Designing a Sanskrit Sandhi Splitter

Major Project Thesis

Supervisors:

Dr. Rahul Garg and **Dr. Sumeet Agarwal**

Submitted by:

Shubham Bhardwaj

(2012EE10480)

CERTIFICATES

By Student:

I am submitting this report dealing with work done during 1st semester of the academic session 2015-2016. I have written this report and all the material taken from other sources has been fully acknowledged. This report accurately reflects all work done by me.

Shubham Bhardwaj

2012EE10480

18-11-2015

By Supervisors:

Shubham Bhardwaj has worked under our supervision during the 1st semester of the academic session 2015-2016. We have read his report. It meets our expectations and accurately reflects the work done by him.

Dr. Rahul Garg

Dr. Sumeet Agarwal

18-11-2015

ACKNOWLEDGEMENT

I wish to express my gratitude to Dr. Rahul Garg and Dr. Sumeet Agarwal for giving me an opportunity to work in the field of Sanskrit Computational Linguistics and for their guidance and support during the whole project. The interest and enthusiasm they displayed in this field was a constant motivating factor throughout the project.

I would also like to thank Dr. Girish Nath Jha, Professor, Computational Linguistics, Special Centre for Sanskrit Studies, JNU for his valuable inputs in the project. I have met him in person and remained in touch with him during the entire semester.

I would also like to place on record the contribution of Ms. Amba Kulkarni, Associate Professor, Department of Sanskrit Studies, University of Hyderabad, with whom I remained in constant touch during the entire semester. She also took great interest in my project and was always quick to help.

My sincere gratitude to Mr. Saurabh Gupta, San Francisco, US. He is a friend of Dr. Rahul Garg and helped me learn web interfacing, which was of great use to me during the project.

I would also like to thank Dr. Samar Husain and Dr. Paroma Sanyal, faculty in HSS Department and also members of Computational Linguistics Discussion Group, IIT Delhi for the valuable inputs and suggestions they provided me on this project.

Last, but definitely not the least, I would also like to thank the members of the evaluation committee for the encouragement they provided to me during the mid-term evaluation process. This also motivated me a lot to put in greater efforts in the project.

Shubham Bhardwaj

2012EE10480

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Motivation

Sanskrit is one of the most ancient languages of India. Many modern Indian languages are considered to have originated from Sanskrit. Even a language like Tamil which is sometimes not considered as a descendant of Sanskrit contains a large number of words used in Sanskrit.

Since Sanskrit was the language spoken in ancient India, a very large number of our ancient texts are in Sanskrit. There is thus no dearth of literature in the language. But the use of Sanskrit declined with time, because of various factors, and it ultimately became a so-called dead language, a language which was spoken no more and produced no more literature.

The Sanskrit used in the Vedas (Vedic Sanskrit) is different from the one in later texts, for example, in Srimad Bhagvad Gita. Vedic Sanskrit underwent many changes with time and the new form of the language (known as Classical Sanskrit) was standardised and codified by the ancient grammarian Paa.nini in a collection of eight books known as A.s.taadhyaayii. This made Sanskrit perhaps the most well codified language in the world and it continues to be so. This work has recently attracted attention from people around the world, and a lot of work is going in understanding and implementing the formulations as presented in A.s.taadhyaayii. Moreover, it has also been argued that Sanskrit has certain features which can make it serve as an Artificial language.

Now, that the interest in Sanskrit has resumed, efforts are also being made to understand the vast literature available in the language. Sanskrit parsers are also being developed for the same purpose. The objective of these parsers is to take Sanskrit texts as input and assign them a structure, which makes it easier to comprehend them.

The form in which a word occurs in a Sanskrit text is decided by the associated gender, number, person, tense, case, etc. (different categories applicable depending on the part of speech a word belongs to), and thus parsing of Sanskrit cannot be done without analysing the words.

This necessitates the use of a morphological analyser. A morphological analyser is expected to output the different attributes (as mentioned earlier) of a given word, so that a sentence may be parsed.

But Sanskrit texts contain many words, which are formed by the combination of two or more words, through a process known as Sandhi. Such words cannot be analysed unless the component words are extracted, and this necessitates the use of a Sandhi analyser/splitter. Thus, as a morphological analyser is required for the development of a Sanskrit parser, a sandhi splitter is required for the development of a morphological analyser. Thus, a sandhi splitter is crucial for the development of a Sanskrit parser.

I formally present the concept of Sandhi and its types next.

1.2 Sandhi

Sandhi is the process by which sound changes take place when two letters combine.

Examples:

1 तस्मै + एतत् -> तस्मायेतत्

2. उप + ऋच्छति -> उपार्च्छति

1.3 When does Sandhi take place?

Sandhi takes place when two letters stand in close proximity with each other. Paa.nini does not use the word 'सन्धि' in A.s.taadhyaayi. He talks about rules governing combination of sounds in context of संहिता, which as defined in Sutra 109 of Chapter 4 of Book 1, means 'closest proximity of letters'.

संहिता will always be applicable within a word, between an upsarga and a verb root and between words in a compound formation. As regards sentences, it is optional. If words of a sentence are spoken together with an uninterrupted voice, संहिता applies and sound changes will take place at word boundaries. If there is a hiatus between two words, संहिता does not apply. When we write those words which were spoken in an unbroken flow of speech, we do sandhi while writing, because, unlike in English, we write what we speak, and sound changes take place in the condition of संहिता.

1.4 Types of Sandhi

Thus, sandhi can take place either within a word, or between two words. Thus, it is of two types:

1. Internal Sandhi

Sanskrit grammar has three kinds of minimal meaningful units (morphemes) – prefixes, roots and suffixes and every word in Sanskrit can be derived from them. When these units combine to form a word, संहिता applies and sound changes take place. This is known as internal sandhi.

For example, उप + ऋच्छति -> उपार्च्छति is a case of internal sandhi.

2. External Sandhi

When sandhi takes place between two words, it is known as external sandhi. For example,

तस्मै + एतत् -> तस्मायेतत् involves external sandhi.

Also, depending on whether the two letters that are combining are vowels, consonants or visarga, sandhis are classified as follows:

1. Vowel Sandhi: Both letters are vowels e.g. प्रति + एकः = प्रत्येकः.

2. Consonant Sandhis: At least one of the two letters is a consonant, e.g. वाक् + मयम् = वाङ्मयम्

3. Visarga Sandhis: A visarga combines with a vowel or a consonant, e.g. वृक्षः + शेते = वृक्षश्शेते

1.4 Organisation of Thesis

A survey of the work done in the field of designing of a sandhi splitter is presented next, and an evaluation of the exisiting sandhi splitters is then presented. The limitations of the existing splitters are discussed, and the need for designing an altogether new splitter is discussed. A new kind of representation of Paa.nini rules has been discussed, and then the algorithm for the design of the new sandhi splitter is brought out.

Chapter 2

R&D IN SANDHI SPLITTING AND LITERATURE REVIEW

2.1 Introduction

In the previous chapter, I motivated the need for the development of a sandhi splitter. In this chapter, I will first focus on the existing sandhi splitters and evaluate their performance. I will then present a review of the current literature in the field of sandhi splitting.

2.2 Existing Sandhi Splitters:

There has been considerable R & D in the field of sandhi splitting. As of now, three distinct sandhi splitters are available:

1. Sanskrit Sandhi Analyzer and Splitter

This is a vowel-sandhi splitter. This was developed at Jawaharlal Nehru University under the guidance of Professor Girish Nath Jha (Professor, Computational Linguistics, Special Centre for Sanskrit Studies). This is available at http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp .

2. Sandhi-Splitter (सन्धि-विच्छेदिका)

This was developed at University of Hyderabad under the guidance of Ms. Amba Kulkarni (Associate Professor, Department of Sanskrit Studies). This is available at http://sanskrit.uohyd.ac.in/scl/ .

Another Sandhi splitter is available at the TDIL website

http://tdil-dc.in/san/sandhi_splitter/index_dit.html but it is the same as the one mentioned above, the only difference being in version. The former is the latest version and so I will focus on it.

3. The Sanskrit Reader Companion

This is a Sasnkrit segmenter and parser, and therefore, also able to split sandhis. This was developed at INRIA, France under the guidance of Professor Gerard Huet, emeritus Professor. This is available at http://sanskrit.inria.fr/DICO/reader.fr.html .

2.3 Evaluating the Splitters

It is important to analyse how well each of these splitters perform. If they are able to detect the correct splits for all words, there may not be much need for a further research in this area, except for devising more efficient algorithms. If these splitters make mistakes, it is important to understand the nature of these mistakes, i.e. whether the splitters are not able to implement certain rules or whether it is a limitation of the lexicon/corpus being used to validate the splits. It is important to know this because only then these splitters can be improved.

This evaluation can be done in two ways:

1. Rule-based

2. Corpus-based

2.3.1 Rule-based Evaluation

As mentioned in the first chapter, Book 6 and Book 8 of Paa.nini's A.s.taadhyaayii contain rules which are governed by संहिता, and thus sandhi takes place as per these rules.

