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 analyse 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.

With a rich deterministic grammar supporting the Sanskrit language, the formation of each word also happens in a structured manner governed by set of rules called Sandhi. Morphemes are the basic unbreakable morphological units that forms the fundamental building blocks for words. The process of merging two or more morphemes to form a word in Sanskrit is called as Sandhi merging. On the other hand, the process of breaking a word into its constituent morphemes is called Sandhi splitting. This process is akin to most of the other languages, such as in English, *"come"* + *"-ing"* $\rightarrow$ *"coming"*, where we lose the additional "*e*" in the word *"come"* while merging. Another such example include words such as "indirect", "impossible", and "illuminate", where all these words have the same prefix as "in-", however, that got modified when merging with the root word. Learning this process could provide fundamental insights into the linguistic cognition of forming words and sentences of a language. Further, learning an automated algorithm that could perform Sandhi splitting and merging in Sanskrit would also provide a framework for learning word organization in languages.

Based on the occurrence of Sandhi, they are broadly classified into two types:

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

   For example, *bho* (changed form of verb *bhū* (to be)) + *anam* (a noun forming suffix) $\rightarrow$ *bhavanam* ("being") is a case of internal Sandhi.

2. **External Sandhi:** On the other hand, when Sandhi takes place between two words or a compound word, it is known as external sandhi.

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

Sandhi can be further classified depending on the type of the last letter and the first letter of the merging words, into the following three categories:

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

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

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

Sandhi splitting involves an additional task of localizing the position at which the splits have to be made in a given word or compound word. Thus, designing a Sandhi splitter is considered more challenging than a Sandhi merger. Hence, most of the existing Sandhi splitters predict a ranked list of splits rather than a single split. A Sandhi splitter or a Sandhi merger is typically evaluated in terms of accuracy when one of the members in the ranked list matches with the ground truth. However, the major challenged involved in evaluation is the lack of a golden dataset having manually annotated results of Sandhi splitting and Sandhi merging. The existing dataset, such as the University of Hyderabad (UoH) dataset[1], has large scale but erroneous ground truth results making any evaluation performed in the dataset as not accurate. Thus, following are the major research contributions of the research:

1. Create a large scale benchmark dataset with manually created ground truth made publicly available to initiate and motive research in this important problem

[1]`http://sanskrit.uohyd.ac.in/Corpus/`

2. Evaluate the performance of three different publicly available tools on Sandhi splitting and Sandhi merging using the created benchmark dataset

3. Analyze the obtained benchmark results and show that there is scope of improvement of existing tools. This emphasizes the requirement for further research in Sandhi splitting and Sandhi merging.

The rest of the paper is organized as follows, Section 2 explains the algorithmic details of the three existing Sandhi splitting and merging tools. Section 3 details the proposed benchmark dataset and the task of manual annotation. Section 4 discusses the experimental performance of Sandhi splitting using the proposed dataset while Section 5 provides the experimental performance of Sandhi merging. Finally, Section 6 concludes the research and provides a direction for future work.

2 Exisiting Sandhi Splitting and Merging tools:

There has been considerable amount of research in the field of sandhi splitting and merging. Automated tools have been constructed to perform Sandhi splitting and Sandhi merging. In this section, we identify and discuss the algorithmic outline of three most popular publicly available tools.

2.1 JNU Sandhi Splitter[2]

Sachin et al. (**?**) developed this tool at Jawaharlal Nehru University under the guidance of Prof. Girish Nath Jha (Professor, Computational Linguistics, Special Centre for Sanskrit Studies). This tool supports only vowel based Sandhi splitting while does not perform consonant and visarga based splitting. The overall architecture of the system is as follows:

1. **Preprocessing:** This 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 example to avoid processing.

2. **Subanta analysis:** This includes splitting the noun phrases into its constituent base and case terminations, where only the base termination are used for further processing.

3. **Fixed List checking:** Common nouns and proper nouns such as place names are exempted from Sandhi processing. Thus, the base terminated words are checked against the *Monier Williams Sanskrit Digital Dictionary* to remove nouns from further processing, with a remaining unified lexicon.

