Spellchecker for Sanskrit: The Road Less Taken

Prasanna Venkatesh T S
Ph.D. Research Scholar
Department of Sanskrit
Ramakrishna Mission
Vivekananda College (Autonomous),
Chennai - 600004.
vipranarayan14@gmail.com

Abstract

A spellchecker is essential for any language for producing error-free content. While there exist advanced computational tools for Sanskrit, such as word segmenter, morphological analyser, sentential parser, and machine translation, a fully functional spellchecker is not available. This paper presents a Sanskrit spellchecking dictionary for Hunspell, thereby creating a spellchecker that works across the numerous platforms Hunspell supports. The spellchecking rules are created based on the Paninian grammar, and the dictionary design follows the word-and-paradigm model, thus, making it easily extendible for future improvements. The paper also presents an online spellchecking interface for Sanskrit developed mainly for the platforms where Hunspell integration is not available yet.

1 Introduction

A spellchecker is a program that checks for misspellings in a text and suggests correct alternatives (Lawaye and Purkayastha, 2016). It is an essential tool for word processing and document preparation.

In the recent decade, digitization of Sanskrit manuscripts and using digital technologies for editing and publishing new content in Sanskrit have seen a tremendous increase. Online Sanskrit communities and many individuals are actively using Sanskrit in writing blogs, emails and chat messages, while some are designing posters in Sanskrit. A spellchecker can help them to communicate clearly and produce error-free content.

After the digitization of a manuscript through an OCR (optical character recognition), often there might be errors in the text due to the visual similarity in certain letters such as ब-व and य-थ (Schnober et al., 2016). These can be corrected using a spellchecker manually or through a pipeline that performs the OCR post-correction automatically. Also, advanced tools like morphological analyser (Kulkarni and Shukl, 2009; Huet, 2005) and sentential parser (Kulkarni, 2019) might produce incorrect results or no results at all, if the input contains spelling errors (Murthy et al., 2012). In such cases, the input can be pipelined through a spellchecker for pre-checking and pre-processing.

A spellchecker can also sometimes help new learners of Sanskrit in learning the correct spellings of words, especially, of those having phonetically similar letters such as श-ष in "शेष", ए-ये in "एक", न-ण in "बाणेन", etc.

This paper presents a Sanskrit spellchecking dictionary for Hunspell based on the word-and-paradigm model and describes its design and implementation. It discusses the suitability of Hunspell for a highly inflectional language like Sanskrit (Section 3) and the format of a Hunspell dictionary (Section 4). It also describes the framing of spellchecking rules based on Paninian grammar (Section 5). The paper also presents a web interface for Sanskrit spellchecking (Section 6). Later, it discusses the evaluation of both the spellchecking dictionary and the web interface (Section 7). The paper also records the challenges unique to Sanskrit in creating a spellchecking dictionary (section 8).

2 Related Works and Previous Attempts

While there exist advanced computational tools for Sanskrit such as word segmenter (Hellwig and Nehrdich, 2018), morphological analyser (Kulkarni and Shukl, 2009), sentential parser (Kulkarni, 2019), machine translation (Kulkarni, 2009), automatic speech recognition (Adiga et al., 2021), etc., a fully functional spellchecker is not available. Spellcheckers and grammar checkers are fundamental tools that users need and expect nowadays with the increase in the use of digital technologies for both creating new content as well as digitising existing texts.

Significant work has been done on spellcheckers for other Indian languages like Hindi (Kaur and Singh, 2015; Pathan et al., 2019; Kanwar et al., 2017; Jain et al., 2018), Marathi (Dixit et al., 2016), Odia (Pradhan and Dalai, 2020), Punjabi (Lehal, 2007), Kashmiri (Lawaye and Purkayastha, 2016), Telugu (Uma Maheshwar Rao G. et al., 2012), Tamil (Segar and Kengatharaiyer, 2015) and Kannada (Murthy et al., 2012, 2017).

