Sanskrit Sandhi Splitting and Merging: Benchmark Datasets and Evaluation

Anonymous EACL submission

Abstract

Sanskrit is one of the oldest and culturally rich languages that originated in 200 B.C. Learning the fundamentals of an old language will provide insights to the overall community of language modeling. There are two broad ways of performing language modeling: (i) Grammar model and (ii) probabilistic model. Words in Sanskrit language are formed by a process called Sandhi merging, which is a combination of two or more morphemes (component words). The reverse process of getting back the component words from the full words is known as Sandhi splitting. It is believed by many scholars that though *Pāṇini's* has written a grammar for Sanskrit words in his monumental work called *Aṣṭādhyāyī*. Analyzing and understanding word formation in Sanskrit is conceptually generic providing us with a framework to write grammars for other languages. However, the publicly available dataset such as the University of Hyderabad dataset is very erroneous making it challenging to evaluate automated algorithms. The primary contribution of this research work is to create a set of golden datasets with manually created ground truth made public for promoting open research in this fundamental problem. Further as the second contribution, we present the results of three publicly available tools for performing Sandhi splitting and Sandhi merging and benchmark their performance. We finally discuss the characteristics of each of the tool and emphasize the need for further research.

1 Introduction

The basic constituents of a language are its characters which are joined together to form morphemes, which is the meaning fundamental part of a language that cannot further divided. Morphemes are combined together to form words, words are combined to form sentences, and sentences are combined to form large texts. The purpose of a language model is to understand the underlying rules and structure that comprises a language. There are two broad approaches for constructing a language model as follows:

- **Grammar based model:** This approach is generally used to model complex but deterministic grammar based languages. The learning models can be rule based or typically require less training data.
- **Probabilistic based model:** When a deterministic grammar is not available, a probabilistic language model can be learnt using a large corpus of data.

Sanskrit is one of the oldest languages designed by mankind having its origin in the Indo-Aryan civilization from 200B.C. Sanskrit has a rich tradition of poetry and literature works written, making it an important linguistic study. Being one of the oldest language, word formation in Sanskrit can be defined by set of deterministic rules and follows a defined structure. *Aṣṭādhyāyī* (meaning a collection of eight books) by *Pāṇini* is the source of Sanskrit's grammar, syntax, and semantics. The importance of *aṣṭādhyāyī* is three fold. The first one, as is well known, as an almost exhaustive grammar for any natural language with meticulous details yet small enough to memorize. Though *aṣṭādhyāyī* is written to describe the then prevalent Sanskrit language, it provides a grammatical framework which is general enough to anal-

yse other languages as well. This makes the study of *aṣṭādhyāyī* from the point of view of concepts it uses for language analysis important. The third aspect of *aṣṭādhyāyī* is its organization. The set of less than 4000 *sūtra* is similar to any computer program with one major difference the program being written for a human being and not for a machine thereby allowing some non-formal or semi-formal *sūtra* which require a human being to interpret and implement them. Nevertheless, we believe that the study of *aṣṭādhyāyī* from programming point of view may lead to a new programming paradigm because of its rich structure. In this way, *Pāṇini's* created a brief and immensely dense work.

¡¡Write about Sandhi and splitting and merging¿¿

The word "sandhi", more properly written as *saṃdhi*, means "junction" or "combination." It refers to the "combination" of two sounds that sit next to each other. The word "sandhi" was borrowed into English, where it refers to the same sorts of changes in any language. So, we can talk about English sandhi, Chinese sandhi, Latin sandhi, and so on.

The concept of "sandhi" might seem strange to you. Fortunately, English shows some signs of sandhi rules. For example, consider three English words that are borrowed from Latin: "indirect," "impossible," and "illuminate." Each of these words starts with "in," but the "n" of this "in" has changed to more closely match the letter that follows it. Note that the pronunciation and spelling of these words have both changed. The same thing happens in Sanskrit, and if sandhi is applied in a text, we must write out all of the changes.