There are 271 such rules in total. Some of these rules deal with internal or external sandhis only while others deal with both.

I manually created a corpus having at least one example for each rule, and at least two examples where the same rule applies to both internal and external sandhis (one example each for the internal and the external sandhi cases). Each example is represented by a sandhied word, followed by the correct splits.

I divided the corpus into two parts: External Sandhi Corpus and Internal Sandhi Corpus. I evaluated all the three splitters for each sandhied word in each of the two corpora.

For most sandhied words, each of these splitters gives a very large number of possible splits. If any of the splits for a given word matches with the correct split, the splitter is considered to have correctly identified the splits, and a 1 is mentioned in the column corresponding to the particular splitter for that word. However, if none of the possible splits provided by the splitter turns out to be correct, a 0 is mentioned in the column.

While evaluating each of the three splitters for external sandhis, even if the splits are not fully correct and there is some error in the spellings of the words far away from the location where the sandhi takes place, I have considered the split as the correct split and marked 1 against that word for the particular splitter. Thus, I have given the benefit of doubt to these splitters.

However, in internal sandhi cases, if the splitters are not able to exactly detect the correct splits, I have marked a 0. This is because internal sandhis are mostly about detecting prefixes and suffixes and even a minute deviation from the correct word can potentially lead to a different prefix or suffix.

The two corpora and the result of evaluation for each word for both types of sandhi cases are available in Appendix A.

Here, I just summarise the results of rule-based evaluation.

External Sandhi Case:

Total Number of Words = 125

INRIA Sandhi Splitter Performance = 49/125 * 100 = 39.2 %

JNU Sandhi Splitter Performance = 21/125 * 100 = 16.8 %

UoH Sandhi Splitter Performance = 48/125 *100 = 38.4 %

Internal Sandhi Case:

Total Number of Words = 135

INRIA Sandhi Splitter Performance = 6/135 * 100 = 4.4 %

JNU Sandhi Splitter Performance = 14/135 * 100 = 10.4 %

UoH Sandhi Splitter Performance = 27/135 * 100 = 20 %

Overall Performance

Total Number of Words = 260

INRIA Sandhi Splitter Performance = 55/260 * 100 = 21.2 %

JNU Sandhi Splitter Performance = 35/260 * 100 = 13.5 %

UoH Sandhi Splitter Performance = 75/260 * 100 = 28.8 %

Inference:

1. The UoH splitter performs best overall. As far as external sandhis are concerned, its performance is more or less the same as that of the INRIA splitter. But it performs much better in case of internal sandhis. This is because it gives all possible outputs of sandhi splitting while the INRIA splitter rejects those splits which are not validated by the corpus used by it. So, by not rejecting the unlocated splits, the UoH splitter performs better!

2. The JNU splitter performs the worst. This was expected of it between it was designed for vowel sandhis only and the corpus it uses to validate its split is much more limited than the corpora used by the other two splitters.

3. All the three splitters perform better with external sandhis than with internal sandhis. This is most probably because the corpora they use may not have the set of prefixes, suffixes,etc.

4. All the three splitters are able to implement less than one-fourth of the total number of rules. And it is mostly the internal sandhi rules that have been neglected. But even with the external sandhi case, more than half of the rules have not been implemented.

5. Interestingly, each sandhi splitter is able to implement certain rules which the other two may not be able to. Howver, the number of cases not detected by any of the three splitters is also large.In fact, in the case of external sandhis, out of 125 cases, it is 55 cases that none of the three splitters is able to detect. However, in case of internal sandhis, none of the splitters is able to detect 99 out of 135 cases. Even if all the three splitters were combined somehow, the new splitter would be able to sort out only 40 % of the cases.

I now present the second kind of evaluation.

2.3.2 Literature-Corpus Based Evaluation

In this case, the corpus I used contains sandhied words available in some Sanskrit. The Sanskrit Computational Toolkit website provide links to such corpora. An example of this is http://sanskrit.uohyd.ac.in/Corpus/CIIL/Vinodini/vinodini-ext.txt . The corpora for the second kind of evaluation are based on these.

It is important to note that I do not use these corpora in their original forms. This is because in a text, there is a possibility of Sandhi taking place between every word and the next word, and one cannot rule out the possibility that many words of a given sentence will undergo sandhi related changes. The problem occurs because it is not necessary that when two words undergo sandhi, the new form that results is written as a single word. In cases of elision, for example, the two words are written as separate words.

Thus, it is not always logical to input word pairs having sandhi between them because the form of the second word may itself have changed on account of it having undergone Sandhi with the next

word. If such pairs are provided as inputs to the Sandhi splitter, it may not be able to process it well, nor is it expected to do so.

Therefore, I wrote a code that takes one of the available sandhi-extracted corpora files as input, and outputs a corpus having sandhied words, in which the last component is not modified (because it has not undergone sandhi with the subsequent word). The code is available in Appendix B.

Also, I automated the process of extracting words from such a corpus, and running each of the sandhi splitters on them, and getting the output. The extraction and evaluation, however, have been done manually. The code for such an automatic retrieval of output is available in Appendix C.

It must be noted that these second kind of corpora contain external sandhi cases only. Even the designers of the original corpora which from these are derived neglected internal sandhis.

I have shown the results for one such corpus, for the first 50 cases. The details of the evaluation can be obtained from Appendix D.

Here, I present the result of my evaluation.

Total Number of Cases: 50

JNU Sandhi Splitter Performance = 2/50 = 4 %

INRIA Sandhi Splitter Performance = 39/50 = 78 %

UoH Sandhi Splitter Performance = 31/50 = 62 %

Inference:

1. The JNU splitter again performs the worst. This is again mainly because it is a vowel-sandhi splitter only.

2. The INRIA and UoH splitter perform much better because they are not restricted to vowel sandhis.

However, it is also very important to note that when it is a literature-based corpus, there are lots of cases of compound formation, which may or may not involve sandhi. The INRIA system is actually a segmenter, so it not only detect sandhis but also compounding. The UoH splitter gives us an option to detect both sandhis and compounding, and the above results were obtained through an exercise of this option.

2.3.3 Understanding the Issues Involved

1. We saw that all the three splitters combined are able to detect only 40 % of the cases.When we observe the rules which have not been implemented by any of the three splitters, we recognise that they are special kinds of rules where the sandhi process does not depend only on two letters, but may also depend on other letters of the words, the words themselves, the words in their vicinity, meaning, etc. Well, all this creates a problem mostly for doing sandhi, and the sandhi splitting process need not take all these special features into account. However, sometimes it is essential to consider one or more of these special features to be able to split sandhied words successfully. As an example, while splitting the word raamaaya.na, one also needs to take into account the presence of ‘r’ at the beginning of the word, because it is this which is causing n of ayana to change to .n while the word raamaaya.na is being formed.

2. The corpora used to validate the splits also put a great limitation on detection of sandhis, mostly internal sandhi cases. They also need to be updated. All prefixes and suffixes and as many more verb roots as possible should be added to the existing corpora.

3. Compounding also needs to be taken care of, only then the splits in many cases will be validated.

Since there are a large number of rules to be incorporated, and many new kinds of features need to be taken into account, it makes sense to work towards designing an alternate sandhi splitter which implements all the rules in the best possible way, rather than forcing new kinds of rules into the existing splitters to update them. This will not only be difficult, but also inefficient.

I now present a literature review as to what kind of research exists in this direction, and overall.

2.4 Literature Review

1. Sandhi Splitter and Analyzer for Sanskrit (With Special Reference to aC Sandhi)

By: Mr. Sachin Kumar, M.Phil. research scholar at JNU (2005-07)

This is the M.Phil. thesis of Mr. Sachin Kumar who contributed to the development of the Vowel-sandhi splitter developed at JNU. He has discussed several limitations of the splitter designed by him. In fact, as mentioned in the thesis, only ten rules have been implemented. This is against a total of about 271 rules.

This document is very useful in understanding sandhi but does not provide any clue to solve the present problem.

2. A Binary Schema and Computational Algorithms to Process Vowel-based Euphonic Conjunctions for Word Searches

By: Kashmir Raja S.V. (Dean, Research, SRM University, Chennai)

Rajitha V. (Meenakshi College for Women, Chennai)

Meenakshi Lakshmanan (Mother Teresa's Women University, Kodaikanal)

This work presents a representational schema that represents letters in a binary format and reduces Paa.nini rules of euphonic conjunctions (sandhis) to simple bit set-unset operations. An efficient algorithm to process vowel-based sandhis using this schema is also presented.

Though the effort is laudable, the whole initiative suffers from a major limitation. This focusses on rules where knowledge of last two letters of the first word and first two letters of the second word is sufficient to decide whether sandhi should take place or not. But as mentioned above, much more information than this (like meaning, adjacent words, other letters of the same word, compound-formation) may be required in the case of certain rules and it may not always be possible or efficient to represent all those conditions in this format.