4. **Sandhi Analysis:** A list of Sandhi rule-base is checked in a database. The segmenter class will against the rule list to mark the potential splitting point and the Sandhi pattern corresponding to the marked morpheme.

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.

2.2 UoH Sandhi Splitter[3]

Kumar et al.(**?**) developed this tool at the Department of Sanskrit Studies, University of Hyderabad under the guidance of Prof. Kulkarni. A previous version of the Sandhi splitter is also available (**?**). The basic outline of the algorithm adopted in this Sandhi splitter is explained as follows:

1. Recursively break a word at every possible position applying a Sandhi rule to generate all possible morpheme candidates.

2. Pass all the constituent candidates through a morphological analyser.

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

[2]Available at `http://sanskrit.jnu.ac.in/sandhi/viccheda.jsp`

[3]Available at `http://sanskrit.uohyd.ac.in/scl/`

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.

2.3 INRIA Sanskrit Reader Companion[4]

Goyal et al. (?) developed this Sansksrit language segmenter and parser at INRIA, France under the guidance of Prof. Gerard Huet. The algorithm details of this tool are explained as follows:

1. Initially the lexicon is analyzed to gather stems and their morphological parameters, such as permitted genders of nominal stems, allowed classes and attested preverbs for roots.

2. In the next stage, more stem generation occurs for roots accounting for the various tenses, moods, absolutives and participles in 10 varieties.

3. Finally, inflexional morphology paradigms derive the infected forms according to the morphological parameters, some of which being read from the lexicon while the others being defined in specific tables.

3 Data Curation

The major research contribution of this paper is to create and benchmark the performance of Sandhi splitting and merging on a large scale database with clean manual annotations. Most of the automated algorithms in computational linguistics are data driven, and requires labeled data both for learning and evaluating models. Thus, the availability of manually annotated dataset forms the most essential driving force for research to be conducted in the domain. We create two types of datasets: (i) rule based corpus and (ii) literature based corpus. The rule based corpus are completely manually created using the *Pāṇini's* sandhi rules. Thus, the rule based corpus has limited data as the complete manual curation is arduous and costly. Literature based corpus contains the text extracted from Sanskrit literary works along with its manually annotated ground truth for Sandhi splitting and merging. The latter set contains words that are more used in practice and thus provides a more practical evaluation of tools. 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.

[4]available at `http://sanskrit.inria.fr/DICO/reader.fr.html`

3.1 Rule based corpus

3.1.1 Example corpus

We created a corpus containing at least one example for each of *Pāṇini's* Sandhi rules. This brings out how many rules are actually implemented by the splitters. This dataset contains 282 examples against the existing 271 rules. The corpus is available at `giturl`. This corpus contains examples for both internal Sandhi and external Sandhi, as shown in Table 1. There are about 150 examples for internal Sandhi and 132 examples for external Sandhi.

Word	Split	Rule	Type
viccheda	vi+cheda	6.1.73.1	internal
ācchādayati	ā+chādayati	6.1.74.1	internal
svacchandaḥ	sva+chandaḥ	6.1.73.1	external
mācchidat	mā+chidat	6.1.74.1	external

Table 1: Rule based corpus example

The compound word along with the split is provided as the ground truth data. We also provide the rule number corresponding to the *aṣṭādhyāyī.*

3.1.2 Benchmark corpus

We identified there is some mismatch between manual evaluation versus automated evaluated so we created a small set of corpus inorder to benchmark automated tools.

This corpus contains 150 examples from the actual literature. 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 this 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.

3.2 Literature based corpus

A huge set of corpus was created manual by UoH[5]. We manually verified their corpus and identified

[5]`http://sanskrit.uohyd.ac.in/Corpus/`

that the corpus is not reliable. We considered following three corpora from the literature:

1. Manually created *Bhagavad-gītā* Corpus
2. Dictionary-filtered UoH corpora
3. *Aṣṭādhyāyī* Corpus

3.2.1 *Bhagavad-gītā* Corpus

The Sandhi split *bhagavad-gītā* corpus at the UoH website had several limitations which made it unreliable to be used for the purpose of automated evaluation. For example, out of the total 431 Sandhi cases within the first two chapters, there were 41 typos, 92 cases of insufficient splits, and 10 cases of even wrong splits. Thus, we manually created a new corpus. This was done for the first nine chapters of *bhagavad-gītā*, having a total of 1432 words.