1.1 Types of Sandhi

-

Sandhi's can be classified based on the place either within a word, or between two words that it takes place. Thus Sandhi are of broadly two types

1. **Internal Sandhi:** Sanskrit grammar has three granular meaningful units (morphemes) prefixes, roots and suffixes, every word in Sanskrit can be derived from these units. When these units combine to form a word, *saṃhitā* applies and certain changes take place. This is known as internal sandhi.

   For example, *bho* (changed form of verb *bhū* bhuu (to be)) + *anam* (anam, a noun forming suffix) → *bhavanam* (bhavanam, being) is a case of internal sandhi.

2. **External Sandhi:** On the otherhand when sandhi takes place between two words either morpheme or a compound wor, it is known as external sandhi.

   For example, *tau* ("both of them") + *ekadā* ("once") → *tāvekadā* ("both of them once") involves external sandhi.

Sandhi's also can be further classified depending on type of the last letter of the word and and the first letter of its adjoining word, which are participating in forming the sandhi. The letters can be either vowels, consonants or visarga based on these sandhis can be classified into the following three categories:

1. **Vowel Sandhi:** Both letters are vowels e.g. *hima* ("snow") + *ālayaḥ* ("house") → *himālayaḥ* ("house of snow")

2. **Consonant Sandhi :** At least one of the two letters is a consonant, e.g. *vṛkṣa* ("tree") + *chāyā* ("shade") → *vṛkṣacchāyā* ("shade of tree")

3. **Visarga Sandhi :** A visarga combines with a vowel or a consonant, e.g. *punaḥ* ("again") + *janma* ("birth") → *punarjanma* (punarjanma, rebirth)

The process of breaking a sandhied word into morphemes it is made up of is known as sandhi splitting. A tool which can perform this task is referred to here as a sandhi splitter. The reverse process of combining two morphemes to form the compound word is known as Sandhi Merging.

Implementing sandhi merging rules are reasonably simple when compared to implementing sandhi splitting tools. In case of sandhi merging tools one need to implemnt the transformation of the letters in case of most of the sandhi rules when concerned. But when we consider sandhi splitting we need to identify the position where exactly the split needs to occur. Thats the resaon most of the splitter tries to predict muliple split instead of suggesting only one split. In our evaluation we consider a split to be correct in case any of the suggested split is correct.

Next section talks about top three state of the art sandhi merging and splitting tools that are available, and their underlying algorithms for implemnting these algorithms.

2 Exisiting Sandhi Splitting and Merging tools:

The process of breaking a sandhied word into the original units it is made up of is known as sandhi splitting. A tool which can perform this task is referred to here as a sandhi splitter. In the same lines the tools which can merge two words to form the compound word is known as sandhi merging tool. There has been considerable amount of research in the field of sandhi splitting and merging. As of now, three distinct sandhi splitters and merging tools are available:

2.1 Sanskrit Sandhi Analyzer and Splitter

[1] This is a vowel-sandhi splitter. The other two kinds of sandhis (consonant and visarga) are not split by this. This was developed at Jawaharlal Nehru University under the guidance of Prof. Girish Nath Jha (Professor, Computational Linguistics, Special Centre for Sanskrit Studies). This is available at `http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp`.

Architecture of the system is as follows:

1. **Pre-processing:** Pre-processing includes marking the punctations, checking the word length to determine whether we can split the words or not, and searching in the corpus if the intput word is already an instance of the examples to avoid processing.

2. **Subanta analysis:** This analysis includes splitting the noun phrases into its constituents base (prtipadika) and case terminations (kraka-vacana-vibhakti). Sandhi analyzer will process the base words only.

3. **Fixed List checking:** After subanta analysis, the input text has been segmented into base and affixes. Now only base words will be processed for sandhi analysis. But before that these words will be checked into a Dictionary (Monier Williams Sanskrit Digital Dictionary), place name list and noun list. The words found in these resources, will be exempted from sandhi processing. These different files are merged to one text file which is named as lexicon.