3. Analysis of Sanskrit Text: Parsing and Semantic Relations

By : Pawan Goyal, Vipul Arora and Laxmidhar Behera (Electrical Engineering, IIT Kanpur)

In this paper, they have presented their work towards building a dependency parser for Sanskrit language that uses deterministic finite automata (DFA) for morphological analysis and 'utsarga apvaada' approach for relation analysis. They talk about their sandhi analyser which uses DFA. But

again, this is limited to about 10 rules, and seems to be in the direction of something that has already been limited. The algorithm they present does not take care of the various conditions that enable the application of sandhi rules, and thus does not prove relevant.

4. From Paa.nini Sandhi to Finite State Calculus

By : Malcom D. Hyman (Max Planck Institute for the History of Science, Berlin)

This paper summarizes the handling of sandhi by Paa.nini and the notational conventions used by him. It also introduces an XML vocabulary for expressing Paa.nini rules. It refers to 40 core rules of external sandhi. It suffers from the same limitation as mentioned above- it is limited to implementation of some rules and does not does not take care of the various conditions that enable the application of sandhi rules.

5. Phonological Overgeneration in Paaninian System

By: Malhar Kulkarni (IIT Powai, Mumbai)

M.M. Vasudevashastri Abhyankarshastri Pathasala, Pune

This paper discusses the problem of overgeneration that is caused by the application of the system of Paanini (rules stated by Paanini and his commentators namely Katyayana and Patanjali). Two cases related to phonological overgeneration are studied and possible solutions are proposed to avoid the problem.

Though this seems to be the only paper which discusses this kind of rules, it deals with Sandhi and not Sandhi splitting. Also, it is not about implementation in the form of a program. Thus, the paper does not serve my purpose.

6. Automatic Sanskrit Segmentizer Using Finite State Transducers

By: Vipul Mittal (Language Technologies Research Center, IIT- Hyderabad)

This paper discusses two approaches to sandhi splitting using FST. However, it also assumes that the rules are in the form of two letters combining to give a letter and does not consider the various other conditions involved.

2.5 Conclusion

The performance of the existing sandhi splitters is poor. The available literature on sandhi splitting is more or less about how to implement some sandhi rules, but it does not serve much purpose because these rules have already been implemented by one or more of the splitters already available. It may turn out that the new ways of implementation of these rules turn out to be more efficient than the existing ones, but this does not solve the problem at hand. Thus, I had to start thinking of the design of a new sandhi splitter all together.

Chapter 3

CODIFICATION OF PAA.NINI RULES

3.1 Introduction

As discussed in the previous chapter, there is a need of designing of an all-together new sandhi splitter which implements all the rules concerning sandhi. A brute force method for splitting sandhi involves scanning all the letters of a given word and trying to split the word at every possible location, and validating the splits obtained by referring to a corpus maintained for the same purpose. But this is a highly inefficient way of sandhi splitting. We need to look for a more efficient algorithm. One way is to look for patterns in the sandhi rules of Paa.nini. If some sort of pattern exists therein, maybe we can exploit it to formulate a better algorithm.

To be able to do this, it is necessary to lay out all the rules. This is essentially what this chapter is all about.

3.2 Source of Rules

Paa.nini's A.s.taadhyaayi is in the form of very brief sutras, which have been explained by his commentators. The source of the rules I have presented is *The A.s.taadhyaayi of Paa.nini Translated into English by Srisa Chandra Basu* in which he has translated the Paanini's aphorisms, as explained by the Commentators Jayadita and Vamana in their well-known book called Kasika vritti. He also draws upon the well-known translation of Laghu Kaumudi by Dr. Ballayante, Mr. Iyenagar's Guide to Paa.nini , Professor Apte's Sanskrit Composition as well as from Dr. Kielhorn's Paribhashendusekhara.

3.3 Sandhi Rules

As discussed in the first chapter, sandhi rules are those rules that are governed by the condition of संहिता. Rules 73 to 157 in Chapter 1 of Book 6, and all rules in Chapter 3 and Chapter 4 of Book 8 fall under this category. They constitute a total of 271 rules.

3.4 Sanskrit Alphabet

Sanskrit alphabet consists of: [2]

1. Vowels –

(a) Short - अ इ उ ऋ ऌ

(b) Long - आ ई ऊ ॠ ए ऐ ओ औ

(c) Pluta – अ3 इ3 उ3 ऋ3 ऌ3 ए3 ऐ3 ओ3 औ3

Each of these vowels can be pronounced in three different ways.

1. Udaatta (Acute accent, high pitch)

2. Anudaatta (Grave accent, low pitch)

3. Svarita (Circumflex, high falling pitch)

Vowels in udaatta mode are written as before, in anudaatta mode, a horizontal line is drawn under them and svarita vowels are written with a vertical line drawn above them.

Consonants:

There are of two kinds:

1. Grouped Consonants

क् ख् ग् घ् ङ् (कवर्ग)

च् छ् ज् झ् ञ् (चवर्ग)

ट् ठ् ड् ढ् ण् (टवर्ग)

त् थ् द् ध् न् (तवर्ग)

प् फ् ब् भ् म् (पवर्ग)

2. Ungrouped Consonants: They are again of two kinds:

Semi-Vowels: य् व् र् ल्

Uu.smaa.na: श् ष् स् ह्

Ayogvaahaa

1. Anusvara ं

2. Visarga (:)

3. Jhivaamuliya

4. Upadhmaaniya

All letters can also be annunaasika or unannunaasika (not annunaasika). The former are denoted by a chandra bindu (ँ).

All the above letters (which include two Vedic Sanskrit categories – Jhivaamuliya and Upadhmaaniya) have been referred to by Paa.nini while listing संहिता rules.

The following two Vedic Sanskrit letters - ळ (retroflex lateral approximant) and its aspirated counterpart ळ्ह् have not been referred to by Paa.inini while listing संहिता rules, and so they are not used in the Sandhis.

3.5 Paa.nini's System of Representation

Shiva Sutras:

Paanini's A.s.taadyaayii makes use of the Shiva Sutras. Shiv Sutras are fourteen verses that organise the phonemes of Sanskrit. Each verse consists of a group of basic Sanskrit phonemes (i.e. open syllables consisting either of initial vowels or of consonants having the basic vowel अ) followed by a single 'dummy letter'.

They are as follows:

1. अइउण् 2. ऋऌक् 3. एओङ् 4. ऐऔच् 5. हयवरट्

6. लण् 7. ञमङणनम् 8. झभञ् 9. घढधष् 10. जबगडदश्

11. खफछठथचटतव् 12. कपय् 13. शषसर् 14. हल्

Pratyaahaara:

A pratyaahaara is a set. Its name has two parts- the first is a phoneme-letter and the second is a dummy letter. A pratyaahara signifies all letters between these two, including the first letter. The consonants are taken in without the अ.

For example,

अच् contains all vowels, हल् contains all consonants and अल् contains all letters.

It must be noted that Paa.nini never uses Pratyaahaaras to denote a single phoneme. So, हल् means all consonants and not just ह्.

Pratyaahaaras are also declined in A.s.taadhyaayi. For example, the letters of a pratyahara shown in genitive case are replaced by replaced by the letters of a pratyahara shown in nominative before the letters of a pratyahara shown in locative case.

For example,

A.s.taadhyaayii, 6.1.77 इको यणचि (इकः यण् अचि)

- इकः is in the genitive case, it means 'in place of इ उ ऋ ऌ'
- यण् is in the nominative case, it means 'are substituted य् व् र् ल्'
- अचि is in the locative case, it means 'before अ इ उ ऋ ऌ ए ओ ऐ औ'

Thus, इको यणचि means य् व् र् ल् are substituted in place of इ उ ऋ ऌ before अ इ उ ऋ ऌ ए ओ ऐ औ.

Also, Paa.nini uses the following five sets: 1. कु - क् ख् ग् घ् ङ्, 2. चु - च् छ् ज् झ् ञ्,

3. टु - ट् ठ् ड् ढ् ण्, 4. तु - त् थ् द् ध् न् and 5. पु - प् फ् ब् भ् म्

3.6 Transliteration Scheme Used

The scheme used in the project for the transliteration of Devanagri to Roman script is Velthius. Though it does not include all the letters which two other schemes (SLP1 and Unicode) include, it has been used because compared to both of them, it is easier to relate to devanagari and also fits well within the representation document.

The scheme is shown below:

Devanagri	**Velthius**
अ	a
आ	aa
इ	i
ई	ii
उ	u
ऊ	uu
ऋ	.r
ॠ	.rr
ऌ	.l
ए	e
ऐ	o
ओ	ai
औ	au
क्	k
ख्	kh
ग्	g
घ्	gh
ङ्	“n
च्	c
छ्	ch

ज्	j
झ्	jh
ञ्	~n
ट्	.t
ठ्	.th
ड्	.d
ढ्	.dh
ण्	.n
त्	t
थ्	th
द्	d
ध्	dh
न्	n
प्	p
फ्	ph
ब्	b
भ्	bh
म्	m
य्	y
व्	v
र्	r
ल्	l
श्	“s
ष्	.s
स्	s

ह् h

Anusvara ं .m

Visarga (:) .h

Velthius has no equivalent representation for the following letters/symbols of Sanskrit.

