SriShell Primo: A Predictive Sinhala Text Input System

Sandeva Goonetilleke † sandeva.goonetilleke@ist.osaka-u.ac.jp
Yoshihiko Hayashi ‡ hayashi@lang.osaka-u.ac.jp
Yuichi Itoh † itoh@ist.osaka-u.ac.jp
Fumio Kishino ‡ kishino@ist.osaka-u.ac.jp
†Graduate School of Information Science and Technology, Osaka University
‡Graduate School of Language and Culture, Osaka University
Yamada oka, Suita, Osaka, Japan.

Abstract

Sinhala, spoken in Sri Lanka as an official language, is one of the less privileged languages; still there are no established text input methods. As with many of the Asian languages, Sinhala also has a large set of characters, forcing us to develop an input method that involves a conversion process from a key sequence to a character/word. This paper proposes a novel word-based predictive text input system named *SriShell Primo*. This system allows the user to input a Sinhala word with a key sequence that highly matches his/her intuition from its pronunciation. A key to this scenario is a pre-compiled table that lists conceivable roman character sequences utilized by a wide range of users for representing a consonant, a consonant sign, and a vowel. By referring to this table, as the user enters a key, the system generates possible character strings as candidate Sinhala words. Thanks to a TRIE structured word dictionary and a fast search algorithm, the system successively and efficiently narrows down the candidates to possible Sinhala words. The experimental results show that the system greatly improves the user-friendliness compared to former character-based input systems while maintaining high efficiency.

1 Introduction

The mother tongue of 14.6 million (74% of the total Sri Lankan population of 19.7 million) Sri Lankans is Sinhala (U S Department Of State, 2007). While computing has become almost ubiquitous in the US and Europe, Sinhala is inadequately supported on computers. Sinhala is a less privileged language that does not have even an efficient and highly user-friendly text input system. This is a major bottleneck in handling Sinhala text on computers in order to develop any natural language processing tools. Even though various kinds of Sinhala fonts and input applications have been proposed, the language is still not well supported by computer systems. Hundreds of Sinhala fonts have been developed, but most of them have their own weaknesses. For example some rare Sinhala characters (such as ඔ ,ඕ) are missing in most of the fonts. Furthermore, the major problems of the current input systems are the lack of user-friendliness and efficiency.

The objective of this research is to propose an efficient and highly user-friendly predictive Sinhala input method, and to evaluate the efficiency and the user-friendliness compared with other input methods. Here, efficiency is quantified by the average typing cost per Sinhala character, and user-friendliness is quantified by ease of remembering. The average edit distance between a user-intuitive character sequence and the input sequences of each input method is taken as a measurement of the difficulty of remembering. Our results have proved that *SriShell Primo* has maximum user-friendliness while maintaining high efficiency.

The rest of the paper is organized as follows. In Section 2 we discuss various Sinhala input methods proposed up to now, and their main features. The main features of the proposed input method *SriShell*

Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages, pages 43–50,
Hyderabad, India, January 2008.

Primo are explained in Section 3. The evaluations are reported in Section 4. Section 5 concludes and outlines future work.

2 Character-based Input Systems

This section reviews the representative Sinhala input systems proposed so far.

These input methods are character-based, forcing the users to memorize key assignments for each and every Sinhala character. This is not an easy task because Sinhala has hundreds of combined characters.

2.1 Direct Input Method

Sinhala fonts assign vowel characters, consonant characters and vowel signs to the ASCII character code. For example, Sinhala අ (=a) was assigned to 0x61 (=ASCII 'a') in most of the fonts. In the direct input method, users have to input the character codes as assigned in a specific Sinhala font. A typical example of this kind of font is the "*kaputadot-com*" font.[1] Most of the online Sinhala sites including news sites use these kinds of fonts.

Sinhala Unicode characters can also be input directly by entering the hexadecimal code. The arrow (a) in Figure 1 shows an example of this method of input.

2.2 Conversion Systems

The direct input method assigns a key for each Sinhala character or a part of a character that may or may not be phonetically associated. For this reason, the key assignments are far from intuitive.

