Kathmandu University Journal of Science, Engineering and Technology, Vol. 14, No. 1, April 2020

Kathmandu University
Journal of Science, Engineering and Technology

A vowel based word splitter to improve performance of existing Nepali morphological analyzers on words borrowed from Sanskrit

Madhusudan Adhikari[*] and Aatish Neupane

Department of Computer Science and Engineering, School of Engineering, Kathmandu University

Abstract

Nepali lexicon is rich in words borrowed from Sanskrit language, called "Tatsama" words. These words have a specific set of rules to split them into affixes and morphemes. This paper details these rules and describes an implementation that can be used to improve existing Nepali Morphological Analyzers which do not yet implement concrete rules for these kinds of words.

Keywords: Natural Language Processing(NLP); morphological analyzers; Sanskrit conjunction; Nepali morpho-analysis; Tatsama words

1. Introduction

Nepali lexicon consists of a wide category of words and one of them is Tatsama words: "those borrowed from Sanskrit language without any changes" [1]. These words follow some regular word formation patterns of Sanskrit and Sandhi is one of the word joining pattern in Sanskrit where letters or sounds combine to form a new word. In this pattern, words may be formed after replacement, addition or deletion of letters undergoing the conjunction (Sandhi) process. This process is used in a variety of situations like combination of two individual words and addition of affixes to the roots, of which examples are shown below:

Affix-Word combination:

Su (सु){prefix}+ Aagata (आगत){root} = Svagata (स्वागत)

Word-Word Combination:

Vikash (विकाश) +Unmukh (उन्मुख) = Vikashonmukh (विकाशोन्मुख)

There are many rules for Sandhi in Sanskrit depending on the letters involved. Understanding Sandhi is not just important to understand the structure of Sanskrit words formation but also important for those languages which borrow words from Sanskrit- like Nepali and Hindi since the rules for Tatsama words are completely based on Sanskrit grammar. Understanding this process is helpful in the field of Natural Language Processing (NLP) of Nepali language as it helps to optimize and improve the working of many NLP components like Morphological Analyzer (MA) and Stemmers for Nepali, specially rule-based ones.

A Morphological Analyzer (MA) is a common tool used in Natural Language Processing (NLP). It is a tool responsible for determining morphemes of a given word. A morpheme is a meaningful morphological unit of a language that cannot be further divided. A rule based MA is unique for every language as each language possesses its own process of forming inflected and derived words from the root. Normally rule-based MA requires a thorough understanding of the word structure of the language for its development.

[*]Corresponding author. Email: mdhsdnadhikari@gmail.com

Existing implementations of morphological analyzers for Nepali are rule-based [3]. These MA utilize a common Nepali word formation process to break down the words to get its root with stemmer at its core. In most cases, it strips off the suffixes and prefixes to reach the root. For instance:

Shahareeyaa (शहरीया) = Sahar(शहर){root}+eeyaa(ईया){suffix}

Videshee (विदेशी) = Vi (वि){prefix}+ desh (देश){root}+ee(ई){suffix}

This technique of stripping affixes to extract the root is very much helpful for words that follow a regular pattern and majority of words follow the same pattern but this may not work for Tatsama words. Nepali compound words forming as a result of the combination of the free and bound morphemes are not always regular in terms of formation and consequently in breaking. Insertion and deletion of one or more free vowels and vowel symbols or dependent vowels is a common phenomenon [3]. An example is:

Atyadhik(अत्यधिक) = Ati (अति){Prefix}+ Adhik (अधिक){root}.

The technique of stripping of affixes cannot work here as this type of words are Tatsama words and they follow rules from Sanskrit, not Nepali.

This paper focuses on development of an algorithm to implement Sandhi rules from Sanskrit as Sanskrit grammarian Panini described in his magnum opus Ashtadhyayi [2]. Though the algorithm being developed is for Sanskrit words, it can be utilized in contemporary NLP tools in Nepali like Morphological Analyzers and Stemmers so that it can deal with some categories of Tatsama words. Not all Sandhi rules are applicable to the Nepali language. This paper focuses on some of those Sandhi rules which are relevant for Nepali lexicon.