Though there is not much research work available on spellchecker for Sanskrit, there were a few attempts in the past to develop one. Tapaswi et al. (2012) proposed and developed a spellchecker based on the morphological rules of Sanskrit. It was a standalone application implemented in Java. Samsādhanī[1] developed and hosted a spellchecker web application where their morphological analyser runs on the text provided by a user and highlights incorrect words. The application, however, is no longer maintained[2]. Patel (2016) built a Sanskrit spellchecker that works based on different vowel and consonant patterns. But it was specific to the Cologne Sanskrit dictionaries, and not for general spellchecking (Patel, 2021). Gasuns (2013) and Kumar (2017) independently created Sanskrit dictionaries for Hunspell. But, as of now, both the dictionaries are not complete and are not maintained. Quintanilha and Líbera (2018) developed a dictionary add-on for Mozilla Firefox which contains a Sanskrit Hunspell dictionary. However, it is also far from complete.

Other than these, to the best of our knowledge, there has been no significant progress in the development of a Sanskrit spellchecking dictionary for Hunspell or any other spellchecker for Sanskrit.

[1]`https://scl.samsaadhanii.in/scl/`

[2]Dr. Amba Kulkarni, personal communication.

3 Why Hunspell?

Hunspell is a free open-source spellchecker and morphological analyser library and also a command-line tool[3]. It is the most popular spellchecker that is used in many applications like LibreOffice[4], Mozilla Firefox[5], Google Chrome[6] and Adobe InDesign[7] and has bindings in numerous popular programming languages (Németh, 2019). In Linux and macOS, after installing the dictionaries the users can enjoy system-wide spellchecking. Such tight integration helps correct misspellings even as the user is writing instead of having them copy and paste the text into an external program just for spellchecking. Therefore, designing a dictionary for Hunspell means we can have Sanskrit spellchecking on almost all the platforms.

Sanskrit is a highly inflectional language and highly productive in derivative morphology such as *Samāsa*, *Kṛdanta* and *Taddhita* (Adiga et al., 2018). A simple list of all the correct forms would be extremely huge in terms of computer storage. Hunspell format allows us to define the base forms and affixes separately which hugely reduces the dictionary size. This division also makes it easier to add new words to the dictionary.

4 Format of Hunspell Dictionary Files

Before going into the design and preparation of the Hunspell dictionary, the format of the dictionary files is briefly discussed here. Hunspell requires two files to define how it should spellcheck for a language and suggest correct alternatives[8] – **a dictionary file** and **an affix file**. The first line of the dictionary file contains the (approximate) count of the number of entries in the file (Németh, 2018). From

[3]`https://github.com/hunspell/hunspell/`

[4]`https://extensions.libreoffice.org/?q=dictionary`

[5]`https://addons.mozilla.org/en-US/firefox/language-tools/`

[6]`https://chromium.googlesource.com/chromium/deps/hunspell_dictionaries/`

[7]`https://helpx.adobe.com/indesign/kb/add_cs_dictionaries.html`

[8]The dictionary and affix file format is discussed here only briefly to understand the design of the current dictionary. For more information refer (Németh, 2018) and (Shepelev, 2021).

sa_IN.dic	**sa_IN.aff**
37058	SFX 1001 Y 17
...	SFX 1001 िस् ीः .
आशिस्/1001	SFX 1001 स् षौ .
अर्चिस्/1001	SFX 1001 स् षः .
भुविस्/1001	SFX 1001 स् षम् .
...	...

Table 1: Samples from the final **dictionary** (sa_IN.dic) and **affix** (sa_IN.aff) files.

the second line onwards, we have the entries, one per line. An entry can be a morpheme or lexeme or can even be a pair of words, and it can be optionally followed by a forward slash ("/") and one or more "flags" which represent its attributes such as suffix, prefix, etc. Table 1 (sa_IN.dic) is a sample from the final dictionary file showing the entries' count, some entries and their flags.

The definitions of these flags are given in the **affix file**. Table 1 (sa_IN.aff) is a sample from the affix file showing the **affix class** corresponding to the flag "1001", used in the dictionary. An affix class definition consists of a **header** (the first line) followed by a number of **affix rules**, each separated by a new line. The affix header consists of four fields describing:

- Whether the class is for a suffix or prefix.
- The paradigm type of the affix.
- Whether the words of this class can take both suffix and prefix or only one of them.
- The number of rules in this class.

Here, `SFX 1001 Y 17` states that this is a suffix class with paradigm type "1001" which has 17 rules, and these rules can be used even if the entry has prefixes.

