Validation and Normalization of DCS corpus and Development of the Sanskrit Heritage Engine's Segmenter

Sriram Krishnan[1], Amba Kulkarni[1], and Gérard Huet[2]
[1] Department of Sanskrit Studies, University of Hyderabad
[2] Inria Paris Center
sriramk8@gmail.com, apksh.uoh@nic.in, gerard.huet@inria.fr

Abstract

The Digital Corpus of Sanskrit records around 650,000 sentences along with their morphological and lexical tagging. But inconsistencies in morphological analysis, and in providing crucial information like the segmented word, urges the need for standardization and validation of this corpus. Automating the validation process requires efficient analyzers which also provide the missing information. The Sanskrit Heritage Engine's Reader produces all possible segmentations with morphological and lexical analyses. Aligning these systems would help us in recording the linguistic differences, which can be used to update these systems to produce standardized results and will also provide a Gold corpus tagged with complete morphological and lexical information along with the segmented words. Krishna et al. (2017) aligned 119,000 sentences, considering some of the linguistic differences. As both these systems have evolved significantly, the alignment is done again considering all the remaining linguistic differences between these systems. This paper describes the modified alignment process in detail and records the additional linguistic differences observed. It also proposes a modification to the existing Heritage segmenter where an additional ranking algorithm is introduced to rank solutions based on joint probabilities calculated from statistical data generated from the results of the alignment process. And finally, the datasets generated during this process are also released publicly.[1]

1 Introduction

Computational processing of Sanskrit is challenging due to sandhi, compounding and the free word order. Sandhi is mandatory between the components of a compound. While the sandhi between words in a sentence is left to the discretion of the speaker, due to the oral tradition, we find a greater tendency to use sandhi even in the written texts. The last decade has seen emergence of several computational tools for analysis of Sanskrit texts at various levels ranging from identification of sentence boundary (Hellwig, 2016), segmentation (Huet, 2005; Hellwig and Nehrdich, 2018), compound analysis (Gupta, 2012), morphological analysis (Kulkarni and Shukl, 2009; Huet, 2005) to sentential parsing (Kulkarni, 2019). The complexity of the sentential parser is reduced if it receives a morphologically analysed segmented text as an input. A collaborative effort between the developers of Sanskrit Heritage (SH) Platform and Saṃsādhanī team (Huet and Kulkarni, 2014) permitted to share efforts on these problematics. The Sanskrit Heritage Platform concentrated on the segmentation guided by the word forms validated through the lexicon. The Saṃsādhanī team focussed on the development of a parser (Kulkarni, 2019). The segmentation algorithm of SH Platform uses a novel approach to finite state technology, through Effective Eilenberg machines (Huet and Razet, 2015). The non-determinism involved in segmentation as well as during the morphological analysis results

[1]The Dataset can be accessed in the following github link: https://github.com/samsaadhanii/datasets under dcs_sh_alignment/

in multiple possible segmentations of the given input string. The main reason behind the non-determinism is the absence of semantic compatibility check during the process of segmentation. The segmenter produces billions of possible segmentations with all relevant linguistic details such as morphological analysis, and links to the dictionary entry. In order to display these billions of solutions, an efficient compact shared representation of these solutions using tabulated display interface was developed (Goyal and Huet, 2016). Krishnan and Kulkarni (2019) enlisted these solutions by ignoring the linguistic details which are irrelevant from the segmentation point of view, merging the solutions that have same word level segmentation and prioritizing the solutions with the help of statistical information from the SHMT corpus.[2]

Some requirements of Sanskrit computational tools are very specific. Sanskrit has a vast literature spreading over several knowledge domains. Most of the important Sanskrit literature is already translated into several languages.[3] In spite of this, scholars want to have access to the original sources. Thus development of the computational tools with convenient user interfaces that allow seamless connectivity to and from the lexical resources, generation engines and analysis tools becomes meaningful. Though the user would like to have an access to all possible interpretations, it is desirable to rank the solutions and display a few of them. Only when none of the displayed solutions is correct, all other solutions should be made available to the user. In order to rank the solutions, one needs some annotated corpus which can be used to learn the priorities.

For this, Heritage segmenter is required to be facilitated with statistical analysis. Recently the digitization of Sanskrit manuscripts shot up. But the amount of annotated data available for Sanskrit is very small compared to the size of the texts available in it from ancient times. An effort towards having such an annotated data was initiated and resulted into the Digital Corpus of Sanskrit (DCS) (Hellwig, 2010 2019).[4] This data, being of reasonable size, can be used for both statistical analyses and machine learning algorithms.

This paper focuses on how DCS's data can be used along with the Heritage Engine's analysis so that we get a proper morphologically tagged and segmented corpus. It starts with describing the annotation schemes of the two systems, their advantages and limitations and the need for alignment. A similar effort towards aligning the DCS annotated data with the analysis of Heritage Engine's data was already reported by Krishna et al. (2017). This work is briefly described in section 3, along with some issues related to the alignment. Looking at the limitations of the previous work, an effort towards building a better dataset was started. A proper alignment between the representations of these systems was done. The alignment process is described in section 4. But there were some additional difficulties due to the differences in the design decisions of the two systems. These difficulties were recorded systematically and are discussed in section 5. The observations are put down in section 6. Statistical information was extracted from the segmented and morphologically tagged corpus. With the help of these statistical information, the Heritage Segmenter was updated with an additional ranking algorithm which is described in section 7. Testing the modified Heritage Segmenter using Bhagavad Gita verses and comparing the performance of the different statistical information, the different metrics, and with the model proposed in Hellwig and Nehrdich (2018) are under the same section.

[2] A corpus developed by the Sanskrit-Hindi Machine Translation (SHMT) Consortium under the funding from DeItY, Govt of India (2008-12). `http://sanskrit.uohyd.ac.in/scl/GOLD_DATA/tagged_data.html`

[3] Most of the literature is in poetry. In spite of having proper translations for these, the true essence of such poetry can be appreciated only in the original!

[4] `http://www.sanskrit-linguistics.org/dcs/`

2 Resources

Before going into the discussion, let us take a look at the morphological annotation and the segmentation annotation of The Digital Corpus of Sanskrit, and The Sanskrit Heritage Engine.

2.1 The Digital Corpus of Sanskrit

The DCS is a sandhi split corpus of Sanskrit texts with full morphological and lexical analysis. There are about 650,000 sentences with more than 4,500,000 word references and 175,000 unique words. All this data was collected from around 400 Sanskrit texts. For every word reference, the following list of attributes is present in DCS:

1. lemma
2. morphological class
3. case, number and gender for nouns along with a combined value termed CNG[5]
4. tense, person, and number values for verbs
5. prefix (optional)
6. finite verb form (optional)
7. infinite verb form (optional)
8. position in sentence, position inside chunk
9. meaning of the lemma

Krishna et al. (2017) represented this data as objects containing the sentence details like chunks, lemmas, and CNG values. A glimpse of what an object looks like is depicted in table 1. This object was used for alignment with the analysis done by Heritage Engine.

Sentence Id	83
Sentence	mauktike yadi saṃdehaḥ kṛtrime sahaje'pi vā
Chunks	[‘mauktika’, ‘yadi’, ‘saṃdeha’, ‘kṛtrima’, ‘sahaja’, ‘api’, ‘vā’]
Lemmas	[[‘mauktika’], [‘yadi’], [‘saṃdeha’], [‘kṛtrima’], [‘sahaja’], [‘api’], [‘vā’]]
Morphological Class (CNG)	[[‘171’], [‘2’], [‘29’], [‘171’], [‘171’], [‘2’], [‘2’]]

Table 1: An example DCS Object data

DCS presents a lemma for each segment along with its morphological analysis. In some cases the derived stem is chosen as a lemma, and in some cases the underived one is chosen. The system is not uniform in deciding the lemmas. This might be a result of context-specific analysis, but in the absence of any tagging guidelines, we do not know the reason for such inconsistency. This corpus is curated single-handedly. Thus we can assume consistency in tagging. However, every human is prone to error. The quality of this data tested on a small sample (Hellwig and Nehrdich, 2018) revealed that around 5.5% of the compound splits are doubtful and around 2% errors are due to segmentation. The recently released data by Hellwig and Nehrdich (2018) contain the split points and sandhi rules proposed by the tagger, but these data are not easy to align with the DCS data.

