International Journal of Engineering Research and Development
eISSN : 2278-067X, pISSN : 2278-800X, www.ijerd.com
Volume 4, Issue 8 (November 2012), PP. 33-36

Design & Analysis of an Exhaustive Algorithm for Sandhi Processing In Sanskrit

Ravi Pal[1], Dr. U. C. Jaiswal[2]
CSED, MMM Engineering College, Gorakhpur(UP), INDIA

Abstract:—It is almost impossible to learn a new language without the study of it's grammar .Automated language processing is in real centrally focused to drive to enable facilitated referencing of increasingly available Sanskrit E-texts. For learning Sanskrit language , the study of it's grammar plays a very important role .Proposed research paper presents a fresh and new approach to processing Sandhi-s, in terms of an exhaustive algorithm using morphological analysis of Sanskrit language words. This new exhaustive algorithm is based on panini's complex codifications rules of grammar. The algorithm has simple beginning and is yet powerful, much comprehensive and more computationally lean.
Keywords:—NLP, Automated language processing, Panini's rules of Sanskrit, MT System, UTF-8.

I. INTRODUCTION

The recognition of Sanskrit language as a highly phonetic language as also one with an extensively codified grammar is widespread. The very name *Samskṛt (Sanskrit)* basically means "*language brought to formal perfection*" [1]. That is the Backus- Naur Form [2] used in the specification of formal languages, has now come to be popularly known as the Paṇini's - Backus Form [2], bears an ample testimony to this fact. Sanskrit E-texts are now being increasingly made available for reference in repositories such as the Gottingen Register of Electronic Texts in Indian Languages (GRETIL) [3].Now the essential first step towards language processing of such Sanskrit E-texts is to develop exhaustive algorithms and efficient tools to handle segmentation in Sanskrit compound words that are an integral part of Sanskrit texts. And that is why this firstly necessitates the processing of *sandhi*-s in Sanskrit language.
In Sanskrit language, mainly we have two categories of complex words. They are:
(i) Sandhi (ii) Samaas

1.1 Sandhi:

When two words are combined to produce a new word whose point of combination is result of annihilation of case-end of former word and case-begin of latter, is known as sandhi and process as sandhi-formation. In short, the resulted new character that has been created at the point of combination is exactly equivalent to the sound produced when those two words are uttered without a pause. And the inverse procedure to Sandhi-formation is known as Sandhi - Wicched. e.g., ***go + yam = gavyam , sakhe + iha = sakhayiha*** etc.

1.2 Samaas:

When two or more words are combined, based on their semantics then and only then the resulting word is known as Samaas or Compound. e.g., ***(pANI ca, pADam) → (pANIpAdam), (dukham, atItah) → (dukhatItah)*** etc.. Unlike Sandhi, the point of combination in Samaas may or may not be a deformed in the resulting word. And the inverse procedure of break-up of a Samaas is known as Samaas-Vigraha. Considering the complexity of this problem, we restricted our focus to only Sandhi-s.

Rest of the paper is organized as follows : Unicode representation is described in section 2, section 3 focuses on the basis of the work, need of the sandhi processing algorithm is described in section 4, mahesvara sutra's are given in section 5, problem statement is given in section 6, approach used for designing algorithm is elaborated in section 7, proposed algorithm is given in section 8, snapshots of results are in section 9 and finally conclusion and future scope is discussed in section 10 after that references cited are given.

II. UNICODE REPRESENTATION

The Unicode (UTF-8) standard [4], is what has been adopted universally for the purpose of encoding Indian language texts into digital format. The Unicode Consortium for text formats has assigned the Unicode hexadecimal range 0900 - 097F for Sanskrit characters. All characters including the diacritical characters used to represent Sanskrit letters in E-texts are found dispersed across the Basic Latin (0000-007F), Latin-1 Supplement (0080-00FF), Latin Extended-A (0100-017F) and Latin Extended Additional (1E00 – 1EFF) Unicode ranges. In this work the Latin character set is being used to represent Sanskrit letters as E-text.