The fields in the affix rules give the information on the characters to strip from the word, the affix to add to the word and the condition. For example, the first affix rule in Table 1 says, in the words with this flag ("1001"), strip िस् from the end and add the suffix ीः to form a valid word. The `condition` field is a regex-like expression that is checked from the end of a word for a suffix rule and from the start for a prefix rule (Németh, 2018). If there is nothing to be stripped from the word, the stripping is written as "0" (i.e. `SFX <flag> 0 <suffix> <condition>`). And if no suffix is to be added to the word, it is represented with "0" (i.e. `SFX <flag> <stripping> 0 <condition>`).

In the dictionary file, the affix flag "1001" is assigned to the word "आशिस्" (Table 1). So, the first affix rule strips िस् from the end of "आशिस्" to form "आश" (note that it does not have a *virama*). Then, the rule adds ीः to the end of "आश" to form "आशीः" which is the nominative singular form of "आशिस्" and therefore, a valid word.

The affix file is not only for defining affix rules. It is also used for containing a lot of options that help improve Hunspell's spellchecking process for the language which would be discussed later (Sections 5.2 and 5.3). In the following section, we describe the preparation of the dictionary and affix files.

5 Design and Preparation of the Dictionary

For designing the dictionary, we follow the word and paradigm model, which is computationally easy to work with. We prepare the dictionary entries and affix rules for nouns and verbs using two different methods, as we would describe in the next section (Section 5.1). Sections 5.2 and 5.3 discuss the use of some of Hunspell's options for improving the suggestions and handling optional characters.

5.1 Preparation of Words and Affix Rules

5.1.1 Nouns

Sanskrit has only two morphological classes at the inflectional level viz. noun and verb. In almost all the nouns, only the last few letters of the nominal base change when it combines with a nominal suffix. For example, the base, नदी, when it combines with the nominative dual suffix औ its final vowel ई is replaced by य् and becomes नद्यौ. Similarly, when भगवत् combines with instrumental dual suffix भ्याम्, the final त् is replaced by द् and becomes भगवद्भ्याम्. For such forms, the affix rules are written with the letters needed to be removed from the *base*, in the stripping field, and the suffix to be added, in the suffix field (Table 2). But

Form	Stripping	Suffix	Affix Rule
नदी	◌ी	◌ौ	SFX 1 ◌ी यौ .
भगवद्भ्याम्	त्	द्भ्याम्	SFX 2 त् द्भ्याम् .
रामस्य	0	स्य	SFX 3 0 स्य .
हरिः	0	◌ः	SFX 4 0 ◌ः .
सीतायाः	0	याः	SFX 5 0 याः .

Table 2: Affix rules for different *subanta* forms

in some cases, the base remains unchanged in its declined form, such as रामस्य, हरिः, गुरुभ्याम्, सीतायाः, भगवत्सु, etc. For such forms, the affix rules are written with "0" in the stripping field (Table 2).

The affix rules cannot be framed for some of the noun forms that completely differ from their bases (irregular forms). For example, दुह् becomes धुक् in nominative singular[9] and अस्मद् becomes मम in genitive singular. These being exceptional cases, the corresponding forms are directly added to the dictionary. Similarly, the indeclinables, which do not have any suffixes, are also directly added to the dictionary without any affix flags.

5.1.2 Verbs

The verb morphology is much more complex than the noun morphology. The verbs in certain tenses or moods have both prefix as well as suffix. Further, in a certain tense (*liṭ lakāra*) there is a reduplication of the verbal stem. We do not discuss the morphological details at length here due to the constraints in the page size. However, the general morphological structure of the finite verb forms is:

UPASARGA (prefix) + **A/Ā**[10] (prefix) + **DHĀTU** (root) + **VIKARAṆA** (tense marker) + **TIṄ** (verbal suffix)

We have here two prefixes and two suffixes. But Hunspell supports[11] only one prefix and two suffixes for a word (Németh, 2018). Hence, we merge A/Ā-prefix with the UPASARGA-prefix to form a single prefix. Also, we merge the root with the tense marker to form the stem thereby reducing the suffix to one.

Due to this limitation of the Hunspell to use at the most a single prefix and at the most a single suffix, we have to transfer the load of base formation under different environments to the dictionary, thereby resulting in more than one stem corresponding to each verb. For example, the root, स्पर्ध् has two entries in the dictionary corresponding to it – स्पर्ध and पस्पर्ध. With regard to the affix rules for the finite verb forms, they are almost same for a given tense or mood.

5.2 Improving Suggestions

For Hunspell to accurately suggest correct alternatives for an incorrect word, it needs a list of characters used in the script and a list of common misspellings. We provide these lists using the `TRY` and `REP` options, respectively, in the affix file.