Natural SinGlish

To resolve this problem the *Natural SinGlish* (Natural Singlish, 2004) typing application was introduced by A. D. R. Sasanka. This application converts the input sequence that is more natural for users into character codes as shown in (b) of Figure 1. English spellings and the English pronunciations are the basis of this system. For example *shree la\nkaa* → ශ්‍රී ලංකා(=Sri Lanka). However, Sinhala has many more characters than English. To avoid ambiguity, this system has introduced several techniques, such as:

[1] `http://www.info.lk/slword/news.htm`

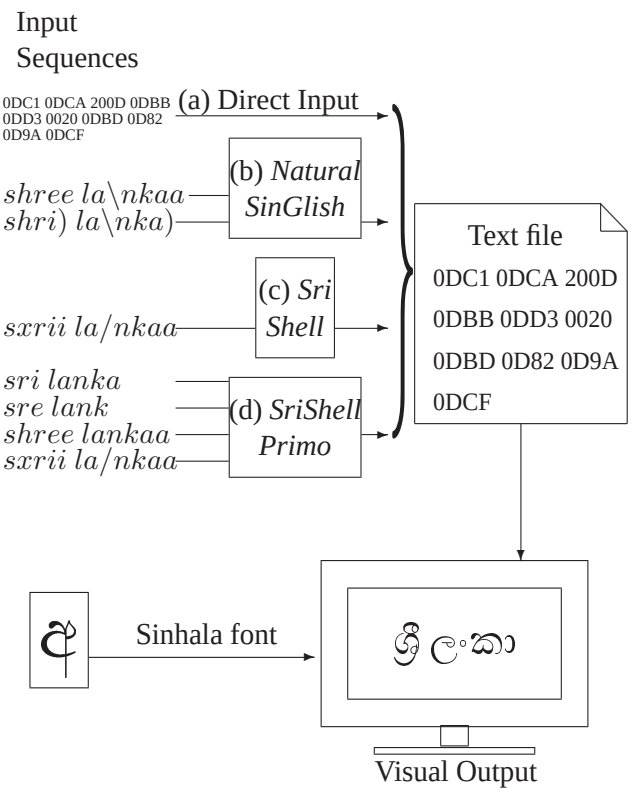

Figure 1: Sinhala character input systems (taking ශ්‍රී ලංකා (śri laṃkā : Sri Lanka) as an example)

- Capitals

a	→	අ(=a)	ta	→	ට(=ṭa)
A	→	ඇ(=æ)	Ta	→	ඨ(=ṭha)

- Key combinations

ea	→	ඒ(=ē)	KNa	→	ඤ(=ña)
oe	→	ඕ(=ō)	Sha	→	ෂ(=ṣa)

- Dead keys: "\" is used as a dead key

  \n → ං(=ŋ)

  \h → ඃ(=h)

This system is simply based on English spellings, making the system quite complex. The characters that have phonetic similarities cannot be typed in a similar manner.

ka	→	ක(=ka)	and	kha	→	ඛ(=kha)
ta	→	ට(=ṭa)	but	tha	↛	ඨ(=ṭha)

da	→	ඩ(=da)	and	nnda	→	ඬ(=ňḍa)
ba	→	බ(=ba)	but	nnba	↛	ඹ(=ňba)

This system is not very efficient in some cases because it uses a lot of upper case letters in the middle of the words, where the user needs to press and release the shift-key frequently.

Sri Shell

Goonetilleke et al. have proposed a Sinhala typing system called *Sri Shell* (Goonetilleke et al., 2007). *Sri Shell* assigns a key combination to each Sinhala character ((c) of Figure 1). The basis of this system is the phonetic notation of Sinhala characters.

Unlike the *Natural SinGlish*, *Sri Shell* has been implemented as an independent module, which allows the input of Sinhala text into any application program. Principles of the *Sri Shell* system are as follows.

- It is based on phonetic notation of the characters:
  - All aspirated consonants can be produced by adding an "h" to the unaspirated consonants.
  - Nasals can be produced by voiceless vowel preceded by "/".
  - Nasal+voiced can be produced by voiced vowel preceded by "/".
- It is consistent:
  - All long-vowels can be produced by doubling the last character of a short-vowel.
  - If two Sinhala characters map to the same roman character, then these Sinhala characters are differentiated by adding an "x." The "x" is added to the one that has a lower occurrence rate.
- It is complete:
  Most of the Sinhala input systems introduced up to now have several missing characters. Especially rare characters such as ඐ ,ඐෘ ,ඔ ,ඔෟ are missing in most systems. *Sri Shell* supports all the characters even though some of them cannot be displayed with most of the fonts.