4. **Sandhi Analysis:** The segmenter class will check the vowel sandhi rule - base in the database to mark the resultant sandhi sounds (marker) for potential splitting and to identify the sandhi patterns for viccheda , corresponding to the marked sound.

5. **Result generator:** At each step of marker and pattern identification, the class will check the segmented words in the lexicon to generate the result. For this purpose, it will use Dictionary (Monier Williams Sanskrit Digital Dictionary) and customized Sanskrit corpora as the linguistic resources. To be a valid segmentation , both the segments must be available in either of the linguistic resources. If the word has more than one sounds marked for sandhi , then only the first word must be present in either of the linguistics resources . The remaining string in this case will continue with the process of rule pattern matching, splitting and search in the linguistic resources.

Here after this tool is known as JNU tool for comparing the results.

2.2 Sandhi-Splitter

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 considered here because it is the latest version.

2.2.1 Segmentation Algorithm

The basic outline of the algorithm is(**?**):

1. Recursively break a word at every possible position applying a sandhi rule and generate all possible candidates for the input.

2. Pass the constituents of all the candidates through the morphological analyser.

[1] `http://sanskrit.jnu.ac.in/rstudents/mphil/sachin.pdf`

3. Declare the candidate as a valid candidate, if all its constituents are recognised by the morphological analyser, and all except the last segment are compounding forms.

4. Assign weights to the accepted candidates and sort them based on the weights as defined in the previous subsection.

5. The optimal solution will be the one with the highest weight.

Here after this tool is referred as UOH for simplicity purpose.

2.3 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`.

The flow of Sanskrit parser is as follows

At first the lexicon is analysed, to gather stems and their morphological parameters, such as permitted genders of nominal stems, allowed present classes (*gaṇa*) and attested preverbs for roots.

In second phase, more stem generation occurs for roots, accounting for the various tenses/moods (*lakāras*), as well as absolutives and in nitives, but also participles (adjectival *kṛdantas*) in 10 varieties.

Finally, inflexional morphology paradigms derive the in ected forms according to the morphological parameters, some of which being read from the lexicon, some of which being de ned in speci c tables.

It is not simple to relate our paradigmatic derivations and the operations of Paninian grammar. Some stem operations relate to morphophonetic operations well identified in the central rules (*sūtras*), such as taking the phonetic grades of *guṇa* or *vṛddhi*. Some are dispersed in various parts of the grammar, such as stem substitutions. Still others, such as retroflexion, occur in the final section (*tripādī*) of the *aṣṭādhyāyī*, which comprises a list of rewrite rules which are applied iteratively at the end of the derivation process, in some kind of phonetic smoothing" phase. Since the forms stored in our data banks are to be matched to the surface realization of the sentence, this phonetic smoothing must be applied at the time of generation of these forms. Some careful analysis must be done in order to understand the mutual interaction of the retroflexion rules and of the external sandhi rules (which are used in segmentation in the Heritage reader). This analysis will be given in section 4.4 below. The main discrepancy between the Paninian processes and our reader operations is in the order of rewritings. Often the stras relevant to a given operation are dispersed in different sections of the *aṣṭādhyāyī* , and thus it is hard to give the trace of the needed stras . However, in many cases, one can recognize the stra operations from the computer program. A complete analysis, in the speci c case of future passive participles .

3 Methodology of Evaluation:

The evaluation was done in two ways: Rule-based and Literature-based.

3.1 Rule-based

The three splitters are evaluated against a corpus which contains at least one example for each of the Paaninian sandhi rules. This brings out how many rules are actually implemented by the splitters. There are 282 examples against 271 rules. This evaluation was done both manually and using a tool developed for this purpose. The corpus is available at ..

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.