5.2.1 Try Characters

Using the `TRY` characters, Hunspell can suggest the correct words when the misspelled words differ from them by a single character (Németh, 2018). Hunspell adds, deletes, or replaces one of these characters to suggest the closest valid dictionary word for the misspelled word (Shepelev, 2021).

`TRY` characters are more effective if they are in the order of their frequency in the literature (Németh, 2018; Shepelev, 2021). For this, we use the Amarakośa of Amarasiṃha which contains the frequently used words in classical Sanskrit and calculate the frequency of the characters[12]. We, then, add the characters to the `TRY` option in the descending order of their frequency as shown below:

`TRY` ◌् ◌ा र त ◌ि क स न व य म प ◌ु ◌ः द ◌ो ◌े ल ◌ं ◌ी श ष ग च ण ध ह ज भ ◌ौ ◌ृ ◌ू थ ऽ ब ट ङ ड ◌ै अ ख ञ घ ठ छ आ उ फ इ ऋ ढ ए झ ऊ ओ ई ऐ औ[13]

[9]दादेर्धातोर्घः *Aṣṭādhyāyi 8/2/32* and एकाचो बशो भष् झषन्तस्य स्ध्वोः *Aṣṭādhyāyi 8/2/37*.

[10]Only in some tenses/moods (*laṅ*, *luṅ* and *lṛṅ lakāras*).

[11]Hunspell also supports two prefixes and one suffix when the COMPLEXPREFIXES option is set. But it is mainly used by languages with a right-to-left writing system.

[12]The algorithm and the results are available at: https://github.com/vipranarayan14/sanskrit-char-frequency

[13]Space-separated for readability

5.2.2 Replacement Definitions

Replacement definitions are provided for handling typical spelling mistakes (Németh, 2018). Based on these replacement definitions, Hunspell makes some replacements in the misspelled word to find the valid forms from the dictionary and suggests the same. Let us see an example. Consider the word “रामह”. It is not a morphologically valid word. We know that the closest valid alternative is “रामः”. But, using only the `TRY` characters, Hunspell will suggest many correct alternatives of which “रामः” is not even among the first three (Figure 1). This is where the `REP` (replacement) definitions become very helpful.

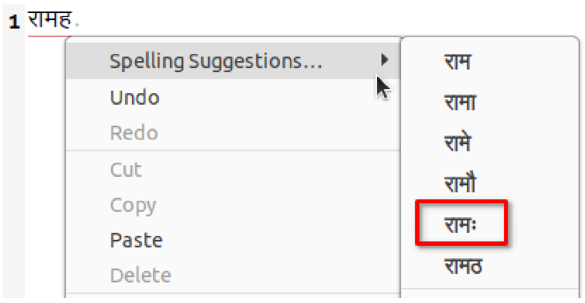


Figure 1: Correct suggestion in the 5th position.

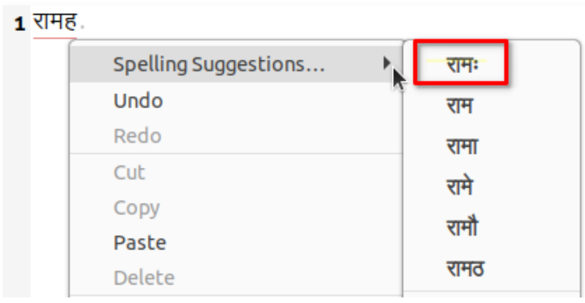


Figure 2: Correct suggestion in the 1st position.

When the replacement definitions shown in the Table 3 are added to the affix file we can see in Figure 2 that “रामः” becomes the first suggestion for the misspelled word “रामह”. This is also because Hunspell gives `REP` suggestions the highest priority in the suggestion list (Németh, 2018; Shepelev, 2021).

First line in the replacement table (Table 3) is the **header**. It tells that there are 125 `REP` (replacement) definitions in the table. The rest of the table consists of the replacement definitions. We use the replacement definitions for suggesting closer valid alternatives

REP 125	
REP ह ः	REP व ब
REP म् ं	REP ब व
REP ऑ ओ	REP य थ
REP ि ी	REP थ य
REP अ आ	REP ध घ
REP आ अ	REP घ ध
REP क ख	…
REP श ष	
…	

Table 3: Some of the replacement definitions used in the dictionary.

for words that are misspelled due to phonetic similarity or visual similarity.