2.2 The Sanskrit Heritage Engine

The Sanskrit Heritage Engine is a platform that hosts a lexicon (The Sanskrit Heritage Dictionary) and various tools like reader, lemmatizer, declension and conjugation engines. The Reader analyses any given text and segments it into all possible splits and displays them in a graphical interface where the user has the option to choose the required split. In addition to the graphical interface, it also enlists the solutions if the number of solutions is less than a threshold, say 100, also providing the sentential analysis. It closely follows Pāṇini's system i.e., all the rules governing the concept of *sandhi* that occur in *Aṣṭādhyāyī* are taken into consideration.

[5]A value denoting the case, number, and gender of the given word, for nouns. Or the tense, aspect, person, number, and gaṇa (present system class) for verbs.

This segmentation is lexicon directed, using forms systematically generated from its own lexicon.

Additionally, the morphological analyses (inflectional and if applicable derivational as well) are also provided for all the segments. This helps the user to disambiguate and correctly pick the intended split and prune the solutions that are not required. Such information also helps in the further stages of sentential analysis like parsing, disambiguation, and discourse analysis. Another advantage is that, this system combines a fast segmentation algorithm using finite-state transducers and dynamic programming with a first-pass of chunking that limits the inherently exponential complexity to small-length chunks, making the whole segmentation analysis fast enough in practice to be usable interactively.

One limitation of this system is that, it cannot arrive at a single solution mechanically. This owes to the fact that the current version does not take into account the meaning compatibility between various segments, which involves processing at sentential level. Another limitation is due to the Out of Vocabulary words. Though the dictionary contains high frequency words, as is the case with any NLP system, Heritage engine also suffers from automatic handling of Out of Vocabulary words. However the interactive interface allows the user to suggest the lemma for such words, which get stored in the local dictionary provided the lemmas are available in the Monier-Williams dictionary. And for some nominal words with certain prefixes like *su, vi, dur* and for words with *taddhita* suffixes, an explicit entry is required to be added in the dictionary.

This paper deals with the ways in which the DCS Gold data is aligned with one of the solutions of SH. This is an extension of the effort by Krishna et al. (2017) which is described in the next section. In addition to that, statistical information containing the frequencies for words, compound components, and sandhi rules are to be generated which are then used for:

1. updating the existing Segmenter in the Heritage Engine, and
2. testing the performance of the updated ranking algorithm and the frequencies

3 A Dataset for Sanskrit Word Segmentation

3.1 Description

Around 119,000 sentences from the originial DCS corpus were considered and an alignment between the DCS and the Heritage Engine was done by (Krishna et al., 2017). This lead to a huge dataset which had the input sequence, ground truth segmentation, and morphological and lexical information about all the phonetically possible segments. This was done primarily for the Word Segmentation task but is also useful for subsequent tasks. The main concern was to make this annotated corpus available for the use of statistical techniques and Machine learning algorithms.

Since there are differences in design decisions between DCS and Heritage Reader, the candidate segments provided by the Heritage Reader had to be adapted and a few additional segments were added so as to match the entries in DCS. The XML based GraphML format was used to represent the candidate space segments. The GraphML files consist of graph structures, $G(V, E)$ as the representation for the analyses of each of those sentences. The nodes, V, are the possible splits, and the values in the edges, E, denoting whether the participating nodes can co-exist in a solution or not i.e., whether or not they have an overlap in the position relative to the sentence, and that the overlapped portion does not follow any sandhi rule.

To match the two systems, the data from the Heritage Reader's analysis was scrapped and certain parameters such as word, lemma, position, morphological information, chunk number, word length, and pre-verbs were extracted. Corresponding to the morphological analysis the CNG was generated for the ease of alignment. With all these parameters as attributes of each of the nodes, graphs were built for each sentence. Standard graph processing libraries were used to extract the data from these graphs.

3.2 Issues already handled

The lemma provided by DCS and the CNG value are the attributes that help in mapping the two systems. Although direct mapping produced some results, there were multiple issues when the actual mapping was experimented with. Krishna et al. (2017) discusses these issues in detail and provides solutions for each of them. A short summary of the issues is presented below.

- One of the issues was with the compounds and Named entities. The DCS provided the lemma based on the context. If in a given context the compound has non-compositional meaning, then the compound was not split into its components. If the meaning in the given context is compositional, then the compound was split into components. The Heritage Reader's analysis is guided by the lexicon. If the lexicon contains a compound entry due to its non-compositional meaning or on account of being a Named entity, then the Reader in addition to producing all possible segmentations, also produced an entry without any segmentation. Deciding non-compositionality is a complex issue (Hellwig and Nehrdich, 2018).

- Another issue was with the derivative affixes, especially the secondary derivatives. These were treated separately in Heritage Reader, but DCS joins them.

- Third, there were inconsistencies while dealing with *anusvāras* which should have actually been the *anunāsikas* when followed by their respective class consonants.

- Fourth, in DCS, some of the non-final (iic) components of compounds have the same word-form as their stems. For example, *mahā*. But the Heritage Reader sticks to the Pāṇinian rules and produces the base stem (*mahat* in this case).

3.3 Issues yet to be handled

In addition to the above mentioned differences, we noticed following discrepancy at the level of analysis between the two systems.

- Use of homonymy index
  One important aspect of the Heritage Reader is that the dictionary has different entries for homonymous stems, and the morphological analyser provides the homonymy index of the stem. For example, the word *siddham* is analysed by Heritage Engine as shown below.

  ```
  [siddha_1 { pp. }[sidh_1]]{n. sg. acc.| n. sg. nom. | m. sg. acc}
  [siddha_2 { pp. }[sidh_2]]{n. sg. acc.| n. sg. nom. | m. sg. acc}
  ```

  If we look at the meanings of these two senses, we find that they are almost opposing each other.

  ```
  siddha_1  [pp. sidh_1] a. m. n. f. siddhā
  (French)  accompli, réalisé ; gagné, obtenu ; parfait
            qui a atteint son but, réalisé son objectif
  (English) accomplished, realized; won, obtained; perfect
            who achieved his goal, achieved his goal
  siddha_2  [pp. sidh_2] a. m. n. f. siddhā
  (French)  empêché, écarté, repoussé.
  (English) prevented, pushed aside, pushed back
  ```

  In DCS annotated data, homonymy of stems is available but mapping the sense of the stems found in the Heritage Reader with those in the DCS is not trivial. Hence Krishna et al. (2017) collapsed these two analyses into one ignoring the sense information. Although it is not used now, such distinctions would definitely be of greater use for sense disambiguation of such homonymous words.

- Level of analysis
  The engine provides both the inflectional as well as the derivational analysis for some *kṛdantas* (primary derivatives) i.e, participles, absolutives and infinitive forms. For example, the word *hitam* is an inflected form of the base root *hita* which can be derived in two different ways. This results in two different analyses for the word *hitam* as shown below.

  ```
  [hita_1 { pp. }[hi_2]]{n. sg. acc. | n. sg. nom. | m. sg. acc.}
  [hita_2 { pp. }[dhā_1]]{n. sg. acc. | n. sg. nom. | m. sg. acc.}
  ```

  The dictionary entries for these two stems show the meaning difference.

  ```
  hita_1    [pp. hi_2] a. m. n. f. hita
  (French)  envoyé, lancé, émis.
  (English) sent, launched, issued
  hita_2    [pp. dhā_1] a. m. n. f. hita
  (French)  placé, mis, disposé | convenable, avantageux ;
            utile, propre à, bon pour <dat. g. loc.> ;
            salutaire | amical, bienveillant ; qui fait le bien
            avantage, profit, intérêt ; bien, chose utile ; bien-être.
  (English) placed, put, disposed | suitable, advantageous;
            useful, suitable for, good for <dat. g. loc.>;
            beneficial | friendly, caring; who does good
            advantage, profit, interest; well, useful thing; well-being.
  ```

  The analyses of DCS data is dependent on the context or the meanings involved. So, DCS stores the analysis of the derived form sometimes and the analysis of the base form otherwise. As Heritage Reader proposes all possible analyses, it is thus challenging to map the Heritage Reader's analysis with the DCS' analysis due to the involvement of meaning. In the case of *hitam*, DCS chooses only the inflectional analysis according to the intended meaning in the sentence.