3.2.2 UoH Corpus

The UoH website has 39 sandhi-split corpora but they are not fully correct. There were around 113, 913 Sandhi splitting cases. There are cases of typing errors, insufficient splits, and incorrect splits. Therefore, they were not directly used for the purpose of automated evaluation. Thus, we filter this dataset to create the error less subset of the original UoH corpus.

Filtering is performed by comparing the splits against a set of thirteen dictionaries[6]. 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 Sandhi splits do not have typing errors, a Sandhi tool was used to check whether the result in each case matched with the word as given in the corpus. If the two words did not match, the original word and its split are neglected. Finally the filtered list contains 18674 split cases. Table 2 provides some example splitting available in the original UoH dataset. 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.

[6]Available at `url`

Word	Split
tumulo vyanunādayan	tumulaḥ + vi + anunādayan
sarvānbandhūnavasthitān	sarvān + bandhūn + avasthitān
śabda iva	śabdaḥ+ iva
nārhati	na + arhati
astamito bhagavān	astam + itaḥ + bhagavān

Table 2: Examples of filtered cases performed in the original UoH dataset. The first two cases were not included while the next three are considered in our filtered dataset.

3.2.3 *Aṣṭādhyāyī* Corpus

Many *sutrā* (rules) of *pāṇini* themselves contain Sandhied words. All the sutras with their splits are available at `http://sanskritdocuments.org/learning_tools/ashtadhyayi/`. 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. Hence, the approach used in the previous case to limit to cases of correct splits could not be applied in this case.

Since the fundamental challenge is the insufficiency of split, the splits which can undergo further splitting themselves are likely to be of greater length than fundamental morphemes. The larger the length of the splits, the more likely they are to undergo further splitting. Thus, 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. 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. Thus, a total of 3, 959 sutras is reduced to 2, 700 where splitting is applicable.

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

4 Evaluation Methodology:

4.1 Sandhi splitting tools

The evaluation was done on all the corporas. Except JNU, UoH and INRIA gives both filtered (top ranked) solutions and elaborated complete list of splits. But JNU provides only complete list of possible solutions. When we evaluated UoH for the performance we found UoH is trying to provide as minimum splits as possible which hampered its accuraccy and surprisingly almost none of the suggested splits are correct, Only elaborated list of split contains the right split. On the other hand INRIA tries to provide a list of filtered solutions as well so the difference in accuraccy between filtered and complete list of soultions is less than 10%. To avoid any kind of bias we are providing the accuraccies from the elaborated solutions list.

4.2 Rule-based corpus

The three splitters are evaluated against the rule based corpus

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 suffixes 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 3: 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 4: Evaluation Results

Cases not detected by any Splitter- 62 (46.9 %) for External Sandhi and 114 (74 %) for Internal Sandhi

4.2.1 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.

4.2.2 Benchmarking our own automated tool:

We identified the loss in case of automated versus manual ealuation of these sandhi splitting tools. The margin of difference between these two methods of evaluation can be taken as the scale for next the evaluations on the literature corpus.

Splitter	Manual Evaluation Results (150)	Automated Evaluation Results (150)
JNU	19(12.66%)	15(10%)
UoH	105(70%)	98(65.33%)
INRIA	127(84.66%)	94(62.66%)

Table 5: Evaluation Results

The results of automated evaluation are reported below. An average error rate of 0.18 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 evaluated. Only automated evaluation has been done for them.

4.3 Literature Based Evaluation

4.3.1 *Bhagavad-gītā* Corpus

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 Chapter wise, here A refers to *adhyāya* in Sanskrit:

From the results it can be observed that INRIA was giving in average around 70% of accuraccy for each of the chapters making the overall accuraccy to 70%. UoH performed betterfor 4th chapter and performed poorly for chapter 9. The average preformance for UoH is around 45%. JNU performed equally for all the chapters and the performance of JNU is just 4.3%.

