 85–90.
8. N. Tapaswi, S. Jain, Knowledge Representation of Grammatical Constructs of Sanskrit Language and Modular Architecture of ParGram, in: Proceedings of International Conference on Advances in Technology and Engineering (ICATE), 2013.
9.V.K. Gupta, N. Tapaswi, S. Jain, Knowledge Representation of Grammatical Constructs of Sanskrit Language Using Rule Based Sanskrit Language to English Language Machine Translation, in: Proceedings of International Conference on Advances in Technology and Engineering (ICATE), 2013.
10. N. Tapaswi, S. Jain, V. Chourey , Morphological-based Spellchecker for Sanskrit Sentences, in: Proceedings of International Journal of Scientific & Technology Research Volume 1, Issue 3, April 2012.
11. S. Ramteke, K. Ramteke, R. Dongare, Lexicon Parser for syntactic and semantic analysis of Devanagari sentence using Hindi wordnet, Proceedings in: International Journal of Advanced Research in Computer and Communication Engineering 3(4), 2014, pp. 6345-6349.
12. P. Bahadur, A. Jain, D.S. Chauhan, Architecture of English to Sanskrit Machine Translation, in: Proceedings of Sai Intelligent System Conference, London, 2015, pp. 616-624.
13. R. K. Pandey and G. N. Jha, Error Analysis of SaHiT - a Statistical Sanskrit-Hindi Translator in: Proceedings of 20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, 2016, pp. 495-501.

14. V. Mishra and R. B. Mishra, Study of example based English to Sanskrit machine translation, in: proceedings of Journal of Research and Development in Computer Science and Engineering, No. 37, pp.43–54.

14. Sunit chand, Sanskrit as inter-lingua language in machine translation, in: Proceedings of Emerging Trends in Electrical, Communications and information Technologies, 2017, pp 27-34.
15. V. Mishra and R.B. Mishra, English to Sanskrit machine translation system: a rule-based approach, International. Journal of Advanced Intelligence Paradigms, 4(2), 2012, pp 168-184.
16. D.Mishra, K.Bali, G.N.Jha, Syllabification and Stress Assignment in Phonetic Sanskrit Text, in: Proceedings of International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian.
17. G. Huet, Formal Structure of Sanskrit Text: Requirements Analysis for a Mechanical Sanskrit Processor, in: proceedings of Sanskrit Computational Linguistics. Lecture Notes in Computer Science, 5402, 2009, Heidelberg.
18. www.unicode.org referred on June 15, 2017.
19. V.K.Gupta, N.Tapaswi, S.Jain,Knowledge Representation of Grammatical Constructs of Sanskrit Language using rule based Sanskrit language to English machine language translation, in: Proceedings International Conference of Advanced Technology and Engineering , 2003, pp:1-5.
20. S. Ramteke, K. Ramteke, R. Dongare, Lexicon Parser for syntactic and semantic analysis of Devanagari sentence using Hindi wordnet, Proceedings in: International Journal of Advanced Research in Computer and Communication Engineering 3(4), 2014, pp. 6345-6349.

21. G.U. Srikanth, Parallel Lexical Analyzer on the Cell Processor, in proceedings of Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement Companion,2010,pp 28-29.
22. M. Nandi, Ramasree RJ, Rule-based Extraction of Multi-Word Expressions for Elementary Sanskrit Texts, in: Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering – 3(11), 2013, pp. 661-667.
23. www.unicode.org referred on April 7, 2017.
24. https://www.merriam-webster.com/dictionary/Sanskrit referred on April 8, 2017.