1. Annunasika (Chandra Bindu)

2. Upadhmaaniya

3. Jhivaamuliya

4. Udatta vowels

5. Anudatta vowels

6. Avagraha (which denotes an elision sometimes)

So, these have been referred to by these very names in the representation scheme to follow.

3.7 NEW REPRESENTATION OF PAA.NINIAN SYSTEM OF RULES CONCERNING SANDHI

I describe here the scheme using which I codified all the 271 sandhi rules. Rules in codified form are available at

https://docs.google.com/spreadsheets/d/1RIFXSiIH9tN5Mvqv4PUII-cBEJ4JJ--Pka_UpornFLA/edit .

1. Rule No – The first digit refers to the book, the second to the chapter in that book, the next one/two/three digits (before the period, if these is one) refer to the rule number in that chapter, and the last digit includes the sub-rules.

2. Last (w1) – This represents the last letter of the first word.

3. First (w2) - This represents the first letter of the second word.

4. Left-Context – This is an umbrella category which covers various features of the first word.

   Some of them are as follows:

   Second_Last(w1) - This represents the second last letter of the first word.

   Pada () denotes the set of all words which are declined forms of a verb root or noun root.

   Upsarga () denotes the set of all prefixes.

   Other kinds of features are specific to the Sandhi rules.

5. Right- Context - This is an umbrella category which covers various features of the second word.

   Some of them are as follows:

   Second (w2) - This represents the second letter of the second word.

   Pada () as described above.

Pratyaya () denotes the set of all suffixes.

Other kinds of features are specific to the Sandhi rules.

6. Overall-Context – This is an umbrella category for the conditions which go beyond the structure of the two words involved.

For example, the sense in which a word is used, the places in which it has to appear for the rule to apply, etc.

7. Action -

This is about the various ways sandhi takes place. They are described below.

Let

a represent the last letter of the first word.

b represent the first letter of the second word.

c represent the new letter that results from the sandhi process.

It must be noted that wherever a and b denote vowels, c is the final change. In cases of consonants and visarga, the resultant c arising out of that sandhi rule, can reundergo sandhi with next b , or previous a, if (c,b) or (a,c) satisfy (Last(w1), First(w2)) for any rule as shown in the above link.

7.1 Aagam Rules :

Aagam

a + b -> acb (It is final form, so not written as a + c + b)

Pre-Aagam

a + b -> ac + b

Post-Aagam

a + b-> a + cb

7.2 Aadesh Rules:

Pre-Aadesh

a + b -> c + b

Post-Aadesh

a + b -> a + c

7.3 Ekadesh

a + b -> c

7.4 Elision Rules:

Pre-Elision

a + b -> b

Post-Elision

a + b -> a

7.5 Prakritibhaav

a + b -> a + b

7.6 Reduplication Rules:

Pre-Reduplication

a + b -> aa + b

Post-Reduplication

a + b -> a + bb

7.7 Comment

This indicates whether a rule is an optional rule or whether it is a rule in the opinion of some particular Acharyas, in the case of which other Acharyas may not it accept as a rule.

Set Notation Used

As described earlier, Paa.nini also uses sets (pratyaharas) in his sutras. But there are many places where he could have used the pratyaharas but he does not do so because the Shiv sutras (which are the basis of all sets) do not explicitly represent all those Sanskrit sounds. I have explained in detail all cases below:

1. There is no pratyahaara which explicitly refers to the long vowels aa, ii, uu, .rr, but there are a large number of sandhi rules where it matters whether the vowel is a short vowel, a long vowel or a pluta vowel. Paa.nini uses the word Dirgha and Pluta to denote the last two kinds of rules. Though these two categories may be considered to represent sets in themselves, I chose to employ an explicit set notation for the representation of Dirgha vowels aa, ii, uu, .rr . Paan.ini uses the ak pratyahar when he wants to refer to all vowels except e, ai, o, au and the Pluta vowels, because it seems that ak is considered to include the above four long vowels as well. But when he wants to refer to long vowels only, he uses the word Dirgha. In the case of Pluta vowels, the Pluta() set was sufficient.

Thus, I add one more verse to the Shiv Sutras. The updated Shiv Sutras in Velthius notation, are listed below. The third sutra is the new addition.

1. a I u .n 2. .r .l k 3. aa ii uu .rr g

4. e o “n 5. ai au c 6. h y v r .t

7. l .n 8. ~n m “n .n n m 9. jh bh ~n

10. gh .dh dh .s 11. j b g .d d “s 12. kh ph ch .th c .t t v

13. k p y 14. “s .s s 15. h .l

Another difference between these and the original Shiv Shutras is that ‘a’ is not attached to any consonant, whereas in the original ones, it was attached with all except the last consonant of each verse.

To refer to sets of letters derived from these, I use the Maheswar Function, written as M (a,b) where b must be a dummy letter from one of the above sutras. As was the case with pratyaaharas, M (a,b) returns a set containing all the letters starting from a and before b.

Thus, M (aa,g) = {aa , ii , uu , .rr)

2. The sets Anudaata() and Svarita() are used to represent Anudaata and Svarita vowels respectively.

3. The notation *k,*kh and *p,*ph represent Jhivaamuliya and Upadhmaaniya respectively.

3.8 Conclusion

The rules have been coded in a format so that one can identify patterns in them. If a rule is seen missing, it is because it is a governing rule (one which has effect on other rules) and these effects have been incorporated in the relevant rules.

It must be also be noted that the representation is not complete in itself like A.s.taadhyaayi is. It is because there are still certain terms that have been deliberately left unexplained. They occur in the Left-Context, Right-Context and the Overall-Context sections. They mostly have to do parts of speech a word belongs to or other such categories. It is not really required to precisely describe them. This will become evident in the next chapter.

Chapter 4

Developing Algorithm for the Proposed Sandhi Splitter

4.1 Introduction

The sandhi rules have been codified with the notation explained in the last chapter. The codified rules are available at https://docs.google.com/spreadsheets/d/1RIFXSiIH9tN5Mvqv4PUII-cBEJ4JJ--Pka_UpornFLA/edit . A detailed analysis of them makes one realise that there are certain features which can be exploited to design a better algorithm than the brute force one referred to in the last chapter.

The three sections on Action, Left-Context and Right-Context seem particularly helpful in this direction. In fact, there are many cases in which what appears to be problem in the implementation of sandhi becomes an advantage while sandhi splitting.

4.2 Clues to Development of a Better Algorithm

1. There are a large number of rules which result in the production of certain kinds of strings, or special kinds of symbols. The presence of such special symbols or such strings leads to very high probability of sandhi splitting at those places.

Some such examples are:

Special letters/symbols: $ (Avagraha), ~n, “n Annunasika, Nasal, Anusvaara, Udatta and Svarita vowels, Space

Special Strings: aay, aav, aar, ay , av, ar ,ai, ou, cch, sk, two vowels coming together, etc

It will be very useful if the splitter checks for the presence of these letters or strings in the input in the beginning itself.

2. None of the sets under the action column contains a reference to the following letters:

kh, ph, tha (these are the second letters of kavaraga, pavarga and tavarga respectively.

It may not be useful to split a word at the place where these letters occur.

3. There are a large number of Aagam rules but most of them refer to the addition of s before k, leading to the formation of the string sk .Similarly, a large number of Post-Aadesh rules exist, but most of them refer to substitution of .s for s. Thus, Aagam and Post-Aadesh rules, though large in number, are not diverse in terms of the effect they produce. On the other hand, there are a large number of Pre-Aadesh and Ekadesh rules with diverse effects.

In case of Prakritibhaav rules, two vowels come together or there is a hiatus. Elision necessarily involves hiatus. They have already been taken care of earlier.

On the whole, it will be useful if the splitter splits decides the priority of splitting a letter by referring to kind of rule (Pre-Aadesh, Ekadesh, etc) it results from.Once the special string sk and the letter .s have been taken care of, in the initial steps, the splitter should split letters belonging to different kinds of rules in the following order.

Ekadesh > Pre-Aadesh >Aagam

Also, some rules are used more often in the literature than other rules. The letters which result from the application of these rules should be given higher priority while splitting the word which contains them. For examples, some such rules are:

1. Lengthening of vowels leading to aa, ii , uu , .rr (Ekadesh rule)

2. Conversion of vowels to semi-vowels – y, v, r, l (Pre-Aadesh Rule)

3. Ekadesh rules leading to e,ai, o, au

4. Consonant sandhi rules leading to substitution of first and third letters of different groups

- (k , c , .t, t ,p , g, j ,.d, d ,b)

5. Visarga sandhi rules leading to r and o

4. The less used rules have more context conditions, and this becomes useful while doing sandhi splitting. While splitting a word at a letter, use can be made of the extra conditions that are associated with the rules which result in those letters.

For example, there is no need to split a word at a if this letter is not preceeded by iim, aar, etc.

Let me know present the algorithm based on these and other considerations (the latter will become clearer with the algorithm itself).