Chapter	Words	JNU	UoH	INRIA
1	157	2 (1.3%)	76 (48.4%)	114 (72.6%)
2	270	10 (3.7%)	132 (48.9%)	188 (69.6%)
3	169	11 (6.5%)	83 (49.1%)	110 (65.1%)
4	165	9 (5.5%)	86 (52.1%)	108 (65.5%)
5	113	4 (3.5%)	52 (46.0%)	76 (67.3%)
6	187	13 (7.0%)	80 (42.8%)	122 (65.2%)
7	120	6 (5.0%)	43 (35.8%)	84 (70.0%)
8	116	6 (5.2%)	54 (46.6%)	79 (68.1%)
9	135	1 (0.7%)	44 (32.6%)	91 (67.4%)
Total	1432	62 (4.3%)	650 (45.4%)	972 (67.9%)

Table 6: Evaluation Results

4.3.2 UoH corpora

Results for UoH corpus is shown in the following table 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 7: Evaluation Results

For UoH corpus JNU performed better than *Bhagavad-gītā* Corpus. performance of JNU splitter for *Bhagavad-gītā* Corpus was around 4.3% where as for UoH corpus the performance is 17.5%. This must be result of the filtering process of data sets. As we checked for dictionary for splitted words which makes the words in right side small and simpler which boosted JNU's accuraccy. Similarly there is a significant difference for UoH result as well. There is an overall increase of 17%.

INRIA there is no significant difference between *Bhagavad-gītā* Corpus and UoH corpus the difference is around 5%.

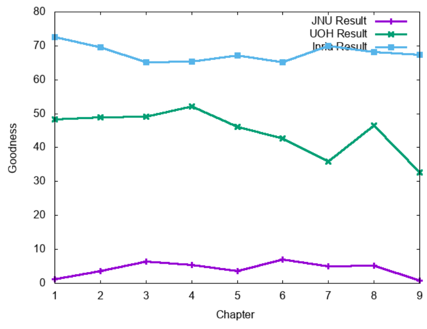

Figure 1: Screenshot of the built QA system showing the orchestration flow for the question - "problem with transfer order"

# Letters (≤)	Sandhis	JNU	UoH	INRIA
10	93	4 (4.3%)	21 (22.6%)	29 (31.2%)
20	571	1 (1.8%)	100 (17.5%)	195 (34.2%)
30	1512	17 (1.1%)	226 (14.9%)	378 (25.0%)
40	2045	18 (0.9%)	263 (12.9%)	444 (21.7%)
50	2302	18 (0.8%)	263 (11.4%)	460 (20.0%)
All	2700	18 (0.7%)	263 (9.7%)	507 (18.8%)

Table 8: Evaluation Results

4.3.3 Word-length filtered *āstaadhyaayi* corpus

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

From the results its evident that when the size of letters were restricted to 10 all the three splitters performed their best. At letters size 20 performance of INRIA increased by a small percentage but the performance of other splitters started dipping. From letters size 20 onwords for all the splitters the performance started decreased. From the results two things can be inferred the first one is there must be some issue with the dataset for example most of the words were not splitted completely etc. or the second inferrence can be the splitters themselves are not capable of handling complex and long size words.

4.4 Sandhitype based analysis

As JNU sandhi merger tool implemented only vowel sandhi rules we splitted all our corpora into the three types of sandhis and carried out the experiments, the results were shown in the following three graphs. As their claim JNU performed better for vowel sandhi cases in other two type of sandhi cases JNU performed poorly. UoH out performed in vowel sandhi cases where as INRIA performed better in the other two sandhi types.