1.1. Status of NLP and morphological analyzer in Nepali

Morphological Analyzer is considered one of the basic building blocks necessary for further NLP works on the language in consideration. [3] points out that the research in NLP for Nepali started in the year 2005, with the release of the first Spell Checker for Nepali and the "Dobhase" English to Nepali machine translation project.. The first MA and Stemmer for Nepali was developed and described in [3], which utilized a rule-based approach, and it used a stemmer at its core. Besides a stemmer, it consisted of POS tagsets, Tokenizer, Free morpheme based lexicon, two sets of affixes each for the suffix and a database for word breaking grammatical rules. In another approach, context free hybrid stemmer which used traditional rule-based system with string similarity approach which was proposed in [4]. It used a threshold distance of 0.5 for stripped word with stems and after stripping the word, the words were compared with the roots stored in the database using string similarity function. A suffix removal stemmer for Nepali was developed by [5]. [6] developed a new stemmer for Nepali with suffix rules. They composed 128 suffix rules, which were executed in step-by-step and iterative manner to eliminate inflections. None of these rules based systems have implemented Sandhi rules for analysis.

1.2. Related works on Sandhi computation in Indo-Aryan languages

[7] has surveyed surveyed various sandhi splitting techniques for different Indian languages - Sanskrit, Hindi, Malayalam and Marathi . They also developed a Vowel Sandhi splitter for Sanskrit [8]. [9] presented rules and rule-based algorithm for sandhi splitting of Marathi compound words. Statistical sandhi splitter for agglutinative languages was developed by [10] and it was tested on Malayalam and Telugu.

2. Discussion

2.1. Sanskrit conjunction (Sandhi) rules relevant to Nepali

As introduced in earlier sections, Sanskrit has a process of combining two letters leading changes in the combining letters. Sanskrit grammarian Panini has mentioned that the application of rules follow a hierarchical order: one rule supersedes another if the conditions necessary are satisfied [2]. Though there are many rules in Sanskrit regarding Sandhi, only limited ones are used in Nepali. The common rules that are applied in the Nepalese lexicon, are represented in Table 1, and Table 2, and are based on the natures of Sandhi.

2.2. Proposed improvement to existing systems

The proposed implementation utilizes techniques developed by previous researchers in the field of Nepali morphological analysis and adds algorithm to break the words as per the Sanskrit conjunction (sandhi) rules. Prerequisites for the system to work are similar to that of the Morphological Analyzer developed by [3]: a lexicon based on free morphemes, a set of affixes, and an algorithm to break the words. The algorithm to break the words is what improves existing implementations to handle Tatsama words and hence, this paper focuses on that algorithm specifically. The general algorithm is as follows:

1. Start
2. Take two arrays, LHS (initially an empty list) and RHS (list of Unicode letters for the word)
3. Keep pulling letters from RHS into LHS (processing step) until both lists contain free morphemes or an affix and a morpheme (base condition)
4. Whenever a half consonant is found, check if any rules match current condition and create a copy of LHS and RHS with new replacements. Keep processing this replacement as a new word until base condition matches or all of RHS is exhausted.

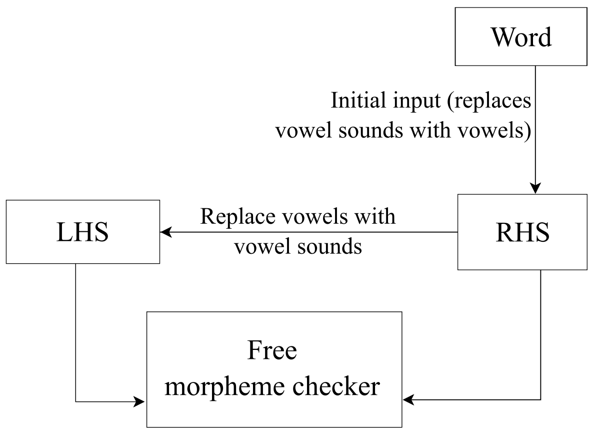

Word
Initial input (replaces vowel sounds with vowels)
LHS
Replace vowels with vowel sounds
RHS
Free morpheme checker

Figure 1: Simplified system flow.