2.3 Problems on Input Systems

Goonetilleke et al. have introduced *average edit distance* (per Sinhala character) as a measurement of user-friendliness. Even though they have succeeded in limiting the average edit distance to 0.35 keys per sinhala character, still the *Sri Shell* input sequence is quite far from users' natural intuition.

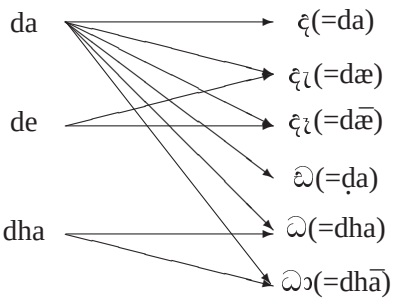

Figure 2: Some many-to-many relationships in test subjects' proposals

Our experiments have proven that users expect to get different Sinhala characters by typing the same key sequence. A few examples of these kinds of situations are shown in Figure 2.

Unfortunately, all the Sinhala input methods proposed up to now have a one-to-one (or many-to-one) relationship between the input sequence and output characters. For this reason users have to memorize how to type each Sinhala character.

To overcome this problem a many-to-many predictive character conversion algorithm is required.

3 Proposal: Word-based Input System

Here we propose a Sinhala input system called *SriShell Primo*. *SriShell Primo* is a word-based predictive converter. A number of predictive input methods have been proposed so far especially for handheld devices and mobile phones (MacKenzie et al., 2007). Among them, eZiText(R) [2] supports some Indic scripts such as Hindi, Tamil, Malayalam etc. The *SriShell Primo* users can input a Sinhala word by typing it in the roman character sequence they think is most appropriate. Even though the roman character sequence for a specific Sinhala word may differ from person to person, the *SriShell Primo* system is still capable of guessing the Sinhala word intended by the users. The user can select the intended word from the candidate list. A screen shot of the system is shown in Figure 3.

3.1 Main Features

SriShell Primo has three main features.

[2] `http://www.zicorp.com/eZiText.htm`

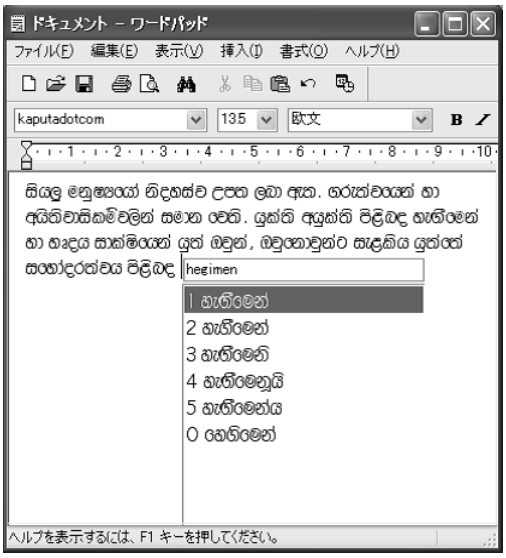

Figure 3: Screen Shot of *SriShell Primo*

1. Covers all possible input sequences

The roman character sequence used to represent each Sinhala word depends on the user. For example:

– *desei, dase, dese, daasee, desee, dasee, daesei, dasay, deesee, desee, dhasay, dhese* → දෑසේ (=dǣsē:in eyes)

On the other hand the input sequences can be ambiguous. For example:

– *bata* → හට(=bhata:soldier), බැට(bæta:hurt), බට(=bata:bamboo or pipe), බාටා(=bāta:a trade name)

The *SriShell Primo* is capable of converting all these possible sequences into the user-intended word.

2. Predicts possible words

SriShell Primo not only gives the Sinhala words that could be completely represented by the input roman character sequence, but the predicted Sinhala words are also added into the menu dynamically.

