Sinhala Word Joiner

Rajith Priyanga
Department of Computer Science and Engineering
University of Moratuwa
Katubedda, Sri Lanka
rpriyanga@yahoo.com

Surangika Ranatunga
Computer Science Department
University of Moratuwa
Katubedda, Sri Lanka
surangika@cse.mrt.ac.lk

Abstract

Word and morpheme joining is a basic operation in Sinhala morphology. When building the Sinhala morphology tool stack, a Sinhala word joiner is a core requirement. Therefore a research is done to study the Sinhala word join rules and implemented a reusable library and a standalone application to cater the requirement. It uses the documented join rules from the grammar books in the original form without any modification, so that the linguistic scholars can also benefit from the outcome. This software can join two Sinhala words or morphemes and return the valid combined forms and the join rule names used for the joining. The word join tool uses a combination of a corpus and hand coded rules to eliminate the false positives.

Keywords: Sinhala word joiner, Morphophonemic tools, corpus based scoring algorithm

1 Introduction

Sinhala belongs to the Indo Aryan sub branch of Indo European language family. Sinhala is a descendent of the Sanskrit language. It is largely influenced by the Pāli language from second century B.C. as a result of the introduction of Buddhism to Sri Lanka. Other than from Pāli, Sinhala has influences mainly from Tamil, Arabic, Portuguese, Dutch and English languages. Sinhala is written in its own script which is a descendent of the Brahmi script.

Even though the Sinhala language and its script have a lot of similarities with their ancestors from India, they have evolved in their unique way over more than 2 millenniums in Sri Lanka which is a geographically separated island from India.

2. Sinhala Morphology

Similar to Sanskrit, Sinhala is also rich with inflectional and derivational morphology. In inflectional morphology, new grammatical form of the same word class as the lemma (base word) are formed by applying morphological operations on the lemma.

E.g.
minis + *u* → *minissu* (මිනිස් + ු → මිනිස්සු)

minis is the lemma form of the noun man.
u is the suffix to generate the plural subject form of the noun.

In derivational morphology, new words that can be from different word classes or with different meanings are formed by applying morphological operations on the lemma.

E.g.
duk + *pat* → *duppat* (දුක් + පත් → දුප්පත්)

duk means suffering. *pat* means become. The combined word *duppat* is an adjective that means poor.

Based on the two level morphology concept, these morphological operations can be represented in two stages.

1. Lexical Representation
2. Surface Representation

E.g.
1. Lexical representation of the plural subject form of the noun lemma *minis* is (*minis* + *u)* (මිනිස් + ු)

2. Surface representation of (*minis* + *u*) is *minissu* (මිනිස්සු)

For morphological synthesis, both the lexical rules and the surface representation rules should be applied. The surface representation rules represent the phonological aspects. They transform the morphemes and words into different forms. The transformation to the morphemes and words in different environments is called Morphophonemics.

In Sinhala language, the most common lexical rules are in the following forms. Their changes happen at the word boundaries.

prefix + word → word
lemma + suffix → word
word + word → word

Where + is the join operator.

Most of the inflectional morphology operations are in the lemma + suffix → word form.

e.g. *minis* + *u* → *minissu* (මිනිස් + ු → මිනිස්සු)

Most of the derivational morphology operations are in the "prefix + word → word" and "word + word → word" form.

e.g.
duk + *pat* → *duppat* (දුක් + පත් → දුප්පත්)
Meaning is explained above.

pol + *aṭṭa* → *pollaṭṭa* (පොල් + අත්ත → පොල්ලත්ත)

pol means coconut. *atta* means branch. *pollatta* means the branch of a coconut tree.

In Sinhala, these sub set of morphophonemic rules are called *sandhi* (join rules). Similar to the *Ashṭādhyāyī* of *Pāṇini* in Sanskrit, there is a Sinhala grammar book named *Sidat Saňgarā* written in 13th century A.D. by *Vēdēha Swāmi*. Even though it is written for poets, *Sidat Saňgarā* describes some of the grammatical aspects of the Sinhala language.