5. Keep processing arrays which have not been replaced until base condition is matched or all of RHS is exhausted.
6. Stop

The general flow of the algorithm is shown in Fig. 1.

2.2.1. Processing step

During the processing step when a letter is transferred from RHS to LHS, isolated vowel sound modifiers such as ा (aa) ू (oo) ै (ai) end up in RHS. This is replaced by a corresponding vowel sound and then checked against the base condition. For instance, while processing a word ``बिचारोन्मुख", a state is such that:

LHS = ब् (B)

RHS = [िचारोन्मुख]

With the replacement, it becomes:

RHS = [इचारोन्मुख] (Icharonmukh). Here the vowel sound modifier sign is changed to a distinct vowel letter ``i".

A vice-versa replacement should happen when ``इ" (i) ends up in the LHS. So, a conversion table is necessary in both scenarios, which is shown in Table 3.

2.2.2. Base condition

Checking base condition requires a list of affixes and free morphemes. Both of these resources are very scarce for Nepali language and thus, we developed our own set of free morphemes to use with our algorithm.

Implementation:

The proposed improvement algorithm was implemented in Python 3 which supports Unicode processing very effectively. An accompanying web application was also developed which showed iterations along with applied rules. Some screenshots from the application we developed using the algorithm are shown in Fig. 2 and Fig. 3.

3. Results

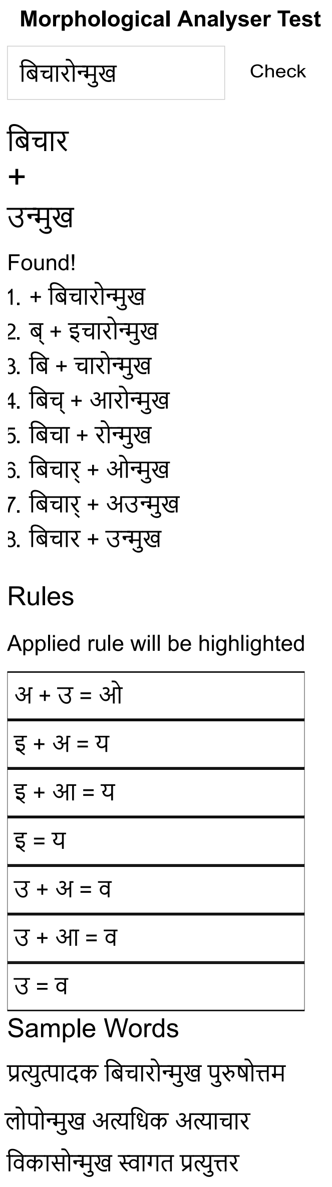
250 different Tatsama words which followed the sandhi rules were chosen from the dictionary [11] and were tested through the system. The technique worked correctly with all the words which were a combination of two entities (affixes-morphemes and morphemes-morphemes) based on vowels. However, it didn't work for words that were formed from consonant sandhi rules. One such word was दिग्दर्शन (Digdarshan). The word follows consonant sandhi rules and is broken down as:

Table 1: Replacement of a single letter.

Initial letter	Following letter	Replaced letter	Replacing letter	Examples
अ (a), आ (aa)	इ (i), ई (ee)	इ, ई	य् (ya)	Ati (अति) + Aachaar (आचार) = Atyaachaar (अत्याचार) Ati (अति) + Adhik (अधिक) = Atyadhik (अत्यधिक)
अ (a), आ (aa)	उ (u), ऊ (oo)	ऊ, उ	व (va)	Su (सु) + Aagata (आगत) = Svagata (स्वागत)

Table 2: Replacement of letters.