  In the case of causative forms of the verbs, DCS chooses the causative form of the verb *bhojay* as the stem for the word *bhojanīyāḥ*. But the Heritage Reader provides the following analysis for the same word.

  ```
  [bhojanīya {pfp. [2] }[bhuj_2]]{f.pl.acc. | f.pl.nom | m.pl.nom.} (1)
  [bhojanīya {ca. pfp. [2] }[bhuj_2]]{f.pl.acc. | f.pl.nom | m.pl.nom.} (2)
  [bhojanīya {pfp. [2] }[bhuj_1]]{f.pl.acc. | f.pl.nom | m.pl.nom.} (3)
  ```

  Of the three analyses provided by the engine, (2) has the causative form. Here one needs to construct the causative form *bhojay* from *bhuj+ca* in order to align the morphological analysis, which is not trivial and involves the rules from grammar. Additionally, the Heritage Engine analyses privative compounds like *anivṛttam* as *a-nivṛtta*, but is also updated regularly to lexicalize such compounds to have non-compositional meaning. Since DCS' analysis depends on the compositionality according to the context, the analysis might not map with Heritage Reader's analysis.

- Enhancement in the Heritage engine
  The Heritage Reader's Engine and the dictionaries have evolved in many ways in the past three years, and hence using the same GraphML files would neglect the improvements carried over during the last few years. One such change was with the way the Named Entities have been handled. In the earlier version, the compounds were always split into possible segments even if it represents a Named Entity. Another modification was regarding the pre-verbs. Earlier only the pre-verbs with derivational lemmas were joined, but in the current version the inflectional lemmas also have the pre-verbs attached alongwith.

In view of the above changes, and in order to resolve the problems with homonymy, we decided to align the Heritage Reader's analysis with the DCS afresh with modifications in the alignment process. This alignment of the manually tagged analyses of DCS with one of the analyses produced by the Heritage Engine would provide us with:

1. Identifying wrong annotations from DCS,
2. Consistent uniform analysis,
3. Probable compounds with non-compositional meaning,
4. Constituency analysis of compound words, and
5. Parallel corpus of segmented-unsegmented texts

4 Alignment Process

The DCS objects for the sentences are used in the same way they were used earlier. GraphML files are used for the representation of the mapped data. The same process of scrapping was used with a slight modification to scrap the derivational information, sense, and the correct representation of the morphological analysis.

The Mapping for every sentence was done in three stages:

1. Representing Heritage Reader's analysis as a graph,
2. Aligning the DCS annotation with Heritage Reader's analysis, and
3. Handling compounds with non-compositional meaning and words with derivational morphology

4.1 GraphML files creation

4.1.1 Scrapping

The first stage corresponds to scrapping data from the Heritage Reader's website and creation of GraphML files for each of the sentences from the DCS corpus. During this stage, CNG values corresponding to each morphological analysis are also obtained, and added as an additional attribute to help in connecting the Heritage Reader's analyses with the DCS entries (Krishna et al., 2017).

4.1.2 Graph Construction

The next part of this stage was creating graphs with nodes having the following form.

```
( id, { color_class, position, chunk_no, word, lemma, sense, cng, pre_verb,
morph, length_word, der_pre_verb, der_lemma, der_sense, der_morph, der_cng,
char_pos } )
```

All these values are extracted from the scrapped data. lemma, sense, pre-verb, morph and cng denote respectively the word's prātipadika/dhātu (stem/root), sense (based on different meanings), upasarga (pre-fix for verbs), morphological details, CNG - (case number and gender value) corresponding to the morphological information. der_lemma, der_sense, der_pre_verb, der_morph, der_cng correspond to the information pertaining to derivational morphology. color_class is an attribute for the interface and it denotes a particular color depending on the phase[6]. For example, iics have yellow, substantive/adjective forms have blue, indeclinable forms such as adverbs, conjunctions, prepositions have pink colors, etc.

Except for the derivational details, the sense information and the position based on character, all the others were created by Krishna et al. (2017). As in Krishna et al. (2017), the graph edge values are used to identify the co-existence of two nodes (segments) in a single solution. The edge value '1' indicates that the two nodes can be a part of a solution. The value '2' indicates that the two corresponding nodes cannot be in a single solution. For example, in the compound *pātālabhāsuram* (Figure 1), *pātāla* and *bhāsuram* are non-conflicting nodes and hence their edge is labeled '1'. But *bhā* and *bhāsuram* are separate nodes which are conflicting, and hence their

[6]Phases are the lexical categories like substantive, vocative, finite verbal forms, etc.

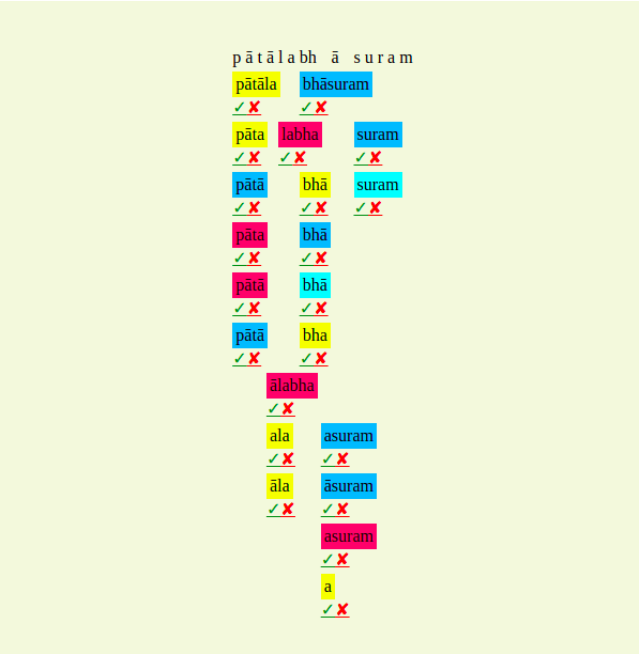

Figure 1: Heritage Reader's analysis of the compound *pātālabhāsuram*

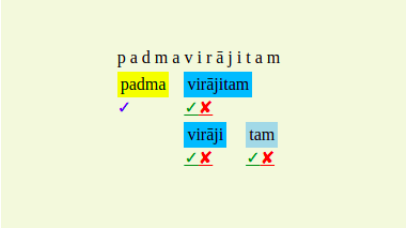

Figure 2: Heritage Reader's analysis of the compound *padmavirājitam*

edge is labeled '2'. Later, the value '3' will be stored for the edges in the correct segmentation solution.

As another example, let us consider the sentence *bindusthānaṃ madhyadeśe sadā padmavirājitam*. There are 2 possible ways in which the word *padmavirājitam* (Figure 2) can be analysed as depicted in table 2. So, the word *virājitam* is in conflict with the part *virāji* since they cannot co-occur together, and hence they will have an edge labeled as '2'. The nodes with *virāji* and *padma* will have an edge with label '1' since they can co-occur. Currently the edge information is not used but could be of use later.