4.3 Proposed Algorithm for Sandhi Splitting

1. Get the word or the pair of words to be sandhi split. In case a continuous text is given as input, analyse the last word of each sentence for sandhi splitting. Then, take the last two words. Proceed in this fashion till all the words of the sentence have been processed.

2. Check for the presence of r and .n in the same word. If the word contains .n followed by r (even if some letters intervene in between), generate a new word replacing .n by n. Evaluate the new word first as per the rest of the algorithm. If results are not obtained, using the original word.

3. Check for the presence of .s in a word. If it is preceded by any vowel other than a (occurring anywhere before, not necessarily, just before it), generate a new word by replacing .s by s. Evaluate the new word first as per the rest of the algorithm. If results are not obtained, using the original word.

4. Check for the presence of the following special letters/ strings in the input:

Special letters/symbols: $ (Avagraha), ~n, “n Annunasika, Nasal, Udatta and Svarita vowels, Space

Special Strings: aay, aav, aar, ay, av, ar, ai, sk, .st.ou, cch, etc

If detected, reverse apply the concerned rules to get the splits. Try to locate the splits in the corpus available. If none of the splits is located, treat both of them as new words and Go to step 1. If one of the splits is validated, test on the remaining ones by sending them as new words.

5. Try splitting the word at all letters except kh, ph and tha. Priority for splitting should be given to those letters which are the result of the sandhis which are more frequently used (as discussed earlier).

Try validating the splits through the corpus. If the splits are themselves long, treat them as new words and Go to step 1. If one of the splits is validated, test on the remaining ones by sending them as new words.

6. If results are not found, split words at letters resulting from rules in the following order:

Pre-Aadesh >> Ekadesh >>Post- Aadesh

Make use of the context conditons simultaneously. If the context conditions are not satisfied, try splitting at other places.

Only when context conditions have been satisfied, send the splits for validation by the corpus available. Again, if the splits are themselves long, treat them as new words and Go to step 1. If one of the splits is validated, test on the remaining ones by sending them as new words.

7. Output the splits that are validated by the corpus.

4.4 Conclusion

The above algorithm will detect splits that are not detected by the other splitters primarily because the latter do not implement most of the rules. Also, because it prioritises as to which letters to split a word at and also makes use of the context conditions, it will be more efficient than than the brute-force algorithm in which the word is split at all locations and then the splits are sent for validation.

Chapter 5

CONCLUSION

5.1 Summary

Sandhi splitting is an important area of concern in Sanskrit Computational Linguistics because sandhi is a very common phenomenon in Sanskrit and neither parsing nor morphological analysis is possible without splitting the sandhied words. The existing sandhi splitters are not able to determine the correct splits for a large percentage of cases either because they are not able to implement the required sandhi rules, or the corpora they use to validate the splits are inadequate. Updating the corpora is one way of reducing the problem. To solve the other problem, a new sandhi splitter was required, which could implement all the sandhi rules of A.s.taadhyaayii. The first step in the design of the new splitter was to list down all the relevant rules in a format that could be directly utilised for coding. Thus, I made an attempt to represent those rules in a new format so that they can be used easily. The new representation of Paa.nini rules concerning Sandhi has been made available at https://docs.google.com/spreadsheets/d/1RIFXSiIH9tN5Mvqv4PUII-cBEJ4JJ--Pka_UpornFLA/edit .

5.2 Future Work

Now that the rules have been documented well, and the algorithm broadly described, the next step would be to precisely state all the steps involved in the implementation of this algorithm. This will involve specifying all the conditions in which a word can be split at a given letter, and these will have to be specified for all the letters. A code will then have to be written for the new sandhi splitter which will involve reverse implementation of all the sandhi rules. The new sandhi splitter will then be evaluated in the same way as the other splitters have been evaluated, and any limitations of it would then be addressed.

REFERENCES

1. R. Briggs, *Knowldege Representation in Sanskrit and Artificial Intelligence*, 1985, The AI Magazine, pp 33-39

2. Srisa Chandra Basu, *The A.s.taadhyaayi of Paa.nini Translated into English*, 1897

3. G. M. Bhatt , *Sandhi.h* , 1998 ,Samskrita Bharati Prakaashan, Benguluru

4. K. D. Dviwedi , *Prau.dh Rachnaanuvaad Kaumudi* , 1960,Vidyaalaya Prakaashan, Varanasi

5. S. Kumar, *Sandhi Splitter and Analyzer for Sanskrit (With Special Reference to aC Sandhi),* 2007

6. M. D. Hyman, *From Paa.nini Sandhi to Finite State Calculus,* Max Planck Institute for the History of Science, Berlin

7. Kashmir Raja S.V., Rajitha V. et al. *A Binary Schema and Computational Algorithms to Process Vowel-based Euphonic Conjunctions for Word Searches*

8. P. Goyal, V. Arora, et al. *Analysis of Sanskrit Text: Parsing and Semantic Relations,* IIT Kanpur

9. M. Kulkarni, M.M. Vasudevashastri, *Phonological Overgeneration in Paaninian System*, IIT Powai, Mumbai

10. V. Mittal , *Automatic Sanskrit Segmentizer Using Finite State Transducers* , IIT- Hyderabad

APPENDIX A

EXTERNAL SANDHI EVALUATION

RULE NO.	EXAMPLE	INRIA JNU UoH
6.1.73.1	स्वच्छन्दः , स्व , छन्दः	1 0 1
6.1.74.1	माच्छिदत् , मा , छिदत्	0 0 1
6.1.76.1	लक्ष्मीच्छाया , लक्ष्मी , छाया	1 0 0
6.1.77.1	प्रत्येकः , प्रति , एकः	1 1 1
6.1.77.2	नद्यत्र , नदी , अत्र	1 1 1
6.1.77.3	मध्वत्र , मधु , अत्र	1 1 1
6.1.77.4	वध्वाज्ञा , वधू , आज्ञा	1 0 1
6.1.77.5	मात्राज्ञा , मातृ , आज्ञा	1 0 1
6.1.77.6	लाकृतिः , ऌ , आकृतिः	0 0 0
6.1.78.1	फलयिच्छा , फले , इच्छा	1 0 1
6.1.78.2	तस्मायेतत् , तस्मै , एतत्	1 0 1
6.1.78.4	तावेकदा , तौ , एकदा	1 0 1
6.1.79.1	गव्यूतिः , गो , यूतिः	0 1 0
6.1.86.1	कोऽसिचत् , कस् , असिचत्	1 0 0
6.1.87.1	देवेन्द्रः , देव , इन्द्रः	1 1 1
6.1.87.2	परोपकारः , पर , उपकारः	1 1 1
6.1.87.3	देवर्षिः , देव , ऋषिः	1 1 1
6.1.87.4	तवल्कारः , तव , ऌकारः	1 0 0
6.1.88.1	अत्रैकः , अत्र , एकः	1 1 1
6.1.88.2	देवैश्वर्यम् , देव , ऐश्वर्यम्	1 0 1
6.1.88.3	तण्डुलौदनम् , तण्डुल , ओदनम्	1 0 1
6.1.88.4	देवौदार्यम् , देव , औदार्यम्	1 0 1

RULE NO.	EXAMPLE	INRIA JNU UoH
6.1.89.1	प्रष्ठौहः , प्रष्ठ , ऊहः	0 0 0
6.1.89.2	विश्वौहः , विश्व , ऊहः	1 0 0
6.1.89.3	अक्षौहिणी ,अक्ष, ऊहिनी	0 1 0
6.1.89.4	स्वैरः ,स्व , ईरः	0 0 0
6.1.94.1	शकन्धुः , शक , अन्धुः	0 1 1
6.1.94.2	मनीषा , मनस् , ईषा	0 1 0
6.1.94.3	कुलटा , कुल , अटा	0 1 1
6.1.94.4	सीमन्तः , सीम , अन्तः	0 0 1
6.1.94.5	सारङ्गः , सार , अङ्गः	0 1 1
6.1.94.6	स्थूलोतुः , स्थूल , ओतुः	0 1 1
6.1.94.7	बिम्बोष्ठः , बिम्ब , ओष्ठः	0 1 1
6.1.101.1	हिमालयः , हिम , आलयः	1 1 1
6.1.101.2	गिरीशः , गिरि , ईशः	1 1 1
6.1.101.3	विष्णूदयः , विष्णु , उदयः	1 1 1
6.1.101.4	होतृकारः , होतृ , ऋकारः	1 1 0
6.1.109.1	सेवतेऽहम् , सेवते , अहम्	1 0 0
6.1.109.2	लोकोऽयम् , लोको , अयम्	1 0 0
6.1.114.1	पुरुषो हसति , पुरुषर् . हसति	1 0 0
6.1.123.1	गवाग्रम् , गो , अग्रम्	0 1 0
6.1.124.1	गवेन्द्रः , गो , इन्द्रः	0 1 0
6.1.127.1	कुमारि अत्र , कुमारी , अत्र	0 0 0
6.1.128.1	ब्रह्मऋषिः ,ब्रह्मा , ऋषिः	0 0 0
6.1.132.1	एष ददाति , एषः , ददाति	0 0 0
6.1.134.1	सैष दाशरथी रामः , सः , एष दाशरथी रामः	0 0 0