Combining letter 1	Combining letter 2	Replacement letter	Examples
Replacement of two letters:			
अ (a), आ (aa)	इ (i), ई (ee)	ए (ae) (अ/ आ + इ/ई = ए)	Upa (उप) + Indra (इन्द्र) = Upendra (उपेन्द्र)
अ (a), आ (aa)	उ (u), ऊ (oo)	ओ (O) (अ/आ + उ/ऊ = ओ)	Sarva (सर्व) + Uttam (उत्तम) = Sarvottam (सर्वोत्तम)
Replacement of one combining letter by another combining letter:			
अ (a)	आ (aa)	आ (अ + आ = आ) (Combining letter 2)	Pustaka (पुस्तक) + Aalaya (आलय) = Pustakaalaya (पुस्तकालय)
Combination of repeated letters:			
अ (a)	अ (a)	आ (अ + अ = आ)	Eka (एक) + Aadhikar (अधिकार) = Ekaadikaar (एकाधिकार)



Morphological Analyser Test

बिचारोन्मुख Check

बिचार
+
उन्मुख

Found!

1. + बिचारोन्मुख
2. ब् + इचारोन्मुख
3. बि + चारोन्मुख
4. बिच् + आरोन्मुख
5. बिचा + रोन्मुख
6. बिचार् + ओन्मुख
7. बिचार् + अउन्मुख
8. बिचार + उन्मुख

Rules

Applied rule will be highlighted

अ + उ = ओ
इ + अ = य
इ + आ = य
इ = य
उ + अ = व
उ + आ = व
उ = व

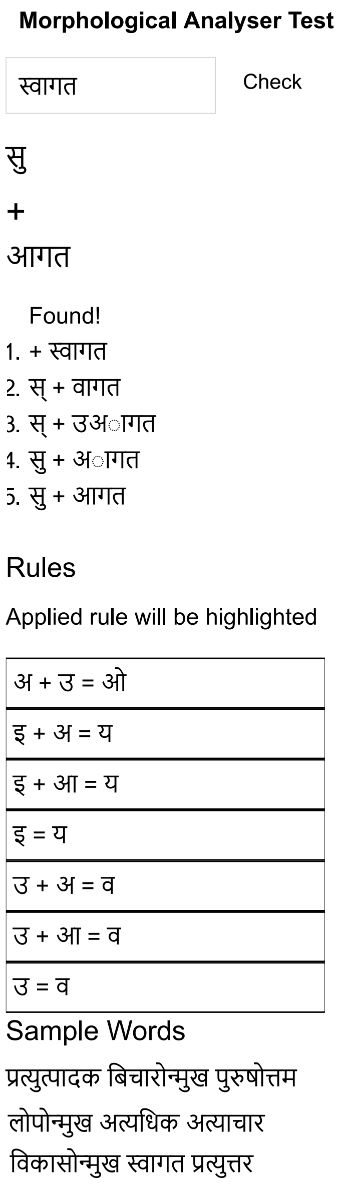
Sample Words

प्रत्युत्पादक बिचारोन्मुख पुरुषोत्तम

लोपोन्मुख अत्यधिक अत्याचार

विकासोन्मुख स्वागत प्रत्युत्तर

Figure 2: Output for the word बिचारोन्मुख (Bicharonmukh) from our web implementation of algorithm in Python 3.



Morphological Analyser Test

स्वागत Check

सु
+
आगत

Found!

1. + स्वागत
2. स् + वागत
3. स् + उअ◌ागत
4. सु + अ◌ागत
5. सु + आगत

Rules

Applied rule will be highlighted

अ + उ = ओ
इ + अ = य
इ + आ = य
इ = य
उ + अ = व
उ + आ = व
उ = व

Sample Words

प्रत्युत्पादक बिचारोन्मुख पुरुषोत्तम

लोपोन्मुख अत्यधिक अत्याचार

विकासोन्मुख स्वागत प्रत्युत्तर

Figure 3: Output for the word: स्वागत (Svaagat) from our web implementation of algorithm in Python 3.