3. Allows word combinations

Normally Sinhala words are separated by a space, but we have found out in our preliminary experiments that sometimes some users omit the space, especially in the case of frequently co-occurring word pairs. *SriShell Primo* allows up to one space omission. Thus *SriShell Primo* gives word pairs also at the end of the menu, if the number of word candidates from the above methods is very small.

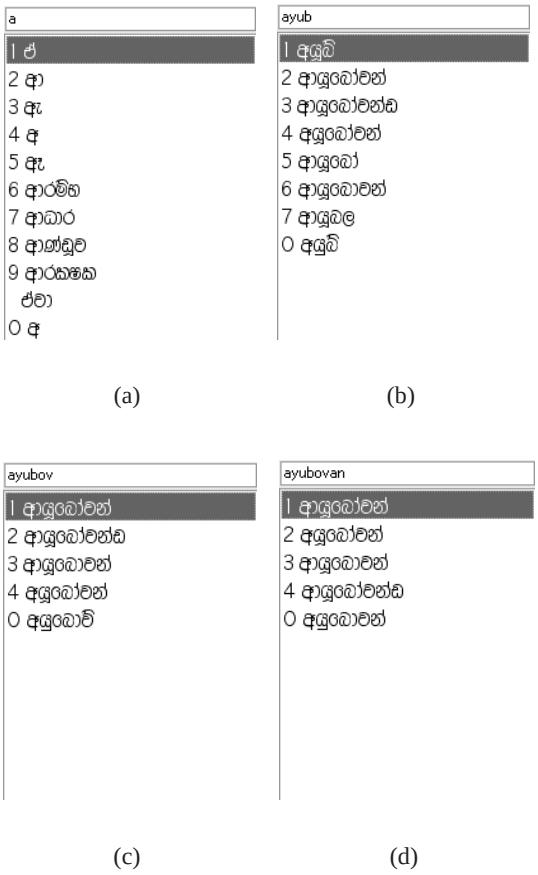

Figure 4: Text Entering Example ආයුබෝවන්(āyubōvan:Welcome)

Figure 4 demonstrates how the menu changes dynamically as user enters the keys, taking ආයුබෝවන් (āyubōvan:Welcome) as an example. When the user starts typing with "a" *SriShell Primo* gives a list of candidates in the menu that starts with ඇ,ආ,ඇැ,ඈ etc. as shown in Figure 4(a). When the user types up to "ayub" the intended word ආයුබෝවන් appears for the first time in the menu as the second choice (Figure 4(b)). Then ආයුබොවන් rises to the first choice of the menu when the user types up to "ayubov" (Figure 4(c)). A user can select the menu at this point by pressing the next required punctuation such as space, comma, period etc. or he/she can type up to "ayubovan" (Figure 4(d)).

3.2 The Algorithm

Input Sequences

Goonetilleke et al. have carried out an experiment to find out how the most frequent Sinhala characters are romanized by Sinhala speakers. We have further divided the roman character sequence for each Sinhala character into the consonant part, consonant sign part and vowel part. Thus we got a table that shows how each consonant, consonant sign or vowel is romanized by various users, as shown in Table 1.

Table 1: Input variation table

ඇ	(=ǣ)	←	aee,a,e,aa,ae,ee
ී	(=ī)	←	ii,i,ee,e,ie,y
ඳ	(=ň̆d)	←	/dx,nd,ndx,/d,d
ව	(=v)	←	v,w,vu,wu,u
ේ	(=ē)	←	ee,e,ei,ay
ශ	(=ś)	←	sx,z,sh,s
බ	(=b̆)	←	/b,b,mb
ඩ	(=ṇ̌d)	←	/d,nd,d
ඟ	(=ň̆g)	←	/g,ng,g
ඇ	(=æ)	←	ae,a,e
...			

Dictionary (TRIE structure)

We have used the Divaina online Sinhala newspaper[3] from January 2005 to May 2006 (about 50 MB of kaputadotcom font text) to create the dictionary. This dictionary contains about 240,000 words with their occurrence frequencies. To improve the search speed, the words are stored in a TRIE structure, where each branch of the TRIE structure represents a consonant part, vowel part or consonant sign part of a Sinhala character. Thus any single Sinhala character can be retrieved up to three hops. To reduce the amount of memory required, at the beginning this data structure is stored in the disk, and when the user starts to type words, the required part of the data structure is copied into the memory.