RULE NO.	EXAMPLE	INRIA	JNU	UoH
8.3.3.1	महाँ असि , महान् , असि	0	0	0
8.3.7.1	तस्मिंस्तरौ , तस्मिन् , तरौ	1	0	1
8.3.8.1	तस्मिँस् त्वा , तस्मिन् , त्वा	0	0	0
8.3.9.1	महाँ इन्द्र , महान् , इन्द्र	0	0	0
8.3.10.1	नॄँः पाहि , नॄन् , पाहि	0	0	0
8.3.10.2	नॄंः पाहि , नॄन् , पाहि	0	0	0
8.3.12.1	काँस्कान् , कान् , कान्	0	0	0
8.3.14.1	पुना रमते , पुनर् , रमते	0	0	0
8.3.15.1	वृक्षस्तरति , वृक्षर् , तरति	1	0	1
8.3.17.1	भो अत्र , भोर् , अत्र	1	0	0
8.3.19.1	क आस्ते , कय् , आस्ते	1	0	0
8.3.20.1	भो इदम् , भोय् , इदम्	1	0	0
8.3.22.1	भगो हसति , भगोय् , हसति	0	0	0
8.3.23.1	वनं गच्छति , वनम् , गच्छति	1	0	0
8.3.27.1	कथन्ह्नुते , कथम् , ह्नुते	0	0	0
8.3.31.1	सञ्च्छम्भुः , सन् , शम्भुः	1	0	0
8.3.32.1	प्रत्यङ्ङात्मा , प्रत्यङ् , आत्मा	1	0	0
8.3.33.1	शम्वस्तु , शमु , अस्तु	0	0	1
8.3.34.1	वृक्षस्तरति , वृक्षः , तरति	1	0	1
8.3.36.1	वृक्षश्शेते , वृक्षः , शेते	1	0	1
8.3.38.1	पयस्पाशम् , पयः , पाशम्	0	0	0
8.3.39.1	सर्पिष्पाशम् , सर्पिः , पाशम्	0	0	0
8.3.40.1	नमस्कर्त्ता , नमः , कर्त्ता	0	0	0
8.3.41.1	आविष्कृतम् , आविः , कृतम्	0	0	1

RULE NO.	EXAMPLE	INRIA JNU UoH
8.3.42.1	तिरस्करोति , तिरः , करोति	0 0 1
8.3.43.1	द्विष्करोति , द्विः , करोति	0 0 1
8.3.44.1	सर्पिष्करोति , सर्पिः , करोति	0 0 1
8.3.45.1	धनुष्फलम् , धनुः , फलम्	0 0 0
8.3.46.1	अयस्कारः , अयः , कारः	0 0 0
8.3.47.1	अधस्पदम् , अधस् , पदम्	0 0 0
8.3.49.1	अयस्पात्रम् , अयः , पात्रम्	0 0 0
8.3.50.1	पयस्करति , पयः , करति	0 0 0
8.3.51.1	महस्परि , महः , परि	0 0 0
8.3.52.1	दिवस्पातु ,दिवः , पातु	0 0 0
8.3.53.1	तमसस्पारम् , तमसः , पारम्	0 0 0
8.3.54.1	इडायास्पतिः , इडायाः , पतिः	0 0 0
8.3.80.1	अङ्गुलिषङ्ग , अङ्गुलि , सङ्ग	0 0 0
8.3.81.1	भीरुष्ठानम् , भीरु , स्थानम्	0 0 0
8.3.82.1	अग्निष्टुत् , अग्नि , स्तुत्	0 0 0
8.3.83.1	आयुष्टोमः , आयु , स्तोमः	0 0 0
8.3.84.1	मातृष्वसा , मातृ , स्वसा	0 0 0
8.3.85.1	मातुःष्वसा , मातुर् , स्वसा	0 0 0
8.3.95.1	युधिष्ठिरः , युधि , स्थिरः	0 0 0
8.3.97.1	अग्निष्ठः , अग्नि , स्थः	0 0 0
8.3.98.1	सुषाम , सु , साम	0 0 0
8.3.101.1	सर्पिष्टरम् , सर्पिस् , तरम्	0 0 0
8.3.103.1	अग्निष्टे , अग्निस् , ते	0 0 0
8.3.105.1	गोष्टोमम् , गो , स्तोमम्	0 0 0

RULE NO.	EXAMPLE	INRIA JNU UoH
8.3.106.1	मधुष्ठानम् , मधु , स्थानम्	0 0 0
8.4.3.1	शूर्पणखा , शूर्प , नखा	0 0 0
8.4.4.1	शारिकावणम् , शारिका , वनम्	0 0 0
8.4.5.1	शरवणम् , शर , वनम्	0 0 0
8.4.6.1	बदरीवणम् , बदरी , वनम्	0 0 0
8.4.7.1	पूर्वाहणः , पूर्व , अहनः	0 0 0
8.4.8.1	इक्षुवाहणम् , इक्षु , वाहनम्	0 0 0
8.4.9.1	कषायपाणः , कषाय , पानः	0 0 0
8.4.10.1	सुरापाणम् , सुरा , पानम्	0 0 0
8.4.13.1	स्वर्गकामिणौ , स्वर्ग , कामिनौ	0 0 0
8.4.24.1	अन्तर्हण्यते , अन्तर् , हन्यते	0 0 0
8.4.25.1	अन्तरयणम् , अन्तर् , अयनम्	0 0 0
8.4.40.1	वृक्षश्शेते , वृक्षस् , शेते	1 0 0
8.4.40.2	अग्निचिच्छेते , अग्निचित् , शेते	1 0 1
8.4.40.3	सच्चित् , सत् , चित्	1 0 1
8.4.40.4	कश्चित् , कस् , चित्	1 0 1
8.4.41.1	वृक्षष्टीकते , वृक्षस् , टीकते	0 0 0
8.4.41.2	वृक्षष्षण्डे , वृक्षस् , षण्डे	0 0 1
8.4.41.3	तट्टीका , तत् , टीका	1 0 1
8.4.42.1	षण्णाम् , षट् , नाम्	0 0 0
8.4.45.1	वाग्नयति , वाक् , नयति	1 0 1
8.4.45.2	वाङ्नयति , वाक् , नयति	1 0 1
8.4.45.3	वाङ्मयम् , वाक् , मयम्	1 0 1
8.4.47.1	दद्ध्यत्र , दध्य् , अत्र	0 0 0

RULE NO.	EXAMPLE	INRIA JNU UoH
8.4.55.1	युयुत्सते , युयुध् , सते	0 0 0
8.4.59.1	त्वङ्करोषि , त्वं , करोषि	1 0 1
8.4.60.1	तल्लीनः , तत् , लीनः	1 0 1
8.4.60.2	विद्वाँल्लिखति , विद्वान् , लिखति	0 0 0
8.4.62.1	वाग्घरिः , वाग् + हरिः	1 0 1
8.4.63.1	तच्छिवः , तत् , शिवः	1 0 1
8.4.63.2	तच्शिवः , तत् , शिवः	1 0 1

Total Number of Words = 125

INRIA Sandhi Splitter Performance = 49/125 * 100 = 39.2 %

JNU Sandhi Splitter Performance = 21/125 * 100 = 16.8 %

UoH Sandhi Splitter Performance = 48/125 *100 = 38.4 %

INTERNAL SANDHI EVALUATION

RULE NO.	EXAMPLE	INRIA	JNU	UoH
6.1.73.1	विच्छेद , वि , छेद	0	0	1
6.1.74.1	आच्छादयति , आ , छादयति	0	0	1
6.1.75.1	चेच्छिद्यते , चे , छिद्यते	0	0	1
6.1.77.1	अभ्युदय , अभि , उदय	1	1	1
6.1.77.2	अन्वयः , अनु , अयः	1	1	1
6.1.77.3	कर्त्री , कर्तृ , ई	1	1	0
6.1.78.1	नयनम् , ने , अनम्	0	1	1
6.1.78.2	भवनम् , भो , अनम्	0	1	0
6.1.78.3	नायकः , नै , अकः	0	1	0
6.1.78.4	पावकः , पौ , अकः	0	0	1
6.1.79.1	माण्डव्यः , माण्डो , यः	0	1	0
6.1.79.2	गव्यम् , गो , यम्	0	1	0
6.1.79.3	नाव्यम् , नौ , यम्	0	0	0
6.1.80.1	लव्यम् , लो , यम्	0	0	0
6.1.80.2	भाव्यम् , भौ , यम्	0	0	0
6.1.81.1	जय्य , जे , य	0	0	0
6.1.81.2	क्षय्य , क्षे , य	0	0	0
6.1.82.1	क्रय्य , क्रे , य	0	0	0
6.1.83.1	भय्य , भे , य	0	0	0
6.1.87.1	प्रेक्षणम् , प्र , ईक्षणम्	0	0	1
6.1.87.2	प्रोत्साहः , प्र , उत्साहः	0	0	1
6.1.89.1	उपैति , उप , एति	0	1	1