Solution	Analyses
padma-virājitam	padma [padma]{iic.} virājitam [vi-rājita { pp. }[vi-rāj__1]]{n.sg.acc. | n.sg.nom. | m.sg.acc.}
padma-virāji-tam	padma [padma]{iic.} virāji [virāj__2]{m.sg.loc. | n.sg.loc. | f.sg.loc.} tam [tad]{m.sg.acc.}

Table 2: Analysis of the chunk *padmavirājitam*

4.1.3 Handling Homonymy

This involves merging the nodes with the same lemma and other parameters (like word, chunk, etc.) but different senses. Since it is not trivial to align DCS' sense analysis with Heritage

Engine's sense analysis, the individual nodes, having different senses, but the same lemma and same CNG value lead to multiple mappings with the DCS. Such nodes are collapsed into one, suppressing the information of senses. It was mentioned in section 3.3 that Krishna et al. (2017) collapsed all these nodes into one. As we have an additional attribute in 'sense', all the sense indices are temporarily stored in this attribute. New graphs were formed with the new nodes.

4.2 Comparison of DCS data and Heritage Reader's graph to analyze the parallels

The second stage of the alignment process deals with the actual analysis for creating the merged parallel database. First, both systems are normalized so that both of them have a uniform representation of the texts. Then the mapping process is initiated.

4.2.1 Normalization

Both the DCS data as well as the Heritage output are normalised to account for the variations in the use of *anunāsika* and doubling of consonants (*dvitva*). For example,

śrīśaṃkaraḥ to *śrīśaṅkaraḥ*
satvena to sattvena

4.2.2 DCS and Heritage Reader comparison

A comparison between the modified DCS data and modified Heritage Engine analyses was done in the following sequence in order to align them:

1. Mapping the base lemma, derived lemma (optional) and CNG.
2. Different conventions
   In the case of pronouns, both DCS and Heritage Engine follow different conventions while assigning the stem. For example, DCS analyses *tvam* as *tvad*, and the Heritage output following Pāṇini produces *yuṣmad* as the stem. These are treated as special cases with normal table lookup.
3. Mapping compound iics[7]
   In the case of iics (non-final components of compounds) of the DCS, the word-form and the stem for some segments are the same. But the Heritage Reader produces the original stem different from the word-form. For example, *mahādeva* is analysed as *mahā-deva* in DCS, but as *mahat-deva* in Heritage Reader. To handle this difference, the word-form (segment) is taken into consideration instead of the stem.

It was then observed that there were mappings with exactly one match, more than one match, at least one lemma not matched, and mappings with both multiple matches and unmatched lemmas.

So the results are categorized into 4 groups:

1. Single parallel mapping obtained for all lemmas in the sentence
2. Sentences that have at least one lemma with multiple parallels
3. Sentences that have at least one lemma without any parallel
4. Sentences that have at least one lemma with multiple parallels and at least one lemma without parallels

The unmapped sentences (3 and 4) are then sent for further modifications.

4.3 Modifications

Three kinds of modifications are done:

- Direct mapping of lemmas of verbs in the tenth gaṇa (class), having causative suffix *ṇic*.
  Eg, *pūjayati* is analysed in Heritage Engine as '*[pūj]pr. [10] ac. sg. 3*', but DCS has the lemma as *pūjay*. A separate list of such pairs like *pūjay-pūj*, *bhūśay-bhūś*, etc, was prepared. Additional nodes are created using such matching entries.

[7] *in initio compositi*

- The preverbs are sandhied with their corresponding lemmas, and derivational lemma is sandhied with its corresponding preverb labeled as der_pre_verb

  Eg, *praśaṃsanti* is analysed as '*[pra-śaṃs]pr. [1] ac. pl. 3*' in Heritage Reader, but DCS has the lemma *praśaṃs*. In this case, a new node with lemma *praśaṃs* is created by performing sandhi between the preverb *pra* and lemma *śaṃs*. Care should be taken when such sandhi is done, since there are certain required tranformations such as retroflexion of n (ṇ) and s (ṣ). For example, *pra* and *nam* become *praṇam*. Finally these new nodes are added to the graph.

- Merged possible components of compounds to form individual lemmas

  Eg, *śaṅkhaśuktyudbhavam* is analysed in Heritage Reader separately as *śaṅkha-śukti-udbhavam*, but *śaṅkha-śuktyudbhavam* is the expected solution according to DCS. So, for this chunk, all possible compounds are constructed and then each of it is compared to the DCS analysis. If the correct one is matched, a new node with the modified lemma, word and other information is created in the graph. The value for the attribute word, is kept as a hyphen-separated compound instead of the sandhied compound for future usage in constituency analysis. We should also make a note here that the total number of combinations is a Catalan number. So, generating all possible combinations for a given compound results in exponentially slow algorithm.

After these modifications, the second stage of analyzing the parallels is done for the modified graphs. Together with the previous results, the number of sentences with single parallels and multiple parallels is taken into consideration for observations.

5 Additional Problems

After the modifications, the mapping resulted in a reasonable amount of success. In the first phase, we were able to map 73,000 sentences and around 18,000 sentences had multiple possibilities. After the proposed modifications, 56,000 sentences were aligned and 66,000 sentences had multiple possibilities. Further results are available in Section 6. There were still issues regarding the lemmas that didn't match at all, and those which had multiple mappings. Let us first look into some of the issues that lead to the lemmas having multiple mappings.

Mapping for the CNG values is not one-to-one. For a given CNG value, there could be multiple morphological analyses. For example, the CNG value of -190 has morphological analyses as 'ca. pp.', 'des. pp.', and 'pp.'. We find multiple analyses being mapped for the same lemma. For the sentence *vasur ādyaṃ śivaṃ cādyaṃ māyābinduvibhūṣitam*, the lemma that had multiple parallels was *vibhūṣay*. On observing the morphological information, the difference between the multiple solutions was in the morphological analyses of the Heritage Engine. There were two entries with the same lemma - *vibhūṣay*. One had the morph as 'pp.' and the other had it as 'ca. pp.' (causative pp.). The DCS clubs both of them and assigns the CNG value as -190.

The DCS sticks with a single CNG value for primary and secondary derivatives but The Heritage Engine produces two levels of morphological analysis where the first level has the base stem and its analysis and the second level has the derived stem and its analysis. Considering the sentence *śrutaṃ vede purāṇe ca tava vaktre sureśvara*, the Heritage Engine's analysis for the word *śrutaṃ* is:

{[śruta { pp. }[śru]]{n. sg. acc. / n. sg. nom. / m. sg. acc.}}

The first level analysis is (*śru*, { pp. }), and the second level analysis is (*śruta*, n. sg. acc. | n. sg. nom. | m. sg. acc.). The CNG value for { pp. } is -190, and the CNG values for "n. sg. acc.", "n. sg. nom." and "m. sg. acc." are 31, 71, and 69, respectively. Although DCS refers to both the primary and secondary analyses, the CNG value assigned in DCS is -190. This lead to multiple mappings of DCS' analysis with the Heritage Reader's analysis.

The assignment of DCS' lemmas is also not uniform. Sometimes, even when the derivational lemma is available, it uses the inflectional lemma if the inflectional lemma is apt according to the context. For example, in the sentence *nīlaṃ nīlaṃ samākhyātaṃ marakataṃ haritaṃ hitam*, the analysis for *hitam* is provided as the inflectional form *hita*, and not it's derivational form *hi* or *dhā*. So, it is hard to map them because there is some amount of information missing.

Let us now look at an issue with compounds. For the sentence *ata eva hi tatrādau śāntiṃ kuryād dvijottamaḥ*, the compound *dvijottamaḥ* has its lemma as *dvijottama* in the DCS. But the compound modifications done to Heritage Reader's solutions leads to multiple entries with the same lemma. There are four possibilities: *dvija-uttamaḥ*, *dvi-ja-uttamaḥ*, *dvijā uttamaḥ*, and *dvi-jā uttamaḥ*. Of these, the last two are analysed as two words, and not a compound, and hence neglected but the first and the second are both possible compounds. In many such compounds, the differences are due to the combinations of the components.