Unlike in Sanskrit, there are fewer (9) join rules in Sinhala language according to *Sidat Saňgarā.*

Following is an example of how the *Sidat Saňgarā* has explained join rules. This join rule is names as *Pūrwạ Swạrạ Lōpạ*.

"*pera sarạ lopā parạ sarạ gatạtạ pæminạ*"
(පෙර සර ලොපා පර සර ගතට පැමිණ)

Meaning: Vowel part of the last letter of the left word is replaced by the first vowel in the second word.

According to the above definition there are two conditions for this rule to be valid.

1. The last letter of the left word must be a combined letter that has a consonant and a vowel part
2. The first letter of the right word must be a vowel.

2.1 Sinhala Word Join Rules

We represent the joined rules described in Sidat Saňgarā in an easily understandable format as follows.
Where

- Ci = consonant (e.g. *k* - ක්)
- Vi = vowels (e.g. *a* - අ)
- Individual letters at the word boundary are written in square brackets. (e.g. [C1])

- Combined letters that have a consonant and a vowel in it is written in [Ci|Vi] form.
    - e.g.
        - *ka* = [*k* | *a*] (ක = ක් + ඇ)
- L and R are the remaining parts of the joining words or morphemes.

1. Pūrwạ Swạrạ Lōpạ

L[C1|V1] + [V2]R → L[C1|V2]R

2. Parạ Swạrạ Lōpạ

L[C1|V1] + [V2]R → L[C1+V2]R

3. Swạrạ

L[C1] + [V1]R → L[C1|V1]R

4. Swạrādeshạ

L[C1|*a*] + [*i*]R → L[C1|*e*]R
L[C1|*a*] + [*u*]R → L[C1|*o*]R
L[C1|*a*] + [*u*]R → L[C1|*ō*]R

5. Gatrādeshạ

L[C1|V1] + [C2|V2]R → L[C1|V1][C3|V2]R

Where C3 is a member of {*y, v, h, k, t, p, n, m*}

6. Pūrwạ Rūpạ

L[C1] + [C2|V2]R → L[C1][C1|V2]R

7. Gatrākshạrạ Lōpạ

L[*n*] + [C2|V2]R → L[*ňg*|V2]R
L[*n*] + [C2|V2]R → L[*ňb*|V2]R

8. Āgạmạ

L[C1] + R → L[C1|u]R
L[C1] + R → L[C1|i]R

L[C1|V1] + [V2]R → L[C1|V1][C3|V2]R
Where C3 = {*y, v, r*}

9. Dwitwạ Rūpạ

L[C1|V1] + [V2]R → L[C1][C1|V2]R

Apart from the above 9 join rules, there are few more join rules that are available in current day Sinhala. Some of them are directly taken from Sanskrit to be used in loanwords. Some of them are commonly used. Following join rule is an example for a rule that is not available in *Sidat Saňgarā*, but currently in use.

- Parạ Rūpạ

L[C1] + [C2|V2]R → L[C2][C2|V2]R

3 Objective

The objective of this research is to implement a reusable word joining tool to be used in the Sinhala language tools stack that is being developed.

The functionality of the target tool can be summarized by a function *f* that has inputs and outputs as follows:

(combined, rule) = *f*(left, right)

Where,

1. left and right are valid Sinhala words or morphemes.
2. combined is the valid joined form of the left and right. null is also a valid value.
3. rule is the name of the join rule used for the joining when the combined is not null.

4. Related Work and Challenges

4.1 Related Work

There have been few attempts to implement morphological synthesizers and analysers for Sinhala verbs and nouns. The basic operation of Sinhala word formation is joining words with morphemes or other words. Without a tool to do the joining operation, it is difficult complete a practical morphological synthesizer for Sinhala. There is no such tool implemented so far.