RULE NO.	EXAMPLE	INRIA	JNU	UoH
6.1.89.2	उपैधते , उप , एधते	0	1	1
6.1.89.3	प्रौढः , प्र , ऊढः	0	1	0
6.1.89.4	प्रैषः , प्र , एषः	0	1	1
6.1.90.1	बहुश्रेयस्यै , बहुश्रेयस्यी , आट् , ङे	0	0	0
6.1.91.1	उपाच्छ्रति , उप , ऋच्छति	0	0	0
6.1.91.2	प्राच्छ्रति , प्र , ऋच्छति	0	0	0
6.1.93.1	गाः , गो , अस्	0	0	0
6.1.93.2	गाम् , गो , अम्	0	0	0
6.1.94.1	प्रेजते , प्र , एजते	0	0	1
6.1.94.1	उपोषति , उप , ओषति	0	0	1
6.1.95.1	कोम् , का , ओम्	0	0	0
6.1.95.2	शिवायोम् , शिवाय , ओम्	0	1	1
6.1.95.3	कदोढा , कदा , ओढा	1	1	0
6.1.95.3	अद्यश्र्यात् , अद्य , अश्र्यात्	0	0	1
6.1.96.1	अयुः , अया , उस्	0	0	0
6.1.96.2	छिन्द्युः , छिन्द्या , उस्	0	0	0
6.1.97.1	पचे , पच , ए	0	0	0
6.1.97.2	पचन्ति , पच , अन्ति	0	0	0
6.1.98.1	घटिति , घटत् , इति	0	0	0
6.1.98.1	झटिति , झटत् , इति	0	0	0
6.1.99.1	पटत्पटदिति , पटत्पटत् , इति	1	0	1
6.1.99.2	पटत्पटेति , पटत्पटत् , इति	0	0	0
6.1.100.1	पटत्पटा , पटत्पटत् , आ	0	0	0
6.1.101.1	उपाचार्यः , उप , आचार्यः	0	0	1

RULE NO.	EXAMPLE	INRIA JNU UoH
6.1.101.2	उपेन्द्रः , उप , इन्द्रः	0 0 1
6.1.102.1	अग्नी , अग्नि , औ	0 0 0
6.1.102.1	वायू , वायु , औ	0 0 0
6.1.102.1	वृक्षाः , वृक्ष , अस्	0 0 1
6.1.104.1	वृक्षौ , वृक्ष , औ	0 0 0
6.1.105.1	कुमार्यौ , कुमारी , औ	0 0 0
6.1.106.1	उपनाहयौ , उपनाही , औ	0 0 0
6.1.107.1	वृक्षम् , वृक्ष , अम्	0 0 1
6.1.108.1	गृहीत , ग्रह , इत	0 0 0
6.1.110.1	अग्नेः , अग्ने , अस्	0 0 1
6.1.112.1	सख्युः , सखि , अस्	0 0 0
6.1.131.1	द्युकामः , दिव् , कामः	0 0 0
6.1.133.1	स्य ते , स्यः , ते	0 0 0
8.3.2.1	सँस्कर्त्ता , सम् , कर्त्ता	0 0 0
8.3.4.1	संस्कर्त्ता , सम् , कर्त्ता	0 0 0
8.3.5.1	सँस्स्कर्त्ता , सम् , स्कर्ता	0 0 0
8.3.5.1	संस्स्कर्त्ता , सम् , स्कर्ता	0 0 0
8.3.6.1	पुँस्कामा , पुम् , कामा	0 0 0
8.3.6.1	पुंस्कामा , पुम् , कामा	0 0 0
8.3.13.1	लीढः , लिढ् , ढः	0 0 0
8.3.16.1	पयःसु , पयर् , सु	0 0 0
8.3.24.1	पयांसि , पयान् , सि	0 0 0
8.3.25.1	सम्राट् , सम् , राट्	0 0 0
8.3.28.1	प्राड्क् शेते , प्राङ् , शेते	0 0 0

RULE NO.	EXAMPLE	INRIA JNU UoH
8.3.30.1	सन्त्सः , सन् , सः	0 0 0
8.3.48.1	कस्कः , कः , कः	0 0 1
8.3.56.1	जलाषाट् , जला , साट्	0 0 0
8.3.58.1	सर्पिःषु , सर्पिः , सु	0 0 0
8.3.59.1	वायुषु , वायु , सु	0 0 0
8.3.60.1	उषितः , उ , सितः	0 0 0
8.3.65.1	अभिषुणोति , अभि , सुनोति	0 0 0
8.3.66.1	विषदिति , वि , सदिति	0 0 0
8.3.67.1	परिष्टभ्नाति , परि , स्तभ्नाति	0 0 0
8.3.68.1	अवष्टभ्यास्ते , अव , स्थभ्यास्ते	0 0 0
8.3.69.1	विष्वणति , वि , स्वनति	0 0 0
8.3.70.1	परिषेवते , परि , सेवते	0 0 0
8.3.72.1	अनुष्यन्दते , अनु , स्यन्दते	0 0 0
8.3.73.1	विष्कन्ता , वि , स्कन्ता	0 0 0
8.3.74.1	परिष्कन्ता , परि , स्कन्ता	0 0 0
8.3.75.1	परिष्कन्द , परि , स्कन्द	0 0 0
8.3.76.1	निष्ष्फुरति , निस् , स्फुरति	0 0 0
8.3.77.1	विष्कम्भिता , वि , स्कम्भिता	0 0 0
8.3.86.1	अभिनिष्टानः , अभिनिस् , स्तानः	0 0 0
8.3.87.1	अभिष्यात् , अभि , स्यात्	0 0 0
8.3.88.1	सुषूतिः , सु , सूतिः	0 0 0
8.3.89.1	निष्णातः , नि , स्नातः	0 0 0
8.3.90.1	प्रतिष्णातः , प्रति , स्नातः	0 0 0
8.3.92.1	प्रष्ठ , प्र , स्थ	0 0 0

RULE NO.	EXAMPLE	INRIA JNU UoH
8.3.94.1	विष्टार , वि , स्तार	0 0 0
8.3.96.1	विष्ठलम् , वि , स्थलम्	0 0 0
8.3.99.1	हरिषेणः , हरि , सेनः	0 0 0
8.3.100.1	भरणिषेणः , भरणि , सेनः	0 0 0
8.3.102.1	निष्टपतिः , निस् , तपतिः	0 0 0
8.3.107.1	अभी षु , अभी , सु	0 0 0
8.3.109.1	पृतनाषाहम् , पृतना , साहम्	0 0 0
8.3.118.1	अभिषसाद , अभि , ससाद	0 0 0
8.3.119.1	न्यषीदत् . नि , असीदत्	0 0 0
8.4.1.1	पूष्णा , पूष् , ना	0 0 0
8.4.2.1	रामेण , रामे , न	0 0 0
8.4.12.1	वृत्रहणः , वृत्र , हनः	0 0 0
8.4.14.1	परिणमति , परि , नमति	0 0 0
8.4.15.1	प्रहिणुतः , प्र , हिनुतः	0 0 0
8.4.16.1	प्रवपाणि , प्रवप , आनि	0 0 0
8.4.17.1	प्रणिपतति , प्र , निपतति	0 0 0
8.4.18.1	प्रणिपचति , प्र , निपचति	0 0 0
8.4.19.1	प्राणिति , प्र , अनिति	0 0 0
8.4.20.1	प्राण् , प्र , अन्	0 0 0
8.4.22.1	प्रहण्यते , प्र , हन्यते	0 0 0
8.4.23.1	प्रहण्वः , प्र , हन्वः	0 0 0
8.4.27.1	रक्षाणः , रक्षा , नः	0 0 0
8.4.28.1	प्रणः , प्र , नः	0 0 0
8.4.29.1	परिमाणम् , परि , मानम्	0 0 0

RULE NO.	EXAMPLE	INRIA JNU UoH
8.4.30.1	प्रयापणम् , प्र , यापनम्	0 0 0
8.4.31.1	प्रकोपणम् , प्र , कोपनम्	0 0 0
8.4.32.1	प्रेङ्खणम् , प्र + इङ्खनम्	0 0 0
8.4.33.1	प्रणिक्षणम् , प्र , निक्षणम्	0 0 0
8.4.37.1	वृक्षान् , वृक्ष , अन्	0 0 1
8.4.40.1	यज्ञः , यज् , नः	0 0 0
8.4.41.1	दुष्टः , दुस् , टः	0 0 0
8.4.41.2	विष्णुः , विष् , नुः	0 0 0
8.4.41.3	पेष्टा , पेष् , ता	0 0 0
8.4.41.4	उड्डीनः , उद् , डीनः	0 0 0
8.4.46.1	कर्त्तव्यम् , कर् , तव्यम्	0 0 0
8.4.46.2	कर्तव्यम् , कर् , तव्यम्	0 0 0
8.4.49.1	अक्षदर्शः , अक्षदर् , शः	0 0 1
8.4.53.1	वृद्धिः , वृध् , धिः	0 0 1
8.4.58.1	शङ्का , शं , का	1 0 1
8.4.61.1	उत्थानम् , उद् , स्थानम्	0 0 0
8.4.65.1	कृष्णर्धिः , कृष्णर् , ध्धिः	0 0 0