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, the slightly incorrect split is still considered as correct. An example is *nayanam* (the act of directing) whose correct split is *ne* (changed form of the root nee meaning direct) + *anam* (a noun forming suffix) but *ne* + *anama* is also considered to be correct, even though the last letter does not have a *halanta*. Another example is *prauḍhaḥ*(fully developed, aged, etc) where both *pra* + *ūḍhaḥ* and *pra*+ *ūḍha* are considered correct. However, the automated evaluation rejects the latter result in each of these cases.

This privilege has not been given to internal sandhi cases, which if the splits are only slightly wrong, there are not considered. This is because internal sandhi is between prefixes, roots and suf-

fixes and small mistakes in each of these has the potential to change the meaning. There are some rules which govern combination of letters that may themselves be the results of application of other rules. An example is *vṛkṣas* (vrik.sas, tree) + *śete* ("sleeps") → *vṛkṣaśśete* ("tree sleeps") where the split *vṛkṣaḥ* +*śete* gives the original form. So, even though the corresponding rule has to do with change of *s* to *ś*, the presence of *visarga* is duly considered correct.

Evaluation results are summarized in the following table

Splitting tool	**Manual**	Automated
JNU	12.4%	11.4%
UoH	26.6%	18.1%
INRIA	19.5%	14.5%

Table 1: Evaluation Results

There is a significant difference in the results for the UoH and the INRIA splitter in the two modes of evaluation. The results of manual evaluation are detailed below as per the type of the sandhi rules :

Splitter	**External Sandhi Cases (132)**	**Internal Sandhi Cases (150)**	**Overall**
JNU	21 (15.9 %)	14 (9.3 %)	12.4 %
UoH	48 (36.4 %)	27 (18 %)	26.6 %
INRIA	49 (37.1 %)	6 (4 %)	19.5 %

Table 2: Evaluation Results

Cases not detected by any Splitter- 62 (46.9

3.1.1 Analysis of Results

- A large number of rules have not been implemented, even if we leave aside those cases where the examples themselves can never be the first two words to start with, i. e. the cases that represent the second or subsequent stages of sandhi between two words, before the final word is obtained.

- The internal sandhi phenomenon seems to have been neglected to a great extent.

Sandhi Merging tool results:

3.2 Literature-Based Evaluation:

This takes into account the sandhied words which appear in actual literature. This evaluation is useful because the sandhi splitters are more likely to be used to split such words. If the performance of the splitters in splitting these words is satisfactory, one may consider neglecting the rules which these splitters have not been able to implement, because those rules may not be so frequent in use. However, a bad performance in this context is a cause of actual concern. Five different corpora were used for this purpose:

Splitting tool	**Manual**	Automated
JNU	x%	20.92%
UoH	x%	32.97%
INRIA	x%	51.77%

Table 3: Evaluation Results

1. 150-word corpus
2. Manually created Bhagvad Gita corpus containing nine chapters
3. Dictionary-filtered UoH corpora
4. Word-length filtered A.s.taadhyaayii corpus

3.2.1 150-word corpus

This was created from 11 different texts. This has 50 examples from one text, and 10 examples each from the other ten texts. This is smaller in size compared to the other literature corpora, hence the evaluation for first was done both manually and using the automated tool. The other three were evaluated with the help of the tool only because of their much larger size.

Splitter	**Manual Evaluation Results**	**Automated Evaluation Results**
JNU	$19/150 * 100 = 12.66\%$	$15/150 * 100 = 10\%$
UoH	$105/150 * 100 = 70\%$	$98/150 * 100 = 65.33\%$
INRIA	$127/150 * 100 = 84.66\%$	$94/150 * 100 = 62.66\%$

Table 4: Evaluation Results

The results of automated evaluation are reported below. The difference of .. between the two sets of results is henceforth considered as the error margin. This is expected to be the boost in performance of the three sandhi splitters if the corpora considered further were to be manually eval-

Merger	Manual Evaluation Results	Automated Evaluation Results
JNU	$x\%$	$75/150 * 100 = 50\%$
UoH	$x\%$	$129/150 * 100 = 86\%$
INRIA	$x\%$	$131/150 * 100 = 87.33\%$

Table 5: Evaluation Results

uated. Only automated evaluation has been done for them.