The reasons for adding replacement definitions for **phonetically similar** characters are:

- Beginners in Sanskrit would not be familiar with the spellings, especially, of words that have some of the phonetically similar characters, such as श-ष in शेष, क-ख in कर, न-ण in बाणेन. Sanskrit has a lot of such words.

- People who are not trained with the INSCRIPT keyboard layout would tend to use the phonetic keyboard layouts for writing Sanskrit. So, their typing errors, for the most part, would also be phonetic.

- Phonetic typing errors also occur if the writer is not familiar with the input schemes such as ITRANS, Velthuis, Harvard-Kyoto, etc. since these popular schemes are themselves based on the Sanskrit phonemes. In Figure 2, the user, intending to write “रामः” using an ITRANS phonetic keyboard layout, has typed “`raamaha`” which resulted in “रामह” (since the key for *visarga* in ITRANS scheme is “H”; not “ha”). Now, the spellchecker using the replacement definitions, correctly suggests the right alternative “रामः”.

The reasons for adding replacement definitions for **visually similar** characters are:

- They can help in correcting spelling errors found in post-OCR documents. For example, an OCR incorrectly recognises the word “मिथ्या” as “मिय्या” due to the visual similarity between “थ” and “य”. The spellchecker using the replacement definitions can help the user to correct it to “मिथ्या”.

- Users using Devanagari keyboards such as the INSCRIPT keyboard make spelling errors more due to the visual similarity of characters than their phonetic similarity.

5.3 Handling Optional Characters

There are many other options in Hunspell for improving the overall working of the spellchecker. One such option is `IGNORE` which can be used to make Hunspell ignore certain characters when spellchecking. Using this option, we ignore the character “ऽ” *(avagraha)* for allowing words such as “काऽपि” and “इतोऽपि” to be written without it as “कापि” and “इतोपि”. Because both forms are considered correct and both are in vogue.

In this section, we described the preparation of the spellchecking dictionary. The following section briefly describes the development of a web interface for Sanskrit spellchecking.

6 Web Interface

A web interface[14] is developed using the spellchecking dictionary and Hunspell, mainly, for supporting platforms such as Android and iPhone where Hunspell integration is currently not available. However, it can be used in desktop browsers also. Sanskrit text in Devanagari script can be typed in or pasted into the editor. The editor marks the incorrect words with a red underline. If an underlined word is clicked, a pop-up window opens next to the word showing a list of correct suggestions (Figure 3). When a suggestion is clicked, the incorrect word is replaced with the suggested word and the pop-up closes.

7 Evaluation

After the Hunspell dictionary was prepared, all the forms of the word paradigms were tested against it with the help of Nodehun[15], a Node.js binding for the Hunspell library, and Mocha[16], an automated JavaScript testing framework. In the test, all the forms were recognised as correct spellings.

To evaluate the spellchecking dictionary, a test corpus was prepared from three random OCR-ed pages of Śrīmad Vālmīki Rāmāyaṇa from the Sanskrit Wikipedia[17]. The *sandhis* and *samāsas* in the corpus were manually split since the dictionary does not support such words. The corpus consisted of 751 unique words.

The dictionary was added to Hunspell and, using a script written in Python, each word in the corpus was tested with the Hunspell spellchecking interface. The Hunspell dictionaries prepared by Kumar (2017) and Líbera (2018) were also tested in a similar manner for comparison. The results were then manually analysed.

Dictionary	Words	AC	AI	RC	RI
A	751	720	31	501	250
B	751	720	31	27	724
C	751	720	31	5	746

Table 4: Comparison of results of dictionaries A, B and C. AC = Actual correct words, AI = Actual incorrect words, RC = Words recognised by the dictionary as correct, and RI = Words recognised by the dictionary as incorrect.

Table 4, shows the comparison of results of the three dictionaries, viz. dictionary (A) proposed in this paper, (B) the one prepared by Kumar and (C) which was developed by Líbera. (A) performed very well compared to (B) and (C). The main reason for the poor performance of (B) and (C) was their limited vocabulary. (B) has only 3228 words in the dictionary file while (C) has only 838 Devanagari words. Though (B) uses affix rules, not many forms are supported by them. These affix rules were generated by the `affixcompress` tool[18] provided by Hunspell and are not grammar-based[19].