III. THE BASIS OF THE WORK

The great ***"Paṇini"*** [5], the sage and scholar dated by historians in the fourth century BC or earlier, codified the rules of the Sanskrit language mainly based on both the extant vast literature as well as the language in prevalent use at the time. His magnum opus, the ***"Aṣṭadhyayi"*** [5], which literally means for ***'work in eight chapters'***, is regarded by all scholars of sanskrit as the ultimate authority on Sanskrit grammar. In four parts each, these eight chapters mainly comprise nearly four thousand ***sutra-s*** [5] or aphorisms, terse statements in Sanskrit. This grammar codification of *Paṇini* is perhaps unparalleled, for it is terse and yet comprehensive, complex yet precise. Intensive study of the matter, taking recourse to authoritative commentaries authored by adroit grammarians, is required intense to get a grasp of the work. Many commentaries on the '*Aṣṭadhyayi'*, such as Sage ***"Patanjali's Mahabhaṣya"*** [6], are available and held as authentic and comprehensive. One such authoritative commentary with a neat, topic-wise classification of ***'Paṇini's aphorisms'*** [5], is the ***"Siddhanta-kaumudi"*** [6], written in the 70's century by the Sanskrit grammarian, ***"Bhaṭṭoji Dikṣita"*** [6]. The most important of these aphorisms were later extracted and compiled into the ***"Laghu-siddhanta-kaumudi"*** by the scholar ***"Varadaraja"*** [6]. It is accepted among Sanskrit scholars that any exploratory work on Sanskrit grammar must necessarily have the aphorisms of *'Paṇini'* as its basis, alternatively or optionally taking recourse to any of the authoritative commentaries. Our work on Sandhi-s is being also based directly on *'Paṇini's aphorisms'*, and not on secondary or tertiary sources of information. The '*Siddhanta-kaumudi'* of *'Bhaṭṭoji Dikṣita'*, famed and accepted amongst scholars as an unabridged source. comprehensive compendium of the entire *'Aṣṭadhyayi'*, has been studied in the original Sanskrit, and the Sandhi-s dealt with in it form the basis of our work.

IV. NEED FOR THE SANDHI PROCESSING ALGORITHM

Sandhi processing algorithm will be a very important component in any NLP system that attempts to analyze and understand Sanskrit for computational purposes. As we know that in the architecture of a computational Sanskrit platform, various linguistic resources such as *'lexicon', 'POS Tagger', 'karaka analyzer', 'subanta analyzer', 'tinanta analyzer', 'linga analyzer', 'sandhi analyzer',and 'samaasa analyzer'* etc. will be needed. All these resources will be interlinked yet ***sandhi analyzer*** [7] is a pre-requisite for analyzing a Sanskrit text because words in Sanskrit language are generally written with no explicit boundaries. This sandhi processing algorithm will be useful in many ways, as Sanskrit has a vast knowledge reserve of diverse disciplines. To make this knowledge reserve available to the users of other languages, an automatic MTS [7] from Sanskrit language to other languages will have to be developed. '*Sandhi Analyzer'* will be an essential initial step for this work. The other applications of this segmented form of Sanskrit text may be in building a search algorithm and spell checker for Sanskrit corpora. A '*sandhi-aware*' system thus will not only be essential for any larger Sanskrit NLP system, but will also be very helpful for self reading and understanding of Sanskrit texts by those readers who do not know or want to go through the rigors of '*sandhi – formation'*. It will also be very much helpful for interpretation and simplification of Sanskrit text. Any NLP system or NL like Sanskrit compiler will have '*sandhi Analyzer'* as a necessary initial component.

V. THE *MAHESVARA SUTRA'S OR* APHORISMS

The backbone of *'Paṇini's code',* The ***'Mahesvara aphorisms'*** [5], said to have come from the beats of a special drum type instrument called '*ḍamaru*' (hourglass drum) held in the hand of ***'Lord Mahesvara'*** (a form of God in the Hindu pantheon), are a set of aphorisms containing the letters of the Sanskrit language alphabets in a certain sequence. These aphorisms form the basis of Paṇini's composition of his grammar aphorisms. The *'Mahesvara aphorisms'* are fourteen in number and are listed below:

1. *a-i-u-ṇ*
2. *ṛ-ḷ-k*
3. *e-o-n*
4. *ai-au-c*
5. *ha-ya-va-ra-ṭ*
6. *la-n*
7. *na-ma-na-ṇa-na-m*
8. *jha-bha-n*
9. *gha-ḍha-dha-ṣ*
10. *ja-ba-ga-ḍa-da-s*
11. *kha-pha-cha-ṭha-tha-ca-ṭa-ta-v*
12. *ka-pa-y*
13. s*a-ṣa-sa-r*
14. *ha-l*

It must be noted here that the last letter in each of the above aphorisms is only a place-holder and is not counted as an actual letter of the aphorism. Here the first four aphorisms list the short forms of all the vowels, while the rest list the consonants. It must be also noted that the letter '*a*' added to each of the consonants is only to facilitate pronunciation and is not part of the consonant proper.