3.2.2 Manually created Bhagvad Gita corpus containing nine chapters

The sandhi-split Bhagvad Gita corpus at the UoH website had several limitations which made it unfit to be used for the purpose of automated evaluation. For example, within the first two chapters, out of the total of 431 sandhi cases, there were 41 typos, 92 cases of insufficient splits and 10 cases of even wrong splits.

Thus, a new corpus was manually created. This was done for half of Gita, i.e. for the 9 chapters.

Results of Automated Evaluation:

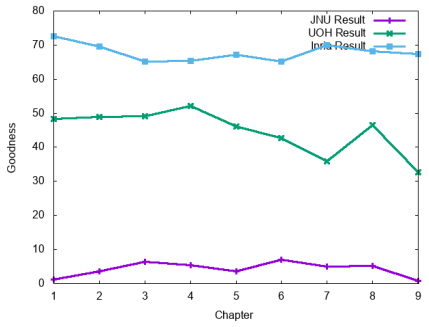

Figure 1: Screenshot of the built QA system showing the orchestration flow for the question - "problem with transfer order"

3.2.3 Dictionary-filtered UoH corpora

The UoH website has 39 sandhi-split corpora but they are not fully correct. There are cases of typos, insufficient splits and even wrong splits. Therefore, they were not directly used for the purpose of automated evaluation. The corpora contained thousands of words in total, and a strategy was worked out to create a subset of them that had no errors.

	JNU	UoH	INRIA
A_1 **[157]**	2 (1.27 %)	76 (48.41 %)	114 (72.61 %)
A_2 **[270]**	10 (3.70 %)	132 (48.89 %)	188 (69.63 %)
A_3 **[169]**	11 (6.51 %)	83 (49.11 %)	110 (65.09 %)
A_4 **[165]**	9 (5.45 %)	86 (52.12 %)	108 (65.45 %)
A_5 **[113]**	4 (3.54 %)	52 (46.02 %)	76 (67.26 %)
A_6 **[187]**	13 (6.95 %)	80 (42.78 %)	122 (65.24 %)
A_7 **[120]**	6 (5.00 %)	43 (35.83 %)	84 (70.00 %)
A_8 **[116]**	6 (5.17 %)	54 (46.55 %)	79 (68.10 %)
A_9 **[135]**	1 (0.74 %)	44 (32.59 %)	91 (67.41 %)
Total [1432]	62 (4.33 %)	650 (45.39 %)	972 (67.88 %)

Table 6: Evaluation Results

The strategy involved checking whether the splits could be located in some dictionary. Five dictionaries were used for this purpose. We restricted ourselves only to such cases where the splits could be located, even though they may be many cases of correct splits where the splits themselves cannot be located in the dictionary, because of various reasons (dictionaries may not contain all the declensions/ conjugations of a word, nor they are expected to). Further to check that the sandhied words in such cases did not have typos, a sandhi tool was used to do sandhi of the splits identified in the dictionary and check whether the result in each case matched with the sandhied word as given in the corpus. If the two words did not match for a particular case, it was neglected.