Table 3: Vowel and Vowel Modifiers Conversion Table

RHS ←−− ························ −−→LHS	
Vowel Sound	Vowel
<Combines with half consonant and removes half consonant>	अ (a)
ा	आ (aa)
ि	इ (i)
ी	ई (ee)
ु	उ (u)
ू	ऊ (oo)
े	ए (e)
ै	ऐ (ai)
ो	ओ (o)
ौ	औ (ou)

दिग्दर्शन (Digdarshan) = दिक् (Dik) + दर्शन (Darshan)

In this case, a consonant sound `क्' (k) merges with `द' (da) to form `ग्' (ga). This finding has opened up possibilities for further improvement to this rule-based analyzer.

4. Conclusion

An algorithm to improve the performance of existing Nepali Morphological Analyzers by borrowing Sanskrit Conjunction (sandhi) rules were successfully created and implemented. This algorithm can be integrated into existing MAs to break Sanskrit words in the Nepali lexicon. These cases of breaking words based on vowel combinations were not handled previously by any Nepali language stemmers and this is an addition to these MA systems with acceptable accuracy. This article described the portion that can be integrated into existing MAs to improve their performance. A feasible position where this algorithm can be integrated with the flow of existing MAs can be a subject of another research and was not discussed in this paper, leaving open options for the respective authors and developers, where to integrate this unit of the algorithm in their MAs.

Limitation and further work

Techniques used by existing stemmers and morphological analyzers doesn't work for many pure Nepali words. An example of such words is:

Lavaai(लवाइ) = Laa (ला){root}+ Aai (आइ) {suffix}

This pattern cannot be solved even by the rules followed by Tatsama words and need a detailed study of its own and is out of scope of this paper.

References

[1] Kulkarni M, Dangarikar C, Kulkarni I, Nanda, A & Bhattacharyya P, Introducing Sanskrit wordnet, In Proceedings on the 5th Global Wordnet Conference (GWC 2010), Mumbai, January 31 - February 4, 2010.

[2] Panini, The Ashṭādhyāyī of Pāṇini, Edited/translated by Srisa Chandra Vasu, 2 Vols, Motilal Banarsidass, Delhi, India. (1962) 2.

[3] Bal B K & Shrestha P, A Morphological Analyzer and a stemmer for Nepali. PAN Localization, Working Papers, 324-31 (2007).

[4] Sitaula C, A hybrid algorithm for stemming of Nepali text. Intelligent Information Management, 5(04),136 (2013).

[5] Paul A, Dey A & Purkayastha, An Affix Removal Stemmer for Natural Language Text in Nepali, International Journal of Computer Applications, 91(6) (2014).

[6] Shrestha I & Dhakal S S, A new stemmer for Nepali language. International Conference on Advances in Computing, Communication, & Automation (ICACCA), Fall, IEEE, Bareilly, India, September 30 - October 1, 2016.

[7] Deshmukh R, Bhojane V, Sandhi Splitting Techniques For Different Indian Languages, International Journal of Engineering Technology, Management and Applied Sciences (ijetmas), 2(7) (2014).

[8] Deshmukh R & Bhojane V, Building Vowel Sandhi Viccheda System for Sanskrit. International Journal of Innovations & Advancement in Computer Science IJIACS, 4, 12 (2015).

[9] Joshi Shripad S, Sandhi splitting of Marathi compound words. International. Journal on Advanced Computer Theory and Engg, 2(2) (2012).

[10] Kuncham P, Nelakuditi K, Nallani S & Mamidi R, Statistical sandhi splitter for agglutinative languages.In International Conference on Intelligent Text Processing and Computational Linguistics, Cairo, Egypt,14-20 April, 2015.

[11] Baral, T D, तत्सम नेपाली व्युत्पत्ति शब्दकोश (Tatsama Nepali Vyutpatti Shabdakosh). Vidharthi Pustak Bhandar, Kathmandu, Nepal (2011).