Total Number of Words = 135

INRIA Sandhi Splitter Performance = 6/135 * 100 = 4.4 %

JNU Sandhi Splitter Performance = 14/135 * 100 = 10.4 %

UoH Sandhi Splitter Performance = 27/135 * 100 = 20 %

APPENDIX B

Code to Extract from Sandhi-Extracted Corpora Sandhied Words Which Do Not Have the Effect of Sandhi in their Last Letter

```
#include<fstream>
#include<iostream>
#include<string>
using namespace std;
int main()
{ifstream fin;
 ofstream fout1;
 ofstream fout2;
 fin.open("Input.txt");                    //File to be processed is saved as Input.txt
 fout1.open("Intermediate.txt");
 char sentence[10000];
 string s;
 string words[10000];
 int i;
 int j=1;
 int k=1;
 while(!fin.eof())
 {fin.getline(sentence,10000);
  fout1<<sentence<<" % "<<"\n";
 }
 fin.close();
 fout1.close();
 fin.open("Intermediate.txt");
 fout1.open("Output.txt");    // File containing sandhied words
 fout2.open("Sandhi_Splits.txt"); // File containing sandhied words with their splits
 while(!fin.eof())
 {   i=1;
```

```
    fin>>s;
    words[0]=s;
    while(s!="%")
    {fin>>s;
    words[i]=s;
    i++;
    }
    if(words[1]=="=>")
    {fout1<<j++ <<". "<< words[0]<<"\n";

    fout2<<k++ <<". "<< words[0]<<"  ,  "<< words[2]<<"\n";
    }
}
return 0;
}
```

APPENDIX C

Python Code to Automate the Process of Getting the Output from a Website by taking Inputs from a File

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import requests
from bs4 import BeautifulSoup

# Creating word list
words = []
with open("Output.txt") as file:
    for word in file.readlines():
        words.append(word.strip())

#  Any One of These Three Tried at One Time
# URL endpoint for http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp
url = "http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp?itext="

# OR URL endpoint for http://sanskrit.inria.fr/DICO/reader.fr.html
url =" http://sanskrit.inria.fr/cgi-bin/SKT/sktgraph?lex=SH&st=t&us=f&cp=t&text="

# OR URL endpoint for http://52.25.246.194/scl/
url =http://52.25.246.194/scl/
for word in words:
    new_url = url + word
    r = requests.get(new_url)
    print (r.content)
```

APPENDIX D

Corpus Used to Evaluate Sandhi Splitters for Sandhi Found in Literature

Book Name : मञ्जूषा
Author : श्री कृष्णभट्ट
Project Name : Development of Tagged Corpora for Sanskrit (DTCS) CIIL Project
Center : DEPARTMENT OF SANSKRIT STUDIES, SCHOOL OF HUMANITIES, UNIVERSITY OF HYDERABAD
Typed by : लक्ष्मी नारायण
Proofcheck by : शिवरामकृष्णा और श्रीमहालक्ष्मी
Sandhi Split by : शिवानन्द शुक्ल

S.No.	Example	JNU	INRIA	UoH
1.	मञ्जूषेति , मञ्जूषा+इति	0	1	1
2.	कटाक्षयन्नाह , कटाक्षयन्+आह	0	0	1
3.	ज्ञानेऽपि , ज्ञाने+अपि	0	1	0
4.	स्वीकार्यैत्यभिप्रायवानाह , स्वीकार्यैत्यभिप्रायवान्+आह	0	1	0
5.	चेति , च+इति	1	1	1
6.	तयोर्लक्षणायामन्तर्भावसम्भवादिति , तयोः+लक्षणायाम्+अन्तर्भावसम्भवात्+इति	0	1	0
7.	समवायेनाकाशोपस्थितिसम्भवाद् , समवायेन+आकाशोपस्थितिसम्भवाद्	0	0	0
8.	वृत्येति , वृत्या+इति	0	1	1
9.	एतच्च , एतत्+च	0	1	1
10.	समवायादिनाऽऽकाशआद्युपस्थितिर्न , समवायादिनाऽऽकाशाद्युपस्थितिः+न	0	0	0
11.	लक्षणाऽनधीनत्वादिति , लक्षणाऽनधीनत्वात्+इति	0	0	0
12.	तामादायार्थव्यवहारवारणाय , ताम्+आदाय+अर्थव्यवहारवारणाय	0	1	0
13.	वाऽन्वयि , वा+अन्वयि	1	1	0
14.	लक्ष्यस्यार्थत्वव्यवहारानुरोधेन , लक्ष्यस्य+अर्थत्वव्यवहारानुरोधेन	0	1	0
15.	शक्त्येति , शक्त्या+इति	0	1	1
16.	नोक्तम् , न+उक्तम्	0	1	1
17.	स्यादतः , स्यात्+अतः	0	1	1
18.	सर्वदाऽसत्त्वेऽपि , सर्वदा+असत्त्वे+अपि	0	1	0

19. व्यवहारानुपपत्तिरस्त्येवेति , व्यवहारानुपपत्तिः+अस्ति+एव+इति 0 1 0

20. तादृशप्रतिपत्तेर्लक्षणाऽधीनार्थव्यवहारे , तादृशप्रतिपत्तेः+लक्षणाऽधीनार्थव्यवहारे 0 1 0

21. वस्तुतस्तु , वस्तुतः+तु 0 0 1

22. व्याप्यवृत्तितयेदानीमित्यस्यानन्वयाद् , व्याप्यवृत्तितया+इदानीम्+इत्यस्य+अनन्वयाद् 0 0 0

23. इत्यपीष्यत , इत्यपि+इष्यत 0 1 1

24. तात्पर्यवन्न , तात्पर्यवत्+न 0 1 1

25. वृत्तित्वमिति , वृत्तित्वम्+इति 0 1 1

26. विभजते-तत्रेति , विभजते-तत्र+इति 0 1 1

27. पारिभाषिकमिति , पारिभाषिकम्+इति 0 1 1

28. अर्थस्तु , अर्थः+तु 0 1 1

29. लक्ष्यपरिभाषितयोर्व्यवच्छेदार्थः , लक्ष्यपरिभाषितयोः+व्यवच्छेदार्थः 0 1 0

30. वाच्यत्वव्यवहारापत्तिरर्थस्य , वाच्यत्वव्यवहारापत्तिः+अर्थस्य 0 1 0

31. चेश्वरेच्छीया , च+ईश्वरेच्छीया 0 0 1

32. सैव , सा+एव 0 1 1

33. कर्तृत्वमिति , कर्तृत्वम्+इति 0 1 1

34. तिङमभिप्रेत्यः , तिङम्+अभिप्रेत्यः 0 0 1

35. पदमिति , पदम्+इति 0 1 1

36. धात्वर्थीभूतेति , धात्वर्थीभूता+इति 0 1 1

37. फलस्यैवात्मनेपदाद्यर्थत्वात् , फलस्य+एव+आत्मनेपदाद्यर्थत्वात् 0 1 0

38. पातित्वंतदाश्रयजनकत्वे , पातित्वम्+तदाश्रयजनकत्वे 0 1 0

39. गौरिति , गौः+इति 0 1 1

40. वैपरीत्यमिति , वैपरीत्यम्+इति 0 0 1

41. गौरुच्यत , गौः+उच्यत 0 1 0

42. वक्तीत्यादौ , वक्ति+इत्यादौ 0 1 1

43. निरूक्तधात्वर्थस्य , निरूक्तधात्वर्थ+अस्य 0 0 1

44. कर्तृत्वमाख्यातादिना , कर्तृत्वम्+आख्यातादिना 0 1 1

45. साधूनेव , साधून्+एव 0 1 1

46. नासाधूनिति , न+असाधून्+इति 0 1 1

47. निषेधानुपपत्तिरिति , निषेधानुपपत्तिः+इति 0 1 1

48. एवमिति , एवम्+इति 0 1 1

49. तादृशेच्छयेति , तादृशेच्छया+इति 0 1 1

50. तत्तदर्थविषयकबोधजनकत्वप्रकारकत्वावच्छिन्नेच्छावृत्तिजनकताकभगवदुच्चारणजन्यत्वमर्थः , तत्तदर्थविषयकबोधजनकत्वप्रकारकत्वावच्छिन्नेच्छावृत्तिजनकताकभगवदुच्चारणजन्यत्वम्+अर्थः 0 0 0

Total Number of Cases : 50

JNU Sandhi Splitter Performance = 2/50 = 4 %

INRIA Sandhi Splitter Performance = 39/50 = 78 %

UoH Sandhi Splitter Performance = 31/50 = 62 %