Similarly the disjoin operation is required for the morphological analyser.

Therefore there is no successful morphological synthesizers or analyzers implemented for the Sinhala language yet.

4.2 Finite State Transducers

There have been attempts to implement word joiners for Indic languages such as Hindi and Sanskrit. All of them have used finite state transducers to do the morphophonemic operations to obtain the surface form. Most of the morphological tool implementations for European languages also use finite state transducers to obtain the surface representation for the lexical representation. Finite state transducers are considered to be the standard solution for this morphology problem.

4.3 Ambiguity

Due to the nature of the join rules, for a given pair of words or morphemes, there can be multiple matching join rules. Also there can be multiple possible combinations for a given pair even when a single join rule is applied.

Therefore it can be summarized that one of the following scenarios can be true for the combined forms and a given two pair of words or morphemes.

1. There is only one matching join rule for the pair
    a. The combined form/forms are valid
    b. The combined form/forms are invalid
2. There are multiple matching join rules and they yield the same combined form.
    a. The combined form/forms are valid
    b. The combined form/forms are invalid
3. There are multiple matching join rules and they yield the different combined forms.
    a. Some of the forms are valid and some of the forms are invalid
    b. All the forms are valid
    c. All the forms are invalid

In order to detect the correct join rule and the correct join forms accurately, two options were considered.

1. Using another set of rules, so that they can be applied on top of the base join rules and get rid of the false positives.

    E.g.
    When joining the lemma *nalu* (නළු) and the suffix *a* (ආ), the *Dvitvạ Rūpạ* join rule is also selected. It can generate the form *nalla*. (නළ්ළා)

    But there is an elimination rule that says The letter *lu* (ළු) is not duplicated when joining. Therefore the form *nalla* is ignored.

2. Check against the universal set of valid Sinhala words, so that you can get rid of the combined words that are not members of that set of valid Sinhala word.

An attempt is made to collect the language rules that can be used to detect the correct join operation for a given pair. But it is learnt that the documented secondary rule set is not complete, so that in some scenarios, access to the Sinhala vocabulary is needed to check the validity of the combined forms.

Also an attempt is made to learn these extra rules from a sample data set with tuples of left, right and combined forms. Sinhala being a low resource language, it is difficult to collect an accurately enriched data set large enough to perform the learning, in order to learn the complete rule set.

Having access to a database of all valid Sinhala words is also not practical. Also there are some valid words generated as combined forms by the join rules, that are not valid combined forms of the given 2 words or morphemes.

Hence a combined solution is proposed. It involves finite state transducers for each join rule and non-tagged corpus of Sinhala words.

5 Methodology

In this research, we implemented a generic Sinhala word joining tool based on the above mentioned 9 base rules. We added 4 more join rules that are currently in use even though they have not been mentioned in *Sidat Saňgarā*.

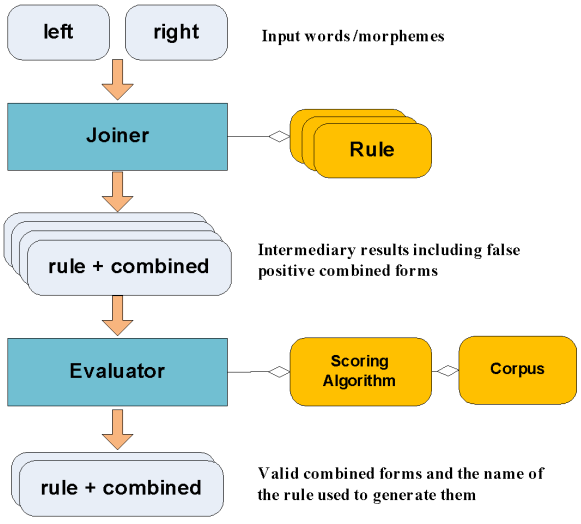
Due to the ambiguous nature of the Sinhala join rules and due to the other exceptions available in the Sinhala language, finite state transducers in isolation cannot solve the problem with the required accuracy.