In the case of (A), all the words declared as correct were also actually correct implying a

[14] It can be accessed at: https://sanskrit-spellchecker.netlify.app

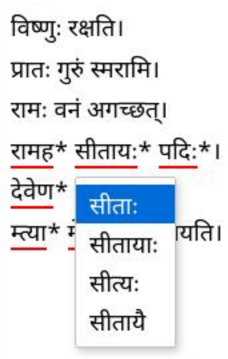
[15] https://www.npmjs.com/package/nodehun

[16] https://mochajs.org/

[17] https://sa.wikisource.org/s/2idh

[18] The tool is used for creating a spellchecking dictionary for Hunspell out of a list of words – https://github.com/hunspell/hunspell/.

[19] https://github.com/Shreeshrii/hindi-hunspell/issues/1#issuecomment-282244359


विष्णु: रक्षति।

प्रात: गुरुं स्मरामि।

राम: वनं अगच्छत्।

रामह* सीताय:* पदि:*।

देवेण*

म्त्या* म यति।

Figure 3: A screenshot of the web interface. All the words marked with * are incorrect. The editor correctly marks them with a red underline. For the incorrect word "सीतायः", the correct form "सीतायाः" is suggested.

100% precision. A few real-word errors were observed but they were not considered for this evaluation as Hunspell cannot handle such errors. Of the words reported as incorrect, a large percentage were wrongly considered incorrect (false negatives). The reasons were mainly traced to incomplete vocabulary and incorrect split of *sandhi* and *samāsa*. Since the dictionary is still in development false negatives due to out-of-vocabulary words are expected. Moreover, for better evaluation and for increasing the accuracy of suggestions, we are in the process of creating a larger test corpus.

The web interface was also manually tested using random words and sentences. The editor correctly marked the misspellings and also suggested correct and proper alternatives for them (Figure 3).

8 Observations

Some of the observations noted while preparing and evaluating the spellchecking dictionary are discussed here.

- **Use of Devanagari:** Though Unicode Devanagari is neither fully alphabetic nor syllabic in nature, but a combination of the two, and Pāṇini's word formation and euphonic rules operate at the alphabetic level, we decided to write the rules for Devanagari rather than other more suitable transliteration schemes such as WX or SLP1, that provide a one-to-one mapping between the Sanskrit phonemes and its romanised representation. The reason for this is, by doing so, we can avoid an intermediary step of converting the input into roman transliteration and then, converting the suggestions back into Devanagari which would require close interaction with the Hunspell APIs. But this would also mean, developing an altogether fresh set of rules if somebody uses IAST for Sanskrit.

- **Word-and-paradigm model:** The paradigm-based approach is advantageous for the preparation of the Hunspell dictionary for Sanskrit as it reduces the number of dictionary entries. Even though Sanskrit is a highly inflectional language, Pāṇini's grammar helps to reduce the forms of most of its words to a few hundred paradigms. We need to make affix rules only for these few hundred paradigms. For all other words having similar forms we just have to add the affix flag associated with the paradigm. This reduces the dictionary's size and at the same time, makes it easy to maintain. This approach also simplifies the process of adding new words. For example, if a new word, "कामदेव" is to be added, we add it to the dictionary file along with the flag associated with its paradigm, "देव". This way, we are able to add support for not only कामदेव but also all its declined forms without much effort.

- **Handling compounds:** The spellchecker's performance drops when there are compounds in the text. Though there are options for writing compound rules in Hunspell, it would not be possible to support all the compounds since Sanskrit is highly productive in terms of compound generation. Therefore, it is better to pipeline the user input into an existing *sandhi-samāsa* splitter like that of Hellwig's (2018) or Huet's (2005) before spellchecking it. This would again require close integration with the Hunspell APIs. Further, there are some words which cannot be handled even if the text is preprocessed with a *sandhi-samāsa* splitter. For example, the ending of some words changes in their compounded forms – राजा becomes राज in महाराज and आत्मा becomes आत्म in आत्मदुश्चरितः. These forms are valid only within the compound and not otherwise. To overcome this problem, such compounded words have to be directly added to the dictionary.