In the Heritage Engine's analysis for the sentence *vada me parameśāna homakuṇḍaṃ tu kīdṛśam*, the word *parameśāna* is analysed as *parama-īśāna*. There are three analyses for the word *īśāna*. Two have participial forms generated from the root *īś*, and the other is generated from the lexicon entry *īśāna [agt. īś_1]* which states that it is an agent noun of the root *īś*. The participial forms and the agent noun belong to different phases (lexical categories) in the analysis - Krid and Noun, respectively. This distinction is present in the Heritage Reader because pre-verbs are not attached to general nominal entries but attached to participles. So, the nominal entry needs to be treated separately, and hence Noun is more preferred than being analysed as derived from the root *īś*. Since DCS has a single analysis which depends on context, and the Heritage Reader produces the possible distinctions, we arrive at multiple alignment. But, in this case, the agent noun, being derived from *īś*, is indeed a *kṛdanta* and should be present alongwith the other two.

In the sentence *kāraṇena mahāmokṣaṃ nirmālyena śivasya ca*, the word *mahāmokṣaṃ*'s analysis, according to DCS, has its lemma as *mahāmokṣa* with the CNG as 31. Such compounds' contexts need to be checked whether they are to be treated as compounds with non-compositional meaning, or whether they are to be split further.

Now, looking at the unmatched lemmas, we encounter the following difficulties. Some words are not analysed by the Engine at all either due to the absence of its *prātipadika* (stem) or *dhātu* (root) in the dictionary or, the engine fails to analyse the words. For example, the word *prameyatvam* is analysed by DCS as having the lemma *prameya* and *tvam*, but the Heritage Engine produces only parasite segmentations, in the absence of a lexical entry for *prameyatva*. Similarly certain *taddhitāntas* (secondary derivatives) like *nirguṇatvam* are not analysed in the Heritage Engine. Further modifications to the engine to analyse the secondary derivatives would help in aligning these words.

The secondary derivatives are treated like compounds in DCS. For example, in the sentence *prakuryāt tu dvijenaiva tadā brahmamayī surā*, the word *brahmamayī* is analysed as a compound of *brahman* and *maya*. The Heritage engine analyses it with the lemma as *brahmamaya*. Since DCS does not differentiate between compounds and words with secondary derivative suffixes, it is not possible to align such words with the Heritage Engine's analysis.

There is another issue with the way indeclinables like *api* are handled. The Heritage Reader analyses it as 'conj.' and 'prep.'. But DCS marks it as 'ind.' and assigns the CNG as 2. A normalization is required to classify properly such indeclinables under 'conj.', 'prep.' etc. The Heritage Reader has various classes for indeclinables like 'adv.'(adverb), 'abs.'(absolutive), 'part.'(particle), 'prep.'(preposition), 'conj.'(conjunction), etc. And *api* is stored both as 'prep.' and 'conj.' in the Heritage Reader. Either these are to be clubbed together or a mapping needs to be made from these two with the 'ind.' of DCS.

Sometimes the distinctions are present in the way certain *pūrva-padas* (non-final components) of compounds are analysed. For example, the component *rūpya* has the analysis as *rūpya* with the morphological analysis as 'iic'. But the Heritage Reader provides the lemma as *rūpya* with

morphological analysis as 'pfp. iic.'.

These are just a handful of examples of the issues encountered. Further analysis of the missed alignments will bring out more such difficulties, but will also provide an opportunity to modify the corpus and the two systems.

6 Observations

Observations were recorded in three phases. In the first phase, the existing dataset from Krishna et al. (2017) (107,000 aligned sentences) was taken into consideration and the parallel corpus for unsegmented and segmented sentences was extracted. In the second phase, the alignment process was run by considering only the sentences from Krishna et al. (2017), and in the third phase, all the sentences from DCS which have proper analysis were considered for the alignment process. Similar to the first phase, the parallel corpora of unsegmented and segmented sentences were extracted.

Type	Phase II	Phase III
Overall Sentences	119,004	621,445
Aligned	65,699	130,439
Aligned (with multiple analyses)	36,755	84,469
Not aligned (with missed analyses)	4148	103,808
Not aligned (with both multiple analyses and missed lemma)	4265	110,760
Not aligned (modifications could not be done)	6925	165,456
Not aligned (due to other reasons)	1212	26,513

Table 3: DCS-SH Alignment Observations

The observations are noted in table 3. The problems discussed in section 5 were encountered in phase II, where the sentences from the existing dataset (Krishna et al., 2017) were passed to the alignment process, and 16,550 sentences have to be checked for issues either in the Heritage Segmenter or in the analyses present in DCS. In Phase III, where all the sentences from DCS had been considered for alignment, 406,537 sentences were not aligned and have to be analysed individually for such issues. If the issue is found to be in Heritage Engine's analyses, then those are solved systematically. In this way, the DCS acts as a tool to improve the Heritage Engine. Currently, only the aligned sentences are considered to form the parallel corpus of unsegmented-segmented sentences. From all the three phases of alignment, these parallel corpora are built and are used to extract the frequency lists which are described in the next section.

7 Modification to Heritage Segmenter

7.1 Statistics Generation

As seen previously, having obtained the aligned sentences, three different parallel corpora of unsegmented and segmented sentences were extracted as follows:

1. The parallel corpus from the alignment done by Krishna et al. (2017) (Phase I)
2. The parallel corpus obtained from the modified alignment done on the DCS sentences used by Krishna et al. (2017) (Phase II)
3. The parallel corpus obtained from the modified alignment done on all the DCS sentences (Phase III)

The following frequencies were obtained from these parallel corpora:

1. word
2. compound component
3. sandhi between words
4. sandhi between compound components

Additionally, frequencies from the SHMT corpus was used. The total list of frequencies is as follows:

1. Frequency list from SHMT corpus
2. Frequency list from DCS-SH aligned parallel corpus (130,000 aligned sentences - Phase III)
3. Frequency list obtained from modified alignment on sentences from Krishna et al. (2017) (70,000 aligned sentences - Phase II)
4. Frequency list obtained from Krishna et al. (2017) (107,000 aligned sentences - Phase I)

7.2 Updated Dovetailing Algorithm

The Sanskrit Heritage Engine's Reader enlists all solutions based on a dovetailing ranking algorithm. A threshold of 100 was used to enlist the best solutions based on the ranking algorithm. But it produces all possible solutions taking into consideration the phase (lexical category) and transition details of each of the segmented forms. Krishnan and Kulkarni (2019) proposed a modification to the Reader where the information about phase and lexical analyses is removed, and the segmented word forms alone are considered for the solutions. Also, an algorithm was introduced which ranks the solutions based on a joint probability calculated from word and transition frequencies of the SHMT corpus. The first metrics used to calculate the joint probability value was:

$$C_{total} = \prod_{i=1}^{n} P_{w_i} \times P_{t_i} \tag{1}$$

which is *word_probability* $\times$ *transition_probability* of every segment. The unigram probabilities of the segments (words) and the bigram probabilities of the transitions (sandhi) are taken into consideration and their cumulative product across all the segments is the the overall product presented above.

The second metrics considered the *word_probability* alone where the resultant value is the product of unigram probabilities of the segments/words:

$$C_{total} = \prod_{i=1}^{n} P_{w_i} \tag{2}$$

These metrics of joint probabilities of the segments and transitions are similar to First Order Markov Model except that the segments are not states and hence the probability of the segment does not depend on the previous segment. And the frequency list had unigram frequencies for the words and bigram frequencies for the transitions. The ranking of the solutions is based on the joint probability of the solution and also the number of segments in the solution. The solutions with higher joint probabilities combined with the least number of segments are ranked higher. These probability values were calculated from the statistical data mentioned in section 7.1. The frequency list that is chosen for the updated segmenter is mentioned in section 7.3. The performance comparison of these two metrics are shown in section 7.4.1.

