Design and Evaluation of User-friendly yet Efficient Sinhala Input Methods

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2009

Sandeva GOONETILLEKE

LIST OF PUBLICATIONS

I. Journal paper

[1] Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino: "*SriShell Primo*: A User-friendly yet Efficient Sinhala Text Input System," *Journal of Natural Language Processing*, (accepted for publication).

[2] Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino: "An Efficient and User-friendly Sinhala Input Method Based on Phonetic Transcription," *Journal of Natural Language Processing*, Vol. 14, No. 5, pp. 147–166, October 2007.

II. International conference proceedings

[1] Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino: "*SriShell Primo*: A Predictive Sinhala Text Input System," in *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pp. 43–50, Asian Federation of Natural Language Processing, Hyderabad, India, January 2008.

III. Local conference proceedings

[1] Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino: "An Efficient and User-friendly Sinhala Input Method Based on Phonetic Transcription," in *IPSJ SIG Technical Reports*, Vol. 2006, No. 124, pp. 101–106, November 2006.

[2] Ryoichi Watanabe, Sandeva Goonetilleke, Yuichi Itoh, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi: "Enhancement of Real-time Performance and Autonomic Movement of Cubes on Active-Cube," in *Technical report of IEICE. Multimedia and virtual environment*, Vol. 103, No. 745, pp. 1–6, March 2004.

[3] Sandeva Goonetilleke, Ryoichi Watanabe, Yuichi Itoh, Yoshifumi Kitamura, and Fumio Kishino: "A Study for Autonomic Movement of Cubes on ActiveCube," in *Proceedings of the IEICE General Conference*, A-16-38, pp. 347, March 2004.

Abstract

This thesis is a compilation of research results of the author, regarding Sinhala input systems, during his doctoral program in the Graduate School of Information Science and Technology of Osaka University, from 2006.

Sinhala, spoken in Sri Lanka as an official language, is one of the less privileged languages; still there are no established text input systems. Equipped with an adequate input system is crucially important in computing in Sinhala; here computing in Sinhala simply means to utilize computers with Sinhala language. Without such a device, ideas originated from Sinhala people cannot be fully verbalized, and hence will not be disseminated to the world. As with many of the Asian languages, Sinhala also has a large set of characters, forcing us to develop an input system that can properly address the issue.

The main objective of this research is to propose a highly user-friendly yet efficient Sinhala text input system. The targeted users of the system are the general Sinhala computer users, who have an average-level of knowledge about computers, and are familiar with Roman character keyboards. We have approached to this goal by implementing two systems: *Sri Shell*, a phonetically-principled system, and *SriShell Primo*, a word-based predictive system. To be user-friendly, *Sri Shell* is based on a phonetically-principled key assignments, while *SriShell Primo* is equipped with a mechanism that accepts user-intuitive key sequences.

Another objective of this research is to establish adequate measures for evaluating the user-friendliness and efficiency of Sinhala input systems, because we think the user-friendliness is quite important, given the targeted users. To this end, we propose an efficiency measure that quantifies the average typing cost per Sinhala character. We also propose a user-friendliness measure that evaluates the intuitiveness of required/acceptable key sequences. These measures are proven useful in evaluating existing Sinhala input systems as well as the proposed two systems.

This thesis consists of six chapters, and is organized as follows.

Chapter 1 gives a brief introduction on Sinhala language and summarizes the use of computers in Sinhala. Based on the argument, our research motivation is stated. Also the organization of this thesis is given.

Chapter 2 provides necessary background information to understand the presented research: linguistic nature of Sinhala language and classification of text input systems. This

chapter then summarizes the desiderata for realizing an effective Sinhala input system.

Chapter 3 proposes a new methodology to evaluate Sinhala input systems. First we discuss the general measures used to evaluate input systems. Text input systems should be evaluated not only by the efficiency, but also by the user-friendliness, especially when the users are not professionals. The efficiency is quantified by the average typing cost per Sinhala character, while the user-friendliness is assessed by the average edit distance between a user-intuitive character sequence and the input sequences of an input system. We report the evaluation results of existing Sinhala systems by employing these measures. We finally prove that the proposed user-friendly measure is valid to evaluate the user-friendliness through questionnaire based experiment.

In Chapter 4, we propose phonetically-principled Sinhala input system called *Sri Shell*. One of the strategies to ensure the user-friendliness is to develop a key assignment which is intuitive or principle-based. In this chapter, we propose a phonetically-principled associative conversion-based direct input system. The system is a light-weighted application independent module that can be realized without any language resources such as corpora or dictionaries. This chapter concludes that *Sri Shell* is moderately user-friendly while maintaining better level of efficiency comparing to other conversion-based direct input systems. It also should be noted that *Sri Shell* is a complete input system that can be utilized in combination with the next proposed system *SriShell Primo*.