VI. THE PROBLEM

Sandhi-s in Sanskrit are points in words or between words, at which adjacent letters coalesce and transform. This is a common interesting feature of Indian languages and is particularly elaborately dealt with and used in Sanskrit language. The transformations that apply are commonly categorized into four categories as follows [5]:

1. *agama* – addition of an extra letter or set of letters

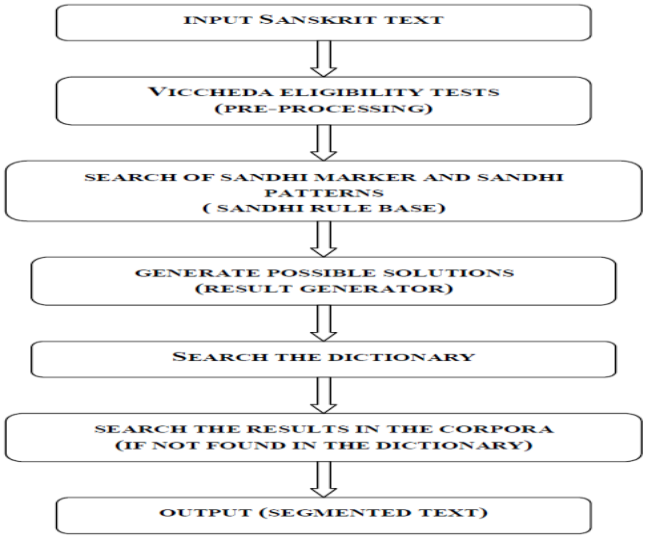
2. *adesa* – substitution of one or more of the letters

3. *lopa* – dropping of a letter
4. *prakṛtibhava* – no change

There are near about seventy '*aphorisms of Paṇini'* that deal with *sandhi-s*. These aphorisms lay out the rules for the above transformations, giving the conditions under which certain letters combine with certain others to give particular results.

The challenge is to develop an Exhaustive Algorithm to handle the entire range of *sandhi-s*. Such an exhaustive algorithm would be useful to generate various word forms of a given Sanskrit language word through the application of *sandhi rules*. Though this task is not difficult for a scholar of Sanskrit language study with a thorough knowledge of the '*Paṇinian system'* [5], it is surly a computationally non-trivial task, given the complexity and number of rules. Existing methods of *sandhi processing*, be they methods to form compound words or even to try to split them, seem to be based on a derived understanding of the functioning of Sandhi-s, and usually go the **finite automata** [8]. However, the present work directly codifies *Paṇini's rules* as is, recognizing that Paṇini's codification of the grammar is based on the '*Mahesvara aphorisms'* that in turn lay out the letters of the alphabet in a non-trivial order. This work presents one novel method of directly representing *'Paṇini's sandhi rules'*.

VII. THE APPROACH FOR DESIGNING ALGORITHM



VIII. PROPOSED ALGORITHM

1. take a Sanskrit word input
   { str(len) = sans_wrd_1 }
2. take another Sanskrit word as input
   { str(len) = sans_wrd_2 }
3. now we have two words : left, right
   { left = sans_wrd_1 & right = sans_wrd_2 }
4. set a variable flag = false
   { flag = 0 }
5. read a rule from the rule base named as suffix of left and prefix of right .
   { lft = suff_left & rgt = prfix_right }
6. try applying each of the sandhi rules on left and right.
7. if one or more rules listed in the corresponding rulebase is matched then set flag = true
   { if (match(str) =1) do flag =1 }
8. display the resultant word formed as the compound word with comments.