- **Spelling variations:** Words with *anusvāra* (ं) such as संभ्रमः and असंबाधम् optionally undergo homo-organic nasalisation and become सम्भ्रमः and असम्बाधम्, respectively. Both the forms are frequently used alternatively in the literature. So both forms had to be added to the dictionary. Also, the letter म् at the end of word is replaced by *anusvāra* if it is followed by a consonant[20]. So, for suffixes ending with म्, extra affix rules had to be created with *anusvāra* in the place of म्.

- **Context-dependent spelling errors:** Some of the spelling variations are context-dependent. For example, as mentioned above, the *anusvāra* at the end the word is valid only if the next word starts with a consonant. Similarly, verb forms which have the *a/ā*-prefix in the conditional mood, will drop it when they are preceded by the word मा[21]. For example, मा भैषीः and मा भवान् कार्षीत् instead of मा अभैषीः and मा भवान् अकार्षीत्, respectively. Such cases cannot be handled by the Hunspell since it only looks at one word at a time. Further, as is true with any spellchecker, it is impossible to detect wrong words in a given context that are correct otherwise (Deorowicz and Ciura, 2005). For example, use of ताम् instead of वाम्, where both the words are correct spelling-wise.

9 Conclusions and Future Work

The paper proposed and described the design and preparation of a Hunspell dictionary for Sanskrit. It also discussed the advantages of the paradigm-based approach which was followed for the generation of the dictionary. The proposed dictionary is grammar-based unlike that of Kumar (2017) and is general-purpose, that is, it can be used for correcting both typing errors and OCR errors. The paper also presented an online spellchecking interface for Sanskrit.

The current limitations of the dictionary and future improvements to be made are discussed below.

- The spellchecking dictionary is a work in progress. Currently, it supports nouns, indeclinables, pronouns and numbers (cardinals and ordinals). It also supports active and middle verb forms. Complex verb forms (*sannantas*), non-finite verb forms (*kṛdantas*), participles, etc. and secondary nominal derivatives (*taddhitas*) are yet to be supported.

- Hunspell's `` `KEY` `` option improves suggestions for words misspelled due to the proximity of letters on the keyboard. This can be utilized for INSCRIPT and phonetic keyboard layouts.

- Sanskrit is also written in many roman transliteration schemes such as IAST, ITRANS, etc. The current spellchecking dictionary works only for Devanagari. Special spellchecking dictionaries have to be made at least for the popular transliteration schemes of Sanskrit.

- Extensions/add-ons for more platforms/softwares/operating systems have to be developed so that users can enjoy the benefits of the dictionary in their favourite writing tools and environments.

Acknowledgements

I thank Dr. S. Muthu for his valuable support and encouragement. I thank Dr. Amba Kulkarni for all the resources and guidance. If not for Victor Shepelev's detailed blogs on Hunspell library (Shepelev, 2021), I could not have understood its format and inner workings. His detailed descriptions and clear-cut examples for the Hunspell format made it easy for me to develop the dictionary. Finally, I would like to thank the reviewers for their valuable feedback and suggestions.

[20]मोऽनुस्वारः *Aṣṭādhyāyi 8/3/23*

[21]न माङ्योगे *Aṣṭādhyāyī 6/4/74* and स्मोत्तरे लङ् च *Aṣṭādhyāyī 3/3/176*

References

Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. 2021. Automatic Speech Recognition in Sanskrit: A New Speech Corpus and Modelling Insights. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5039–5050, Online. Association for Computational Linguistics.

Devaraja Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K. Ramasubramanian, and Malhar Kulkarni. 2018. Improving the learnability of classifiers for Sanskrit OCR corrections. In *Computational Sanskrit & Digital Humanities: Selected Papers Presented at the 17th World Sanskrit Conference*, Vancouver, Canada.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 2002. Natural Language Processing: A Paninian Perspective.

Sebastian Deorowicz and Marcin Ciura. 2005. Correcting spelling errors by modelling their causes. *Int. J. Appl. Math. Comput. Sci*, 15:275–285.

Veena Dixit, Satish Dethe, and Rushikesh K Joshi. 2016. Design and Implementation of a Morphology-based Spellchecker for Marathi, an Indian Language. *International Journal of Science Technology and Engineering*, 3(2):92–96.

Marcis Gasuns. 2013. Sanskrit-Hunspell. `http://samskrtam.ru/sanskrit-hunspell/`.

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. Association for Computational Linguistics.

Gérard Huet. 2005. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *Journal of Functional Programming*, 15(4):573–614. Publisher: Cambridge University Press.