Just to illustrate this, we consider the five cases listed below.

tumulo vyanunādayan → *tumulaḥ+vi+anunādayan*

sarvānbandhūnavasthitān → *sarvān + bandhūn*

	JNU	UoH	INRIA
A_1 **[157]**	47 (29.94 %)	90 (57.32 %)	87 (55.41 %)
A_2 **[270]**	71 (26.30 %)	181 (67.04 %)	174 (64.44 %)
A_3 **[169]**	52 (30.77 %)	126 (74.56 %)	111 (65.68 %)
A_4 **[165]**	47 (28.48 %)	121 (73.33 %)	105 (63.64 %)
A_5 **[113]**	29 (25.66 %)	77 (68.14 %)	64 (56.64 %)
A_6 **[187]**	47 (25.13 %)	114 (60.96 %)	100 (53.48 %)
A_7 **[120]**	29 (24.17 %)	64 (53.33 %)	53 (44.17 %)
A_8 **[116]**	30 (25.86 %)	67 (57.76 %)	63 (54.31 %)
A_9 **[135]**	27 (20.00 %)	65 (48.15 %)	55 (40.74 %)
Total [1432]	379 (26.47 %)	905 (63.20 %)	812 (56.70 %)

Table 7: Evaluation Results

+ avasthitān
śabda iva → śabdaḥ+ iva
nārhati → na + arhati
astamito bhagavān → astam + itaḥ + bhagavān

The first two cases were not included because at least one word in each of the splits could not be located in any of the dictionaries used for this purpose. The last three cases were considered for the purpose of evaluation.

Results: Total Number of Cases: 18,326

Splitter	No. of Cases Correctly Identified
JNU	3214 (17.53 %)
UoH	11405 (62.23 %)
INRIA	13416 (73.20 %)

Table 8: Evaluation Results

3.2.4 Word-length filtered *āstaadhyaayi* corpus

Many *sutrā* (rules) of *pāṇini* themselves contain many sandhied words. All the sutras with their splits are available at . This was found to be another good source which could be used for the evaluation of the three splitting tools. However, even this source suffered with the limitation of insufficient splits. Moreover, a very significant number of splits could not be expected to be located in any dictionary, because these were the forms of the different sets/ *mahesvara sutrā* that *pāṇini* uses to codify Sanskrit grammar, syntax and semantics. Hence, the strategy used in the previous case to limit to cases of correct splits could not be applied in this case.

Since the problem initially arose because of cases of insufficient splits, another strategy was worked out. The splits which can undergo further splitting themselves are likely to be of greater length than those cases in which further splitting is not possible. The larger the length of the splits, the more likely they are to undergo further splitting. All those cases where the length of the splits was less than a specified word length were analysed together and the results were noted for different values of the word lengths-10,20,30, 40 and 50.

The following five examples are used here for the purpose of illustration. At least one of the splits in each of the first two cases is considerably long, and further splitting is evident. When the word length is reduced, the possibility of further splits is also reduced, though not eliminated.

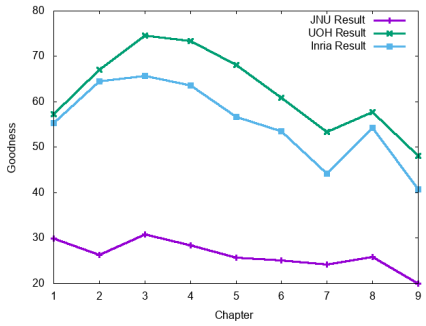

Figure 2: Screenshot of the built QA system showing the orchestration flow for the question - "problem with transfer order"

Merger	No. of Cases Correctly Identified
JNU	6517 (35.56 %)
UoH	12895 (70.36.2 %)
INRIA	12611 (68.81 %)

Table 9: Evaluation Results

So, in the next two cases, though the word length is reduced, the first split of the third case and the second split of the fourth case can themselves be further split. It is only for the last case that further splitting is not possible.

prathamacaramatayālpārdhakatipayanemāśca → *prathamacara-matayālpārdhakatipayanemāḥ+ca*
udupadhādbhāvādikarmaṇoranyatarasyām → *udupadhāt+bhāvādikarmaṇoḥ+anyatarasyām*
taddhitaścāsarvavibhaktiḥ → *taddhitaḥ+ca+ca+asarvavibhaktiḥ*
vṛddhirādaic → *vṛddhiḥ+ādaic*
vija iṭ → *vijaḥ+iṭ*