Having obtained all the solutions, the sentences were ranked using the first metrics. Around 21,000 sentences (from SHMT corpus) with at most three split locations in each of them, were chosen for testing. Although it resulted into 98% recall, the algorithm increased the response time of the segmenter tremendously for long sentences and sentences with huge compounds. So, the algorithm was shifted to the graphical Segmenter that generates a graphical interface where the user can choose or reject specific segments. An example is shown in Figure 3. The sentence is initially divided into chunks based on sandhi rules. Each chunk is analysed further where segmentation and morphology recognition are done parallely and only if the segment obtained after the split is present in the lexicon of the Heritage Engine and is morphologically correct, it is registered into the graphical interface. This process developed by (Goyal and Huet, 2016) is now modified by adding a dovetailing algorithm to rank the solutions and maintain

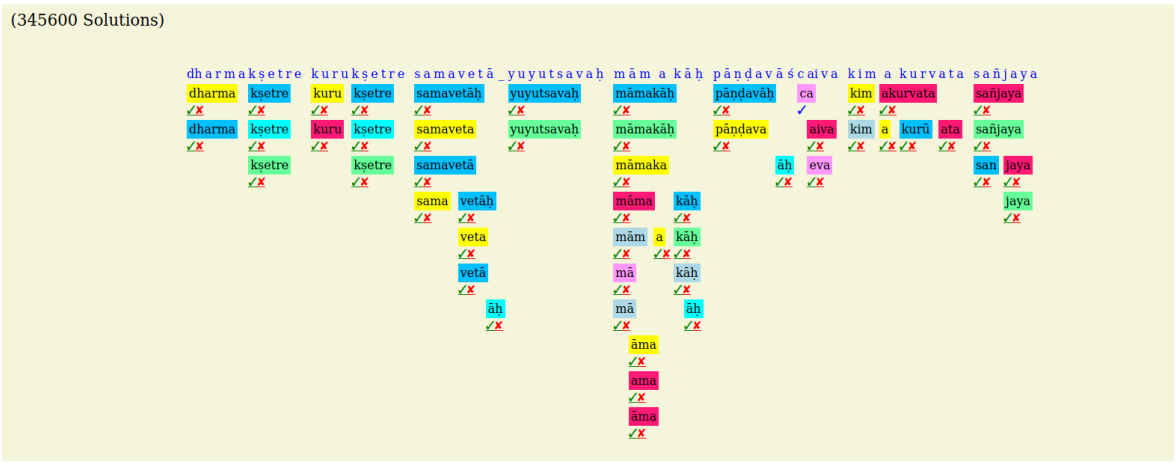

Figure 3: Heritage Segmenter Analysis for Bhagavad Gita 1.1

a bucket of best n solutions[8] based on the probabilities calculated from the frequencies. This value is calculated for every segment registered in the graph. For every chunk, a number of segments will be registered resulting into a number of segmentations each forming from non-overlapping segments in sequence. For each of the segmentations of the chunk, probabilities are also calculated cumulatively. After every chunk is segmented to form various segmentations, these are saved into a global list of chunk segmentations. And once the generation of the graphical interface is done, this list of chunk segmentations is run over the new dovetailing algorithm which traverses through this list and forms the entire list of solutions. And it keeps a bucket of the best n solutions based on the total joint probability of each solution. Finally, only these n solutions are displayed along with the graphical interface.

7.3 Choosing the correct frequency list

To compare the performance of the frequencies generated as shown in section 7.1, Bhagavad Gita Chapter Three's verses were run on the updated Segmenter with these four frequency lists. Of the four frequencies, SHMT performed the best by producing the correct solution (in the top 50 ranks) for 33 out of 43 verses. With the other frequencies, 29 out of 43 verses were analysed. Table 4 shows the distribution of solutions based on position for these four frequency lists.

Solution Rank	SHMT	Phase III (DCS-SH)	Phase II	Phase I
1	10	11	9	5
<= 2	15	13	11	5
<= 3	17	14	12	8
<= 4	19	15	12	9
<= 5	21	16	13	9
<= 10	23	20	17	13
	53.48%	46.5%	30.7%	39.53%

Table 4: Cumulative Position Distribution - Bhagavad Gita

Since there could be overlap of Bhagavad Gita in the lexical resources of SHMT, *Meghadhūta*[9] was chosen for the performance comparisons. 60 *pādas* from 15 verses were tested with all the four frequency sets and the results are shown in table 5.

It is clear from this observation that DCS-SH performs the best where the correct solution was found to be present in 55% of the sentences tested. Comparing both *Bhagavad Gīta's* and

[8] n is temporarily fixed as 100

[9] *Meghadūta* is a poetry text written by *Kālidāsa*.

Solution Rank	SHMT	Phase III (DCS-SH)	Phase II	Phase I
1	21	21	21	21
<= 2	27	28	27	27
<= 3	29	30	30	30
<= 4	30	31	30	30
<= 5	30	32	31	31
<= 10	31	33	32	32
	51.66%	55.0%	53.33%	53.33%

Table 5: Cumulative Position Distribution - Meghadūta

Meghadūta's observation would bring us to a conclusion that both the SHMT and DCS-SH have similar observations, but the scope of development of DCS-SH is high since more sentences might be added with annotations in the corpus. Around 400,000 sentences are to be checked for issues for not being aligned properly. If the issues in the missed sentences are solved, then the number of aligned sentences would increase. So, the statistical data obtained from DCS-SH aligned corpus was considered for calculating the probabilities of the segments.

7.4 Experiments with Bhagavad Gita corpus

7.4.1 On Heritage Segmenter

The unsegmented-segmented verses from Bhagavad Gita were used as development corpus on the updated Segmenter. For 278 verses, additional segmented solutions were added because of the differences in the way Heritage Segmenter deals with compounds. For example, named entities could be analysed as both standalone words as well as compounds with compositional meaning. For some of the compounds, the compositional analyses weren't present and had to be added. For the remaining verses, only one segmented solution was kept. The analysed sentences were put under three categories:

1. found the correct analysis and it is in the top 100 solutions (FOUND)
2. found the correct analysis and it is not in the top 100 solutions (MISSED)
3. could not find the correct analysis (WRONG)

Testing was done in two stages. The first stage was used for development of the Segmenter. Using the feedback from the first stage of testing, the segmenter was updated and the second stage was conducted. For the first stage only the first metrics described in section 7.2 was used. For the second stage, the performance of the ranking was checked with both the metrics separately and a comparison of the two stages was noted as given in table 6.

After the first stage of testing, the verses under MISSED and WRONG categories were analysed manually for possible issues in the Heritage Segmenter. Those under the WRONG category had the following reasons, predominantly:

- unrecognized words
- issue in sandhi
- improper compounding

Those under the MISSED category had solutions beyond the 100-mark. The joint probability had to be checked individually for each of the segments. Also, the issue was with the compositional analyses of compounds by the Heritage Engine. For some, the parallel corpus had to be modified. For the remaining verses, the issues were reported for updating the Segmenter.

The precision and recall values were calculated per sentence where the criteria for precision is to get the ratio of the number of sentences for which the correct solution was found to the total number of sentences. And recall was calculated as the ratio of the number of sentences for which solutions were obtained without leaving any word/chunk unrecognized to the total number of sentences. The precision increased from 74.28% to 98.28% (and 98.85%) from stage 1 to stage 2 while the recall remained the same. Also, there were 4 verses which couldn't be analysed in top

Category	Stage 1	Stage 2 (Metrics 1 - word * transition)	Stage 2 (Metrics 2 - word)
FOUND	520	688	692
MISSED	84	4	0
WRONG	94	6	6
TIMEOUT	2	2	2

Table 6: Bhagavad Gita on the updated Segmenter

100 solutions when the first metrics was used. The position difference between the two metrics is given in table 7.