Amita Jain, Minni Jain, Goonjan Jain, and Devendra K. Tayal. 2018. "UTTAM" An Efficient Spelling Correction System for Hindi Language Based on Supervised Learning. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 18(1):8:1–8:26.

Shailza Kanwar, Manoj Sachan, and Gurpreet Singh. 2017. N-Grams Solution for Error Detection and Correction in Hindi Language. *International Journal of Advanced Research in Computer Science*, 8(7):667–670.

Baljeet Kaur and Harsharndeep Singh. 2015. Design and Implementation of HINSPELL -Hindi Spell Checker using Hybrid approach. *International Journal of Scientific Research and Management*, 3(2). Number: 2.

Amba Kulkarni. 2009. *Anusaaraka: An approach for MT taking insights from the Indian Grammatical Tradition.* PhD, University of Hyderabad, Hyderabad.

Amba Kulkarni. 2019. *Sanskrit Parsing: Based on the Theories of Shabdabodha.* D.K. Print World Ltd, Shimla : New Delhi.

Amba Kulkarni and Devanand Shukl. 2009. Sanskrit Morphological analyzer: Some Issues. *Bh. K Festschrift volume by LSI.*

Shreedevi Kumar. 2017. Sanskrit Hunspell. `https://github.com/Shreeshrii/hindi-hunspell/tree/master/Sanskrit`. Original-date: 2014-12-03.

Aadil Lawaye and Bipul Syam Purkayastha. 2016. Design and Implementation of Spell Checker for Kashmiri. *International Journal of Scientific Research*, 5:199–200.

Gurpreet Singh Lehal. 2007. Design and Implementation of Punjabi Spell Checker. *International Journal of Systemics, Cybernetics and Informatics.*

S Rajashekara Murthy, A. N. Akshatha, Chandana G Upadhyaya, and P. Ramakanth Kumar. 2017. Kannada spell checker with sandhi splitter. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 950–956.

S Rajashekara Murthy, Vadiraj Madi, Sachin D, and Ramakanth Kumar P. 2012. A Non-Word Kannada Spell Checker Using Morphological Analyzer and Dictionary Lookup Method. *International Journal of Engineering Sciences & Emerging Technologies*, 2(2):43–52.

László Németh. 2018. Hunspell – Format of Hunspell dictionaries and affix files. `https://github.com/hunspell/hunspell/releases/download/v1.7.0/hunspell5.pdf`.

László Németh. 2019. Hunspell: About. `http://hunspell.github.io/`.

Dhaval Patel. 2016. SanskritSpellCheck. `https://github.com/drdhaval2785/SanskritSpellCheck`. Original-date: 2014-10-04.

Dhaval Patel. 2021. Hunspell for Sanskrit? - Issue #91 - sanskrit-lexicon/COLOGNE. `https://github.com/sanskrit-lexicon/COLOGNE/issues/91`.

Shabana Pathan, Nikhil Khuje, and Punam Kolhe. 2019. A Survey on Creation of Hindi-Spell Checker to Improve the Processing of OCR. *International Journal of Research in Engineering, Science and Management*, 2(3):517–519.

Amrutanshu Pradhan and Sasanka Sekhar Dalai. 2020. Design of Odia Spell Checker with word Prediction. *International Journal of Engineering Research & Technology*, 8(1). Publisher: IJERT-International Journal of Engineering Research & Technology.

Adriana Quintanilha and Vinícius Della Líbera. 2018. Sanskrit Spell Checker. `https://addons.mozilla.org/en-US/firefox/addon/sanskrit-spell-checker/`.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Jananie Segar and Sarveswaran Kengatharaiyer. 2015. Contextual spell checking for Tamil Language. In *14th Tamil Internet Conference*, Singapore.

Victor Shepelev. 2021. Rebuilding the spellchecker: Hunspell and the order of edits. `https://zverok.space/blog/2021-01-28-spellchecker-5.html`.

Namrata Tapaswi, Dr Suresh Jain, and Mrs Vaishali Chourey. 2012. Morphological-based Spellchecker for Sanskrit Sentences. *International Journal of Scientific & Technology Research*, 1(3):1–4.

Uma Maheshwar Rao G., Amba P. Kulkarni, Christopher Mala, and Parameshwari K. 2012. Telugu Spell-checker. *Center for Applied Linguistics and Translation Studies University of Hyderabad Hyderabad, India.*