Results on *āstaadhyaayi* Total Number of Sutras 3,959 Sutras where Sandhi Split Applicable 2,700

No. of letters (≤)	Samples	JNU	UoH
10	93	4(4.3)	21(22.6)
20	571	10(1.75)	100 (17.5)
30	1512	17(1.12)	226(14.9)
40	2045	18 (0.88)	263(12.86)
50	2302	18(0.78)	263(11.42)
All	2700	18 (0.66)	263 (9.74)

Table 10: Evaluation Results

4 Sandhitype based analysis

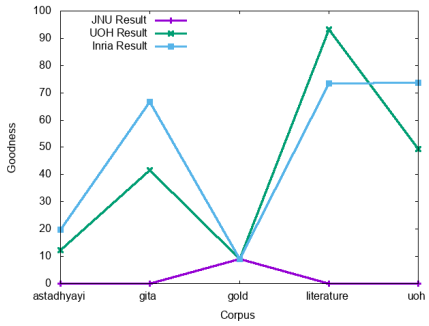

Figure 3: Visarga

5 Analysis of Errors

Literature-Based Evaluation The literature based evaluation results are also not very impressive. The cases were looked into and the reasons behind the poor performance can be categorised as follows:

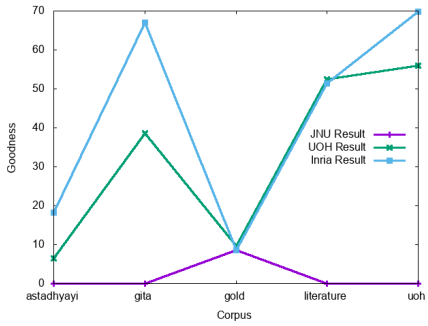

Figure 4: Consonant

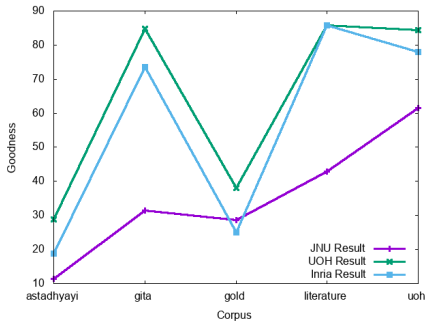

Figure 5: vowel

5.1 Rules not Implemented

It seems that some of the rules which are even frequent used have not been implemented by one or more of the three splitters. For example, the visarga of *saḥ* and *eṣaḥ* is elided optionally when any letter other than *a* follows it, and there are many such cases of this elision, for example, in Srimad Bhagvad Gita. But none of the three splitters is able to undo this elision to get back the visarga. For example, none of the three splitters is able to do the following split

sa yogī → *saḥ* (that) + *yogī* (who practises yoga)

It will be incorrect to say that rules of elision, in general, are not implemented by any of the three splitters. For example, the split of

bālakā hasanti → *bālakāḥ* (boys) + *hasanti* (laugh) is detected correctly by INRIA.

5.2 Optional Rules

There are some rules which are optional in nature, and less frequently used. For example, when *e* at the end of a pada is followed by a vowel, the *y* of *ay* into which it changes can be optionally elided. The resultant form after elision is less common, but we do have cases in Srimad Bhagvad Gita, for example, of this kind. The following case is an example where none of the three splitters is able