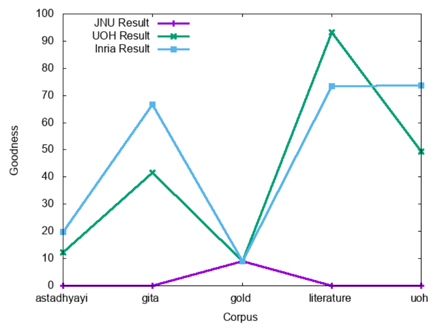

Figure 2: Visarga

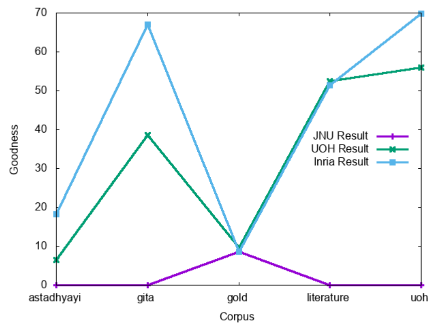

Figure 3: Consonant

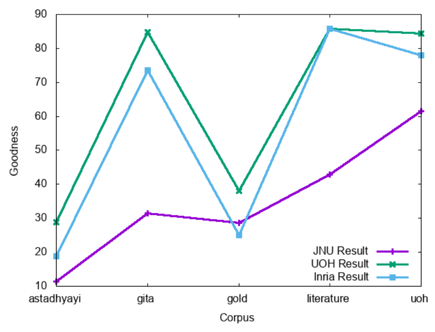

Figure 4: vowel

4.5 Evaluation of Sandhi Merging tools

Implementation of Sandhi merging tool is simpler when compared to implementing sandhi splitting and parsing tool. The advantage in sandhi merging is most of the rules consider only the last letter of the current word and the first letter of its adjacent word. So it is clear to use which rule. All the Sandhi Merging tools allow only two words to be merged. So in our automated tool we took first two pairs for merging and we took the resulting compound word for the next input along with its adjacent word.

INRIA sandhi merging tool has an option to choose between internal and external sandhi So for our analysis we tried both internal and external sandhis and merged the result, and gave positive merge if either of one tool merges it correctly. Other two tools has no distinguish between internal and external sandhis.

For sandhi merging we did only automated evaluation, and the results were presented in the following subsections.

4.5.1 Example corpus

We evaluated our example rule based corpus to see how many rules were implemented in each of the merging tool.

Merging tool	**Results (282)**
JNU	20.92%
UoH	32.97%
INRIA	51.77%

Table 9: Evaluation Results

From the table it is clear that most of the rules were implemented in INRIA which is 51%. This implies that half of the rules were not implemented by any of the tools. In UoH almost 33% of the rules were implemented where as in JNU only 21% of the total rules were implemented.

4.5.2 Benchmark dataset

We than ran on the benchmark dataset to see the sample output. Results were shown in the following table

Merger	**Results (150)**
JNU	75(50%)
UoH	129(86%)
INRIA	131(87.33%)

Table 10: Evaluation Results

Though very few rules were implemented by UoH the performance of UoH is comparable with that of INRIA where almost 51% of the rules were implemented. Both UoH and INRIA performed an overwhelming 86% and 87% respectively. Where as JNU performed poorly and the performance is only 50%.

4.5.3 *Bhagavad-gītā* Corpus

Merging results for the sandhied words of *Bhagavad-gītā* first 9 chapters were shown in the following table.

Chapter	**Words**	**JNU**	**UoH**	**INRIA**
1	157	47 (29.9%)	90 (57.3%)	87 (55.4%)
2	270	71 (26.3%)	181 (67.1%)	174 (64.4%)
3	169	52 (30.8%)	126 (74.6%)	111 (65.7%)
4	165	47 (28.5%)	121 (73.3%)	105 (63.6%)
5	113	29 (25.7%)	77 (68.1%)	64 (56.6%)
6	187	47 (25.1%)	114 (61.0%)	100 (53.5%)
7	120	29 (24.2%)	64 (53.3%)	53 (44.2%)
8	116	30 (25.9%)	67 (57.8%)	63 (54.3%)
9	135	27 (20.0%)	65 (48.2%)	55 (40.7%)
Total	1432	379 (26.5%)	905 (63.2%)	812 (56.7%)

Table 11: Evaluation Results

From the results it is clear that all the merging tools performed their best for the third chapter and performed badly for the ninth chapter. UoH merging tool performed slightly better than the INRIA tool. Performance of UoH and INRIA tools stands at 63% and 57% respectively where as JNU's performance is only 26.5% only.