Position	Metrics 1 - word * transition	Metrics 2 - word
1	308	364
2	89	118
3	51	48
4	32	40
5	39 (74.14%)	28 (85.42%)
6	20	18
7	12	9
8	11	9
9	13	5
10	10 (83.57%)	7 (92.28%)
11-20	58	28
21-30	16	8
31-40	12	4
41-50	8	5
51-100	9	1
> 100	4	0

Table 7: Bhagavad Gita on the updated Segmenter

With the first metrics, 74.14% of the verses had the correct solution in the top 5 ranks. With the second metrics, 85.42% of the verses had the correct solution in top 5 ranks. With both metrics, 6 verses could not be analysed at all. These are the reasons for such cases:

- periphrastic perfect forms: like *darśayāmāsa*, *āśvāsayāmāsa*, etc., are recognized as two segments instead of one. These aren't a combination of preverbs and verbs but are verbal compounds where the components are not strongly bound. Hence, it is difficult to glue them together to make them form a single segment.
- Since Bhagavad Gita is a part of the epic *Mahābhārata*, it is bound to contain certain words which were in use prior to Pāṇini's Aṣṭādhyāyī. Some of these are referred to as *ārṣaprayoga* (usage from *ṛṣis*. One such example is *priyāyārhasi*). It should actually be split into *priyāyāḥ* and *arhasi* according to the meaning, but the sandhi between these two words does not produce these splits.

7.4.2 Comparison with the model from Hellwig and Nehrdich (2018)

For comparing with similar models, the model from Hellwig and Nehrdich (2018) (rcNN) was chosen because it was also trained with DCS sentences. And the model used character level recurrent and convolutional neural networks for segmentation. Since DCS sentences were used for the alignment to generate the frequencies, a comparison was done between these two keeping in mind the following differences:

- The rcNN produces only one solution, whereas the updated Heritage segmenter produces the top n solutions, and

- rcNN is trained with DCS which has non-compositional analysis for the compounds, according to the context, while the Heritage Segmenter's compounds have compositional analysis predominantly.

So, the comparison was restricted to word level segmentations where the compound components were considered similar to the other words in the sentence. Thus, the parallel corpus generated earlier was modified to exhibit only the segmentations without differentiating the word splits with the compound component splits. And only the first solution was taken into account for comparison. 254/700 verses were analysed correctly by rcNN. In the case of the Heritage Segmenter, it was 308 with the first metrics and 364 with the second metrics.

Additionally, 1000 sentences from the test set of Hellwig and Nehrdich (2018) were tested on both the models (rcNN and the updated Segmenter). The Heritage Segmenter was unable to recognize at least one word in 385 of those sentences. So, the remaining 615 sentences were considered for a fair comparison. The rcNN was able to identify the ground truth segmentation for 371/615 sentences. The updated Heritage Segmenter was able to identify 295/615 sentences correctly. On observation, it was noted that the difference in the performance was predominantly due to the compositionality of the compounds in the Heritage Segmenter. While the ground truth segmentation in the test set has the non-compositional analysis, the Heritage Segmenter produces the components separately. A few such examples of the compounds are presented in table 8:

No.	Ground Truth Segment	Segment from Heritage Segmenter
1	kṣīramadhurā	kṣīra madhurā
2	tiktabījā	tikta bījā
3	mahārājaphalaḥ	mahārāja phalaḥ
4	pañcavidham	pañca vidham
5	ātmaka tvam	ātmakatvam
6	pañcarājiphalaḥ	pañca rāji phalaḥ
7	avakīrṇa vat	avakīrṇavat

Table 8: Comparison of Compositionality

As compositionality depends on context and cannot be finalised at the level of segmentation, and the Heritage Engine's lexicon has a limitation on many of the non-compositional lexicon entries of the compounds, the Segmenter specifies the compound components in the analysis. Hence, in the ground truth segmentations of the sentences that weren't segmented correctly, the compounds were split into their components for having compositional analysis in addition to the non-compositional analysis.

This increased the performance in both the systems. 401/615 sentences were analysed correctly by rcNN and the Heritage Segmenter was able to analyse 498/615 sentences of which 421 were analysed in the first position. The observations are presented in table 9.

615 test sentences	with ground truth compounds		with compositional analyses of compounds	
	rcNN	Updated Segmenter	rcNN	Updated Segmenter
FOUND	371	295	401	421
WRONG	244	320	214	117

Table 9: Comparison of rcNN vs Updated Segmenter

From the observations and the comparison of the compositionality, it is observed that the necessity of non-compositional analyses of certain compounds is questionable in the ground truth. In the same way, certain compounds which could have both compositional as well as non-compositional analyses (like named entities), are to be lexicalised in the Heritage engine's lexicon.

Thus, the differences in the way compounds are analysed in DCS and Heritage Segmenter is one of the main reasons for this performance difference. And that is the major obstacle to solve while trying to align these two systems to produce a proper dataset for segmentation tasks in Sanskrit.

8 Inferences and Conclusion

The alignment process and the experiments with the dataset obtained thereafter provided us three important cues to proceed forward towards the development of Heritage Segmenter. Several anomalies of the Segmenter were discovered and corrected with the help of the results from the experiments.

8.1 Improvement to the Heritage Engine

First, the lexicon was updated by the addition of words that went unrecognized by the engine. The Bhagavad Gita verses were used here as a development corpus to provide feedback for the Segmenter. As discussed in section 2.2, such entries were added to the lexicon to handle out of vocabulary words. Mostly, *taddhitas* with suffixes *tva, tā, vat*, compounds with prefixes *sa, su, vi, dus, etc.* and those agent nouns and action nouns which were encountered in the verses and which are not generated systematically by the engine were added.

Second, certain combinations of preverbs with root forms and kṛdantas were not permitted, which prevented the recognition of corresponding forms. The segmenter has a table, fixing for any root and gaṇa what voices are allowed for a given sequence of preverbs. Adding entries to the table allowed the recognition of the missing padas.

Third, to handle the words with suffixes *cit, cana* (refer section 7.4.1), the corresponding pronomial forms were glued with the suffixes and made to be generated by the engine.

Fourth, lexicalised compounds were treated as a whole even if they could be generated as a combination of component pieces (for the named entities whose meanings are non-compositional).

Having added these words, we were able to obtain better results (24% increase in precision). As discussed in 7.4.1, the engine couldn't produce solutions for 94 verses in stage 1 as each of the verse had at least one unrecognized word. In the second and third stages, solutions were found for 88 of these.

Also, the engine couldn't find the correct solution in the best 50 solutions for 84 verses because of the compounds which had their components split by the engine (not recognized as named entities). These were recognized after adding the lexical entries for these compounds.

Fifth, the dovetailing algorithm was also modified to make it consistent with the graphical interface. This was faster, produced more accurate results than the algorithm that was built for the Reader. Since the graphical interface does not narrow down to a single or a limited number of possible solutions, it requires additional effort from the user for providing the correct segmentation, although it is efficient and user-friendly. The updated ranking algorithm helps in truncating the entire list of possible solutions to a limited set (say, 5, 10, or 50), without any significant loss of recall. It is observed in table 7 that 85% of the verses have their solutions in top 5 ranks, and more than 90% of the solutions are found in the top 10 ranks owing to the improvement in the lexicon and the modified ranking algorithm.

Sixth, the frequencies were provided as a list in Krishnan and Kulkarni (2019) where accessing the frequencies delayed the process. To make it consistent with the data structures used in the engine and also to speed up the access rate, the frequencies were integrated into the newly formed decorated trie structure which holds both the words and frequencies by sharing the suffixes, similar to the trie structures built for the lexicon. Instead of populating the frequencies in the generated trie structure that holds the existing lexicon, a separate trie structure was built for these frequencies, making it a separate module.

Seventh, when only the word probability was used, correct solutions were produced in the top 10 ranks for 92% of the verses. The other metrics, where both the word probability and the sandhi probability of each of the segments was used, produces correct solutions in top 10 solutions for 83%. The metrics is similar to first order Markov model except that the current probability does not have any dependence on the previous state's probability because the segments are not considered as states as these segments are taken all at once. And the probabilities are calculated for each of the segments and cumulatively their product is obtained.

Finally, the alignment process provides for every word in the sentence, its base lemma, derived lemma, CNG value, preverb and morphological analysis. Currently, only the word and its frequency are being used by the algorithm. Experiments with other parameters should be tried for better results. Also, these could be used in further stages of sentential analyses.