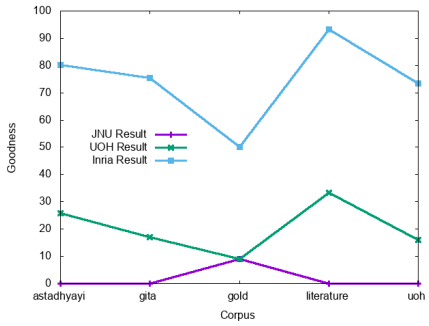

Figure 6: Visarga Merge results

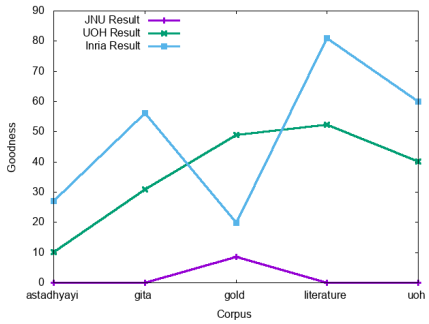

Figure 7: Consonant Merge results

to detect the correct split.

vartanta iti → *vartante* (exist) + *iti* (this)

[Had the optional rule not been applied,*vartante* +*iti* would have led to *vartantayiti*].

5.3 Cascading split effect

There are some rules in which the effect of combination of two words is not restricted to the change in sound at the extreme boundaries of the two words (last sound of first word + first sound of second word). Other letters can also get affected. For example, in

uttara (north) + *ayana* (movement) → *uttarāyaṇa* ("northward movement", refers to movement of Sun towards Tropic of Cancer), the *r* of *uttara* causes the change of *n* of *ayana* into *ṇ*. In absence of *r*, no such change takes place in the case of

dakṣiṇa (south) + *ayana* (movement) → *dakṣiṇāyana* (southward movement, refers to movement of Sun towards Tropic of Capricorn) The three sandhi splitters do not seem to have taken care of such changes. So, while they are able to split *dakṣiṇāyana* correctly, the same is not true for *uttarāyaṇa*.

Another example is the case of change of *s* into *ṣ* when it is immediately preceded by some vowels

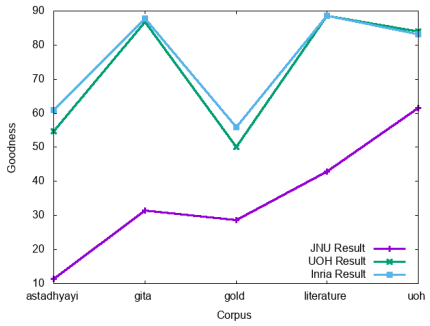

Figure 8: vowel merge

(for example, *i*, and this *ṣ* changing its subsequent letter because of another sandhi rule, for example *th* into *ṭh*. Thus, we have the examples of:

prati + *sthita* → *pratiṣṭhita* ("well-established")

and *yudhi* + *sthiraḥ* → *yudhiṣṭhiraḥ* (one who is stable in war) where the sandhied forms are **not split by any of the three splitters**.

5.4 Multiple splits

The process of sandhi splitting involves splitting the sandhied word at different potential locations, and validating the splits to check which one of them is correct. If the set used for validation is not complete, even correct splits may sometimes not be validated. For example, in

a (not) + *chedyaḥ* ("solvable", "penetrable") → *acchedyaḥ* ("not solvable/penetrable")

the fact that none of the three splitters has been able to split the sandhied word may have to do with the possiblity that may *a* not have been validated as a proper split.

5.5 Compounding effect

The process of compounding, due to which words come together without necessarily their being a change when they merge, also creates problems. While the UoH and the INRIA tools do have the provision of decompounding along with sandhi splitting, the JNU splitter does not have a way to do both together. For example,

lakṣyasyārthatvavyavahārānurodhena → *lakṣyasya* + *arthatvavyavahāra* + *anurodhena*

The second split is not validated without decompounding, and thus even though, only vowel sandhis are involved, the JNU splitter is not able to correctly split the word. Even the INRIA and UoH splitters are not always able to get around this problem. For example, none of the three splitters is able to detect this:

prapañce'vāntaravibhāgapravibhāgabhinnānantapadārthasaṅkule'pi
→ *prapañce + ava + antara-vibhāga-pravibhāga-*
bhinna + ananta-pada + artha-saṅkule + api