Thus, a compiler customized for the join rules in the above mentioned format is developed. All the join rules are implemented as finite state transducers. In some European language solutions, multiple finite state transducers are cascaded to obtain the final results.

In Sinhala language studies, when two words or morphemes are joined together, one should be able to specify the join rule used to join them. Therefore we included that as a requirement to the problem statement. As a result of that the finite state transducers are not cascaded in our solution. Instead they were applied individually on the original word pair.

A set of large word-frequency pairs is cleansed and stored in a database to be used as a corpus.

Figure 1 shows the



For a given left right word/morpheme pair, the joiner applies all the join rules. Some finite state transducers arrive at end states. Therefore they yield combined forms. The finite state transducers that do not arrive at an end state do not yield a combined form. All the pairs of combined forms and the join rules used to generate them are returned as intermediary results. There can be both false positives and true positives among them.

A concept called Scoring algorithm is introduced to evaluate all the combined forms generated by the join rules. This scoring algorithm uses a corpus and some static elimination rules to assign a score to a given quadruple - left, right, combined form and rule.

The evaluator then selects the results with a score larger than a preconfigured threshold. The best value for the threshold with respect to a given scoring algorithm is obtained by regressing the joining operation with different threshold values for a sample data set that has a manually verified results set. For the current scoring algorithm, the threshold is set as 2.

This value has an impact from the quality of the corpus. If this set to 0, the number of false positives increases due to the impurities in the corpus. If this is set to a larger value, the number of false negatives increases since the evaluator tends to reject the valid combined forms that have low occurrence frequencies in the corpus.

5.1 Scoring Algorithm

Scoring algorithm is a concept introduced to eliminate the false positives.

```
interface JoinResultsScoringAlgorithm
{
        public long evaluate(
                String left,
                String right,
                String combined,
                JoinRule rule);
}
```

The software application developed as a part of this research has an implementation of a

JoinResultsScoringAlgorithm. It uses a corpus and some hand coded elimination rules.

It first checks whether the combined form is an invalid joined form of the left and right words or morphemes according to the elimination rules. If it is invalid, the score is set as -1.

If the combined form is not invalid, the word is looked up in the corpus. The occurrence frequency of the word is set as the score. It is a non-negative number.

New JoinResultsScoringAlgorithm instances can be plugged into the application to obtain better results.

6 Results

The precision and recall is measured for the joining results of the following different data sets.

6.1 Dataset 1

8 different grammatical forms of 50 Sinhala nouns are generated by joining their lemma and relevant suffixes. 412 noun forms are expected for the 400 word-morpheme pairs.

6.2 Dataset 2

50 pairs of complete words are joined to generate combined words.

6.3 Precision and Recall

True positives are the correct combined forms for a given pair generated by the application as the end results.

False positives are the incorrect combined forms for a given pair generated by the application as the end results.

False negatives are the expected combined forms for a given pair, but not given by the application as end results. Some of them were eliminated at the scoring algorithm.

	True positives	False positives	False negatives
Dataset 1	401	22	9
Dataset 2	46	0	4
Total	447	22	13

Precision = True positives / (True positives + False positives)
= 447/(447+22)
= 0.9531

Recall = True positives / (True positives + False negatives)
= 447/(447+13)
= 0.9717

7 Conclusion

7.1 False Positives

The analysis showed that the main reason for the false positives is the lack of elimination rules.

E.g.
The input : *ali* + *ā* (අලි + ා)
ali is the lemma of the noun elephant
ā is the suffix to form the singular subject form

Expected output : (*aliyā* - අලියා, *āgamạ*)

Actual outputs : (*aliyā* - අලියා, *āgamạ*), (*allā* අල්ලා, *dwithwạ rūpạ*)

aliyā means elephant. *allā* means the god Allah.