Procedure

When the user enters the text, *SriShell Primo* creates a list of all possible Sinhala character sequences that can be represented by the user's character sequence using the Input variation table. *SriShell Primo* travels along the TRIE structure in order to find out whether the Sinhala character sequences in the list are real words or not. As a result a candidate list is created and sorted in descending occurrence frequency order. For example in Figure 4(a) the candidates from 1 to 5 are created at this point.

Then *SriShell Primo* searches the Sinhala character sequence list to find out whether there is any sequence that matches the beginning of a Sinhala word. Those predicted words are also added at the end of the candidate list. The candidates from 6 onward in Figure 4(a) are added at this point.

If *SriShell Primo* was unable to find any candidates up to this point, it searches for word pairs that can be matched with the input character sequence, assuming that the user could have omitted a space in between.

Finally the *SriShell* (Goonetilleke et al., 2007) conversion of the character sequence is also added at the end of the candidate list, in order to allow typing a new word that is not included in the dictionary. The candidate number 0 in Figure 4(a) is added at this point. This candidate list is displayed as a menu, where the user can select the word that he/she intended by using a mouse or up/down arrow keys.

This process is repeated on each keystroke of the user. The user can enter the selected item to his/her document by striking the space key or any punctuation key.

4 Evaluation

This section describes the evaluation of the proposed input method. Following (Goonetilleke et al., 2007), we have also evaluated the proposed method in terms of efficiency and user friendliness.

4.1 Experiment

We have carried out an experiment to calculate the efficiency and user-friendliness of the proposed method. First, we allowed several minutes for the test subjects to practice *SriShell Primo*. Then they were asked to type a few paragraphs that contained 385 to 504 Sinhala characters from a general Sinhala newspaper. We informed them that they could type any Sinhala word by inputting any roman character sequence that they think best to represent the

[3] http://www.divaina.com/

specific Sinhala word. *SriShell Primo* keeps a log of typed keys, menu items selected, and time lapses in between. This experiment was carried out on a group of 6 subjects (2 female and 4 male, age 20-29 years).

4.2 Efficiency

The most general way to calculate efficiency is to experimentally compute the maximum typing speeds for each input method. Masui (Masui, 1998) has also used this measure to evaluate his character input method. However, the input sequences of the existing input methods are quite far from the average Sinhala computer users' intuition, and it is not easy to train people for typing Sinhala using those input methods, in order to carry out an experiment to measure their efficiencies. Hence, instead of the actual typing speed, Goonetilleke et al. have introduced *average typing cost per Sinhala character*, which represents the normalized typing speed, as a measure for efficiency. They have defined the average typing cost by Equation 1. There the weight of a normal key is set to 1, and w_{shift} and w_{repeat} are determined by applying the least square method as shown in Equations 4 and 5.

$$\begin{aligned} typing\_cost &= \frac{1}{\#\_Sinhala\_characters} \\ &\times (normal\_keys \\ &+ w_{shift} \times shifts \\ &+ w_{repeat} \times repeats) \end{aligned} \quad (1)$$

$$w_{shift} = \frac{t_{xY} + t_{Xy}}{t_{xy}} - 2 \quad (2)$$

$$w_{repeat} = \frac{t_{xx}}{t_{xy}} \quad (3)$$

where,

t_{xy} = average time lapse between two alpha key strokes

t_{xx} = average time lapse to repeat an alpha key stroke

t_{xY} = average time lapse between an alpha key and a shifted alpha key

t_{Xy} = average time lapse between a shifted alpha key and an alpha key

$$w_{repeat} = 0.87 - 0.73 t_{xy} (|r| = 85\%) \quad (4)$$

$$w_{shift} = 2.50 - 2.92 t_{xy} (|r| = 69\%) \quad (5)$$

Accordingly we define average typing cost per Sinhala character for *SriShell Primo* by adding the menu selecting time factor as shown in Equation 6.

$$\begin{aligned} typing\_cost &= \frac{1}{\#\_Sinhala\_characters} \\ &\times (normal\_keys \\ &+ w_{shift} \times shifts \\ &+ w_{repeat} \times repeats \\ &+ w_{select} \times selections) \end{aligned} \quad (6)$$

$$w_{select} = \frac{t_{sel}}{t_{xy}} \quad (7)$$

where,

t_{sel} = average time taken to select an item from the menu

Results

We have calculated the typing cost per Sinhala character from our experiment. The results are shown in Figure 5. The X-axis shows t_{xy}, the average time lapse between two alpha key strokes, while the Y-axis shows the average typing cost per Sinhala character. For comparison purposes we have plotted the best result obtained by Goonetilleke et al. as shown in Table 2.