4.5.4 UoH Corpus

We ran the filtered UoH corpus on all the three merging tools. and the results are shown in the following table.

For UoH corpus also UoH performed better than other two merging tools.

4.5.5 *Aṣṭādhyāyī* Corpus

Sandhi merging tools also performed better when the characters length is less than or equal to 20.

Merger	No. of Cases Correctly Identified
JNU	6517 (35.56 %)
UoH	12895 (70.36 %)
INRIA	12611 (68.81 %)

Table 12: Evaluation Results

# of letters (≤)	Sandhis	JNU	UoH	INRIA
10	93	26 (28.0%)	58 (62.4%)	51 (54.8%)
20	571	120 (21.0%)	343 (60.1%)	269 (47.1%)
30	1512	250 (16.5%)	763 (50.5%)	634 (41.9%)
40	2045	315 (15.4%)	988 (48.3%)	803 (39.3%)
50	2302	355 (15.4%)	1096 (47.6%)	890 (38.7%)
All	2700	411 (15.2%)	1260 (46.7%)	1016 (37.6%)

Table 13: Evaluation Results

All three merging tools performances started deccaying when we increase the word length.

4.5.6 Sandhi type based evaluation

We devised our experiment results in to three different types sandhis. JNU merger performed well in case of vowel sandhi examples only. for other two types of sandhis JNU merger did not performed well. All the merging tools performed in case of vowel sandhis but performed poorly inc ase of consonant sandhis. IN all the three types of sandhis UoH merging tool performed better than the other two merging tools.

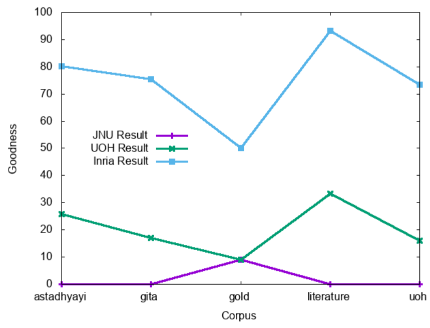

Figure 5: Visarga Merge results

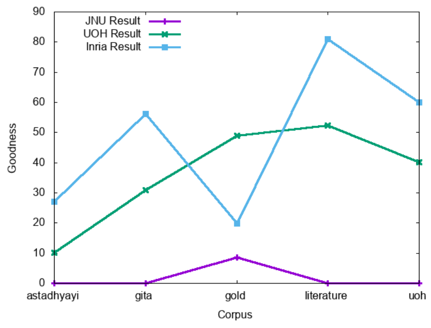

Figure 6: Consonant Merge results

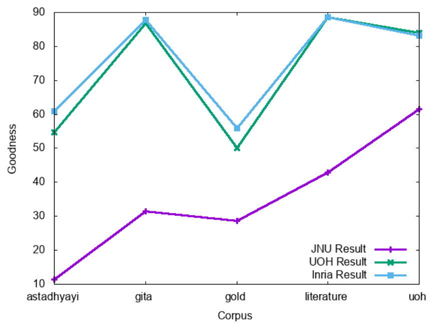

Figure 7: vowel merge

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:

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

6 Conclusion

In this paper we discussed different tools that implement Sanskrit sandhi merging and splitting mechanisms. We create three manually created data set to benchmarking both sandhi merging and splitting tools. Alwo we curated ambiguous existing corpus to get reasonably accurate sandhi example corpus. We also benchmarked the above three tools with our dataset. From our experiments it is evident that in case of Sandhi splitting INRIA performed better where as JNU performed very poorly. In case of sandhi merging UoH and INRIA performed almost equivalently where UoH performed slightly better than INRIA. JNU again performed poorly in case of merging also.

In JNU only very limited number of vowel sandhis were implemented. When we see sandhi merging the number of rules that were implemented in all the three tools is almost 50%. Which opens the door for implementation of other rules which were missing.