The word *allā* is not a valid combined form of the lemma *ali* and the suffix *ā*.

Some of the false positives are due to the impurities available in the corpus.

It is suggested to add an exceptions database and introduce more elimination rules to the scoring algorithm to reduce the false positives.

7.2 False Negatives

The analysis showed that the main reason for the false negatives is the incompleteness of the corpus. The occurrence frequencies of the valid combined forms that are not available in the corpus are set as 0. Therefore they are eliminated at the evaluator.

It is not possible to achieve a corpus with all the Sinhala words. Therefore it is required to use

statistical methods to learn further rules that can be used in scoring algorithms.

Optionally they can be joined with the base join rules to modify the finite state transducers. But it is preferred to keep the base join rules in the pure form so that the output of the tool is in line with the Sinhala language theories.

7.3 Performance

Since the corpus we have used contains 1.2 million entries (including impurities) the database lookup takes a considerable time on test machines. Therefore an average join operation for a given word pair takes around 20 milliseconds on a 2.5 GHz processor.

7.4 Future Enhancements

A possible future enhancement would be to generate a large sample dataset of tuples of left and right words or morphemes, combined forms and rule name using the current word joiner tool version and use that dataset to mine the rules using statistical methods. It is required to get human input to verify the accuracy of the generated dataset.

The rule set mined by this exercise can handle words that are not available in the corpus too.

Reference

1. Jha G.N. et al. (2009) Inflectional Morphol-ogy Analyzer for Sanskrit. In: Huet G., Kul-karni A., Scharf P. (eds) Sanskrit Computa-tional Linguistics. Lecture Notes in Computer Science, vol 5402. Springer, Berlin, Hei-delberg
2. Ido Dagan, Lillian Lee, Fernando Pereira. Similarity-based methods for word sense disambiguation. Proceedings of the 35th an-nual meeting on Association for Computa-tional Linguistics – 1997
3. W.S.Karunathilaka. (2013). Sinhala Bhasha Vyakaranaya. Colombo, Sri Lanka: M.D. Gunasena (pvt) Ltd.,.
4. Kimmo Koskenniemi. A general computa-tional model for word-form recognition and production. Proceedings of the 22nd annual meeting on Association for Computational Linguistics – 1984
5. Lauri Karttunen, Ronald M. Kaplan, Annie Zaenen. Two-level morphology with composition. Proceedings of the 14th conference on Computational linguistics - 1992
6. M.F. Porter. An algorithm for suffix stripping. Program 1980
7. Malcolm D. Hyman. From Pāṇinian Sandhi to Finite State Calculus. Lecture Notes in Computer Science 2009
8. Mehryar Mohri, Fernando Pereira, Michael Riley. Speech Recognition with Weighted Finite-State Transducers. Springer Handbook of Speech Processing 2008
9. Munidasa Kumarathunga (2000). Vyakarana Vivaranaya. S. Godage. Colombo, Sri Lanka
10. N. Murali, R.J. Ramasreee, K.V.R.K. Acharyulu. Kridanta Analysis for Sanskrit. International Journal on Natural Language Computing 2014
11. Utpal Sharma, Jugal Kalita, Rajib Das. Un-supervised learning of morphology for build-ing lexicon for a highly inflectional language. Proceedings of the ACL-02 workshop on Morphological and phonological learning – 2002
12. Mittal, V. 2010. Automatic Sanskrit Segmen-tizer using Finite State Transducers. In: Pro-ceeding of Association for Computational Linguistics - Student Research Workshop.
13. Wilhelm Geiger (1938). A Grammar of the Sinhalese Language. Ceylon Branch of the Royal Asiatic Society (Colombo). Colombo.
14. Department of Census and Statistics Sri Lanka. (2012). Census of Population and Housing. Retrieved from http://www.statistics.gov.lk/PopHouSat/CPH2011/Pages/Activities/Reports/FinalReport/Population/FinalPopulation.pdf