Table 2: Average typing cost by Goonetilleke et al.

t_{xy}	best results	Input Method
200	2.18	*Sri Shell*
400	2.16	*Sri Shell*
600	1.99	*kaputadotcom*

When comparing existing input methods *SriShell Primo* has a very high degree of freedom in its input character sequences. *SriShell Primo* has a predicting function embedded where the users can reduce keystrokes per Sinhala character. This means the keystrokes per Sinhala character can be highly variable from person to person in *SriShell Primo*. Thus, unlike Goonetilleke's experiment results, we did not observe any correlation between the typing speed and the typing cost per Sinhala character. This implies that the efficiency of *SriShell Primo* is independent of users' typing speeds. However, we can

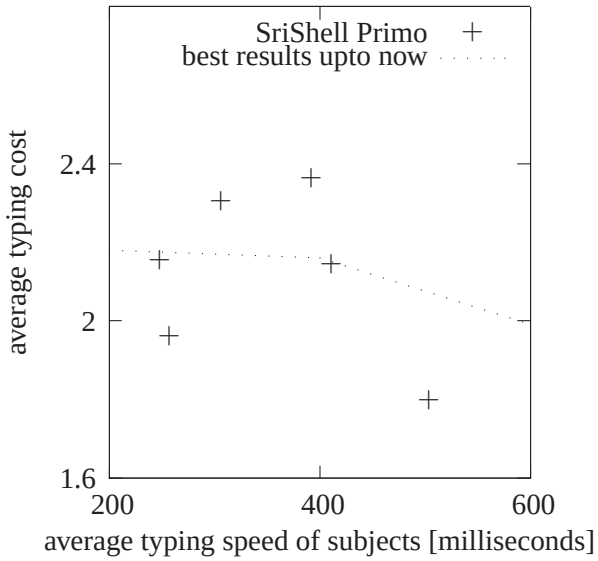

Figure 5: Average typing cost

say that the efficiency of *SriShell Primo* is not worse than *Sri Shell* and *kaputadotcom* because 4 out of 6 subjects who participated in our experiment were able to type Sinhala text more efficiently compared to the best efficiencies obtained by Goonetilleke's experiments.

4.3 User-friendliness

User-friendliness is strongly associated with how easy it is to remember the predefined input sequence for each Sinhala character. Goonetilleke et al. have taken the difference between the input character sequences of each input method and user intuitive character sequence as a measure of how difficult it is to remember the input sequence for each Sinhala character. They have measured the difference between the input key sequence of each input method and the proposed romanized sequence by several Sinhala speakers on several words by the edit distance between the two strings as shown in Equation 8.

$$avg\_edit\_dist = \frac{1}{\#\_Sinhala\_Chars} \times edit\_dist(user\_intuitive\_character\_sequence, input\_sequence\_of\_specific\_input\_method) \quad (8)$$

Table 3: Average edit distances

Input Method	Average edit distance
kaputadotcom	1.42
Sri Shell	0.44
Natural SinGlish	0.35
SriShell Primo	≤ 0.04

Edit Distance

The **Levenshtein distance** or **edit distance** between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character (Wagner et al., 1974).

The user-friendliness of *SriShell Primo* is completely dependent on the input variation table (Table 1). By adjusting this table it is possible to make *SriShell Primo* accept all user intuitive input sequences. As we have included all the conversions derived from Goonetilleke's experiment, we can expect a very high level of user-friendliness.