8.2 Integrating the updates with *Saṃsādhanī* tools

The collaborative effort between the Sanskrit Heritage Engine and *Saṃsādhanī* team (Huet and Kulkarni, 2014) enables cross-platform analyses of sentences. Currently, it is left to the user to choose the correct or intended solution in the Heritage segmenter, and then send it to the *Anusāraka* in the *Saṃsādhanī* for parsing the sentence. With the help of the ranking incorporated in the engine, it is now possible to narrow down to the best n solutions, and choose from those. Again, looking at the performance comparison of the Bhagavad Gita corpus, 85% of the verses had their solutions in the top 5 ranks making it easy for the users to redirect to the *Saṃsādhanī*. So, the Heritage Segmenter acts as the front end to segment the sentence, and provide the best solution(s) where the user chooses one solution (instead of individual segments as in the original segmenter's interface), to parse with the help of the *Anusāraka* in the back-end.

With the development of the statistical data, the value of n should decrease gradually. As around 400,000 DCS sentences have to be dealt with further for analysing the issues in alignment, the number of aligned sentences is bound to increase. The frequencies of words and transitions also increases along with new words being added to the lists.

8.3 Improvement to the reference corpora

As discussed in section 2.1, there are a few limitations of DCS like the inconsistencies in the analysis, errors due to segmentation and compound splits. The alignment process helped in managing these issues and avoiding these in the resultant dataset. These are discussed here along with the state of the other corpora used - SHMT and Bhagavad Gita.

First, in some of the cases, the *anusvāras* were converted to corresponding *anunāsikas*.

Second, the inclusion of both the base and derived stems in the dataset with the help of the analyses from Heritage Engine, modifying the iics of certain compounds which were in the lemma format to the iic format, handling pre-verbs by gluing them to their corresponding word forms, etc were additionally done to modify the analyses of DCS.

Third, the analyses produced by the Heritage Segmenter has details like homonymy index which denotes the word sense. Mapping these with the senses of DCS words is a tedious task, as the mapping is one to many from DCS to Heritage's lexicon and hence, different senses of the same lemma were clubbed together in the analyses of the SH engine.

Fourth, many of the compounds which had non-compositional meaning were analysed as a whole entity. Sometimes, they have to be changed to components (iics and ifcs) depending on the context, which is out of the scope of current research.

Fifth, the main advantage of SH platform is the availability of morphological constructs which is obtained with the help of the generative tagset of the platform. And DCS provides abstract entities like CNG values which could have multiple analysis possibilities. The alignment process helped in mapping these CNG values with the morphological analyses, thus providing the resultant dataset both parameters.

SHMT corpus was used for comparisons of the performance of the Segmenter over different metrics, and different ranking algorithms. With the help of these comparisons, we can infer

that the improvements to the alignment process would result into an improved dataset, which would then provide us better statistics. In section 7.3, the closeness of the three generated corpora with respect to the recall of the engine was discussed (29/43 verses producing results). SHMT stood out with 33/43 verses producing correct results. But in table 4 the closeness of the SHMT corpus with the dataset generated from overall alignment was shown (53% of the verses obtained the correct solution with SHMT corpus; 46% with the latest dataset from latest alignment). Two thirds of the DCS sentences are to be analysed for issues. Also, amongst the aligned sentences, 130,000 had single solution possibilities. The remaining 85,000 sentences had multiple mappings. Truncating these into a single solution is beyond the scope of the current research as the different possibilities arise mostly due to compound formations and contextual differences.

In section 7.4.1, it was mentioned that out of 700 verses of the Bhagavad Gita, 278 verses were provided with additional segmentation solutions because of the unmatched named entities or compounds which either require compositional analyses or non-compositional analyses. This is again beyond the scope of the current research as the differences are present in the meanings of the verses and also it depends on the design decisions of the systems under use.

Finally, the comparison with the model from Hellwig and Nehrdich (2018) (section 7.4.2) provided us clear distinctions on how this updated segmenter is better with respect to differentiating the segmentation within compounds and that between different words, providing limited multiple solutions owing to the different meaning possibilities, etc. The updated Segmenter did outperform the model with more number of verses analysed correctly for the Bhagavad Gita test verses. Although the neural network model performed better for the test sentences from Hellwig and Nehrdich (2018), when the compositionality of the compounds was included amongst the ground truth segmentations, the updated Segmenter did perform better. This showed the importance of compositionality at the segmentation level and how the Heritage Segmenter needs to handle the compositionality of the compound words. It also brought out the necessity of updating the lexicon with the unrecognized words. Other computational parameters like response time, training time and space, etc are also to be considered to propose a strong comparison of these two systems.

The intention behind this work was to provide a dataset which is rich in morphological and lexical analyses, and which provides as much details as possible regarding the sentences and words. The alignment process did pave the way to building such a dataset which was used on the existing Segmenter to improve it further. Thus, the resultant dataset could give rise to a homogeneous gold corpus which could in turn be used for various further tasks of sentential analysis.

References

Sushant Dave, Arun Kumar Singh, Prathosh A. P., and Brejesh Lall. 2021. Neural compound-word (sandhi) generation and splitting in sanskrit language. In *CODS-COMAD 2021: 8th ACM IKDD CODS and 26th COMAD, Virtual Event, Bangalore, India, January 2-4, 2021*, pages 171–177. ACM.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Linguistic Modeling*, 4(2):117–126.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In *COLING*, pages 1011–1028.

Anilkumar Gupta. 2012. *Sanskrit Compound Processor*. Ph.D. thesis, University of Hyderabad, Hyderabad.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium, October-November. Association for Computational Linguistics.

Oliver Hellwig. 2009. Sanskrittagger, a stochastic lexical and pos tagger for Sanskrit. In *Lecture Notes in Artificial Intelligence*.

Oliver Hellwig, 2010–2019. *The Digital Corpus of Sanskrit (DCS)*.

Oliver Hellwig. 2016. Detecting sentence boundaries in Sanskrit texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 288–297, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Gérard Huet and Amba Kulkarni. 2014. Sanskrit linguistics web services. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 48–51.

Gérard Huet and Idir Lankri. 2018. Preliminary design of a Sanskrit corpus manager. In Gérard Huet and Amba Kulakrni, editors, *Proceedings of Computational Sanskrit and Digital Humanities, 18th WSC*.

Gérard Huet and Benoît Razet. 2015. Computing with relational machines. *Mathematical Structures in Computer Science*, pages 1–20, October.

Gérard Huet. 2003. Lexicon-directed segmentation and tagging of Sanskrit. In *XIIth World Sanskrit Conference, Helsinki, Finland. Final version in Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich.*, pages 307–325, Delhi, August. Motilal Banarsidass.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614.

Amrith Krishna, Pavankumar Satuluri, and Pawan Goyal. 2017. A dataset for sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, page 105–114. Association for Computational Linguistics, August.

Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, and Pawan Goyal. 2018. Free as in free word order: An energy based model for word segmentation and morphological tagging in Sanskrit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2550–2561, Brussels, Belgium, October-November. Association for Computational Linguistics.

Amrith Krishna, Ashim Gupta, Deepak Garasangi, Pavankumar Satuluri, and Pawan Goyal. 2020a. Keep it surprisingly simple: A simple first order graph based parsing model for joint morphosyntactic parsing in Sanskrit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4791–4797, Online, nov. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2020b. A graph-based framework for structured prediction tasks in Sanskrit. *Computational Linguistics*, 46(4):785–845, December.

Sriram Krishnan and Amba Kulkarni. 2019. Sanskrit segmentation revisited. In *Proceedings of ICON 2019, the International Conference on Natural Language Processing*, Hyderabad, December.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177.

Amba Kulkarni. 2019. *Sanskrit Parsing based on the theories of Śābdabodha*. IIAS, Shimla and D K Printworld.

Amba Kulkarni. 2021. Sanskrit parsing following indian theories of verbal cognition. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–38, April.

Peter Scharf and Malcolm Hyman. 2009. *Linguistic Issues in Encoding Sanskrit*. Motilal Banarsidass, Delhi.