IX. RESULTS SNAPSHOTS

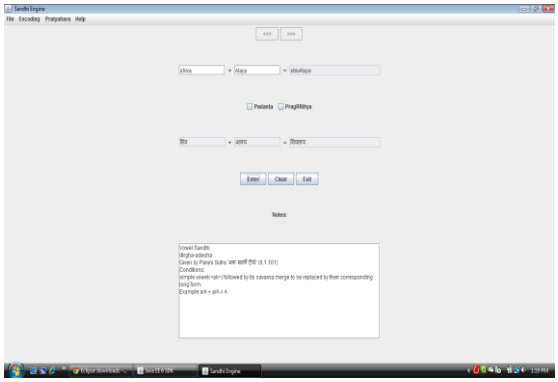


Figure 1: *Snapshot for voval sandhi*

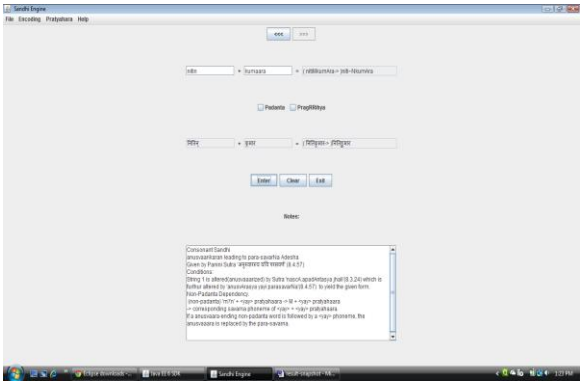


Figure 2: *Snapshot for consonant sandhi*

X. CONCLUSION AND FUTURE SCOPE

In this paper we have focused only on sandhi-formation and presented a workable solution in terms of an exhaustive algorithm.In the machine translation of Sanskrit language words we need to perform sandhi formation of the input words that are given to us. Thus at the very first stage of morphological parsing, there is a need for us to consider the sandhi formation of words. This way the proposed paper presented helps in fulfilling one of the very basic needs of the Sanskrit translators. The algorithm designed has been generalized to handle any number of possible subwords for particular inputs. The algorithm described may be enhanced to take care of all possible types of Sandhi-s by elaborating the rules given for the Sandhi Rule.

REFERENCESS

[1]. **Higher Sanskrit Grammar**,By M. R. Kale, Motilal Banarasi Dass Publishers.

[2]. **The Panini-Backus Form in Syntax of Formal Languages,** By Rao T. R. N., Kak Subhash, Center for Advanced Computer Studies, University of Southwestern Louisiana, 1998.

[3]. **Gottingen Register of Electronic Texts in Indian Languages (GRETIL)**, www.sub.uni-goettingen.de/ebene_1 /fiindolo/gretil.htm.

[4]. **UTF-8 encoding table and Unicode characters**, http://www.utf8-chartable.de/

[5]. ***Laghu-siddhanta-kaumudi***, By Varadarāja, Translated with commentary by Panḍit Visvanatha Sastri Prabhakara, Motilal Banarsidas Publishers, Delhi,1989.

[6]. ***Siddhanta-kaumudi***, By Diksita Bhattoji, Translated by Srisa Candra Vasu, Volume 1, Motilal Banarsidas Publishers, Delhi, 1962.

[7]. **Developing a Sanskrit Analysis System for Machine Translation**, By Girish N. Jha, Sudhir K. Mishra, R. Chandrashekar,Priti Bhowmik, ,Special Center for Sanskrit Studies,Jawaharlal Nehru University, New Delhi, http://sanskrit.jnu.ac.in/subanta/Paper/Kerala.PDF

[8]. **From Pāninian Sandhi to Finite State Calculus**, By Hyman Malcolm D., Sanskrit Computational Linguistics: First and Second International Symposia, Revised Selected and Invited Papers, ISBN:978-3-642-00154-3, Springer-Verlag, 2009

Ravi Pal obtained B. Tech in Information Technology (**Gold Medalist**) from Dr Ram Manohar Lohia Avadh University,Faizabad (UP) in 2010.Presently, doing M.Tech(CSE) from MMM Engineering College, Gorakhpur .He has abiding passion for teaching and has interest to teach a number of courses.mail at: raviratnakerpal@gmail.com

Dr Umesh Chandra Jaiswal is working as Associate Professor in the Department of Computer Science and Engineering, MMM Engineering College, Gorakhpur, UP(INDIA). His area of interest is Natural Language Processing, Design and Analysis of Algorithms, Operating Systems, and advanced Computing. He has published good number of papers in various journals, international and national conferences.mail at ucj_jaiswal@yahoo.com