However, if there is any lack of user-friendliness in *SriShell Primo*, when the user tries to input a Sinhala word by entering the character sequence that he/she thinks most appropriate to represent a specific Sinhala word, he/she will not get that Sinhala word as a candidate in the *SriShell Primo* menu. At that point the user will have to correct the input character sequence in order to get the correct Sinhala word. As there may be other reasons for not having the user-intended Sinhala word in the menu due to mistypings etc., we can say the edit distance between the user intuitive input sequence and the input sequence of *SriShell Primo* is absolutely less than or equal to the edit_dist between input sequence with errors and input sequence without errors as shown in Equation 9.

$$edit\_dist(user\_intuitive\_input\_sequence, input\_sequence\_of\_SriShell\_Primo) \leq edit\_dist(input\_sequence\_with\_errors, input\_sequence\_without\_errors) \quad (9)$$

Results

As a measure of the user-friendliness, we have calculated the average edit distance per Sinhala character, which should be less than or equal to typing errors per Sinhala character. The results are shown in Table 3 with Goonetilleke's experiment results for comparison.

The results show that there is a big difference between the user intuitive character sequence and the input sequence proposed by *kaputadotcom*. Even though *Natural SinGlish* and *Sri Shell* were able to reduce this significantly, they were not good enough for a novice user because they require the user to memorize how to enter each Sinhala character. We can say that *SriShell Primo* was able to remove this barrier completely because anybody can enter Sinhala text correctly without acquiring any additional knowledge. Our experiment shows that the users average error rate is 4%, which means that the users were able to correctly type 96% of the Sinhala characters in the text, given the current input variation table.

At the same time *SriShell Primo* was able to keep the efficiency to an average of 2.1 key strokes per Sinhala character, and some users were able to reduce it to as few as 1.8 key strokes per Sinhala character. This reduction was achieved by the system's capability for predicting possible words while allowing shorter key sequences.

5 Conclusions and Future Work

This paper experimentally proved that the proposed predictive Sinhala input method has maximum user-friendliness, while maintaining high efficiency. This method can also be well applied to other languages with many characters but that lack well known 1-to-1 correspondences between the written characters and roman key sequences; these include Indic languages such as Sanskrit and Hindi.

Our future work has two main thrusts: to broaden the applicability and to improve the prediction.

We need to have a dictionary with better coverage to ensure better applicability. To do this, we will develop a systematic and automatic way to generate morpho-syntactically related derivational word forms, and store them efficiently in the dictionary. For example, our dictionary currently includes {ගස(=gasa : tree), ගස්(=gas : trees), ගසට(=gasaṭa : to tree), ගසේ(=gasē : in tree), ගසත්(=gasat : tree also), ගසුත්(=gasut : trees also), ගසෙන්(=gasen : from tree), ...} etc. However, we would like to generate these derivational forms from the root ගස(=gasa : tree).

On the other hand, to improve the accuracy of prediction, we will explore two dimensions: adaptation to an individual user and evaluation of linguistic contexts (Hasselgren et al., 2003). We see that the first dimension would enable a prompt improvement and will seek a means to adjust the candidate ordering in the input variation table by looking at a user's natural preferences in the inputs.

Acknowledgement

This research was supported in part by "Global COE (Centers of Excellence) Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

A. D. R. Sasanka 2004. *Natural Singlish*, `http://www.geocities.com/naturalsinglish/`.

Robert A. Wagner and Michael J. Fischer 1974. The String-to-String Correction Problem. *Journal of the ACM*, Volume 21(1), 168–173.

Toshiyuki Masui 1998. An efficient text input method for pen-based computers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 328 – 335.

U S Department Of State 2007. Background Note: Sri Lanka. `http://www.state.gov/r/pa/ei/bgn/5249.htm`.

Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, Fumio Kishino 2007. An Efficient and User-friendly Sinhala Input Method Based on Phonetic Transcription. *Journal of Natural Language Processing*, Volume 14, Number 5, 147 – 166.

I. Scott MacKenzie, Kumiko Tanaka-Ishii 2007. *Text Entry Systems: Mobility, Accessibility, Universality.* Morgan Kauffman, 344 pages.

Jon Hasselgren, Erik Montnemery, Pierre Nugues, Markus Svensson 2003. HMS: A Predictive Text Entry Method Using Bigrams. *Proceedings of the Workshop on Language Modeling for Text Entry Methods, 10th Conference of the European Chapter of the Association of Computational Linguistics* 43 – 49.