In Chapter 5, we propose a word-based predictive Sinhala input system called *SriShell Primo*. The most prominent feature of this system is its high user-friendliness. A key to the user-friendliness is a pre-compiled *input variation table* that lists weighted correspondences between conceivable Roman character sequences and the associated Sinhala phonemes. This table is constructed to accept and adapt to the key sequences for a wide range of users. The introduction of this device however calls for the system to realize a mechanism to choose the best Sinhala character sequence toward the given user input sequence. We therefore propose a word-based predictive system to narrow down the ambiguities. This word-based system is also beneficial, as it can propose completion candidates during the input process. This chapter concludes that *SriShell Primo* has maximum user-friendliness while exhibiting a level of efficiency that is comparable to the most efficient direct input system.

Chapter 6, summarizes the results, and proposes research issues for improving the proposed systems, as well as more general research agenda for computing in Sinhala.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this section is to review the situation of computer uses in Sri Lanka, especially how general people use computers with Sinhala language, and to summarize issues which may prevent further expansion of the computer uses. Note that we use the term “Computing in Sinhala” to simply denote the notion of *utilization of computers with Sinhala language.*

In Sri Lanka the use of computers has begun to spread rapidly, due to the reduction in price and improvements in performance. However, people who do use Sinhala for their information interchange via computer are still very limited. Section 1.1 illustrates the current situation of computing in Sinhala, and argues that one of the major reasons in the limited computer use in Sinhala is lack of appropriate input system. Section 1.2 discusses the basic technical elements required for implementing an input system: Sinhala characters, their encoding and their rendering. Based on these discussions, we state our research motivation in Section 1.3. Lastly Section 1.4 describes the organization of this thesis.

1.1 Computing in Sinhala

The mother tongue of 74% of the total Sri Lankan population of 20.1 million, distributed all over Sri Lanka except the northern and the central areas, as shown in Figure 1.1, is Sinhala [1]. In Sri Lanka, there are three official languages, Sinhala, Tamil and English. Most of the governmental affairs in Sri Lanka are carried out in Sinhala. The education system also uses Sinhala up to the high school or university levels.

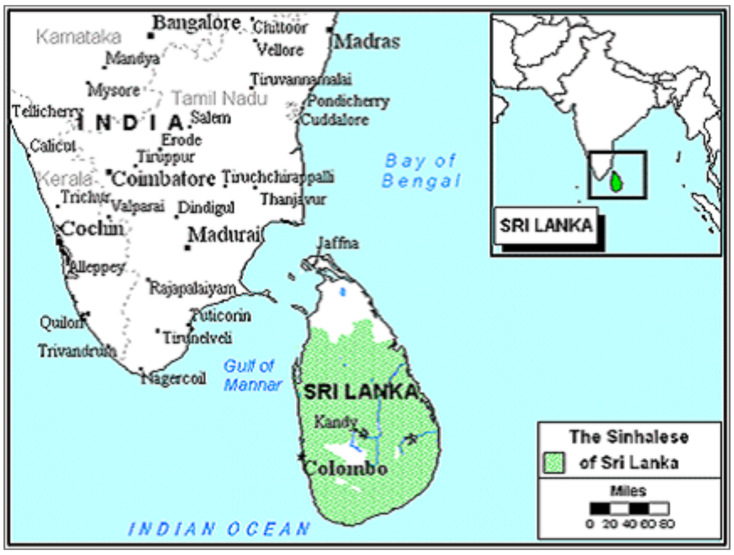


Figure 1.1: Distribution of Sinhala Native Speakers in Sri Lanka

According to the statistical data, along with the vast spread of computers in Sri Lanka, number of computer users in Sri Lanka has reached to an extent where one out of every 558 Internet users is a Sri Lankan [2]. However, the people who use Sinhala for their information interchange via computer are very limited. As one consequence of this situation, there are a very small number of Sinhala contents available on the web; where only one out of every 13,710 Wikipedia articles is in Sinhala [3].

There may be various reasons for such a limited use of computers in Sinhala. The most prominent reason for this is there are no effective Sinhala input systems. Without effective input systems, ideas originated from Sinhala people cannot be fully verbalized, and hence will not be disseminated to the world. Once substantial amount of Sinhala documents are created by using the input system, the idea or knowledge in them will be further utilized by employing linguistic tools such as OCR (optical character recognition), TTS (text to speech), and MT (machine translation). Actually some researchers have already proposed Sinhala optical character recognition tools [4, 5] and Sinhala text to speech tools [6]. Additionally, several machine translation systems also have been proposed such as: Japanese-Sinhala by Thelijjagoda et al. [7] and Sinhala-Tamil by Weerasinghe [8].

The next section discusses Sinhala characters, their encoding and their rendering; which are fundamental technical elements required for implementing a Sinhala input system.

1.2 Technical Elements of Sinhala Computing

This section discusses the basic technical elements required for implementing a Sinhala input system: identifying the complete set of Sinhala characters, encoding them, and rendering them. None of these is an easy task, because Sinhala has hundreds of *conjunct characters*. A *conjunct character* is a combination of several character components whose function is a phonetic modification. Therefore the definition of character also may differ from person to person. A detailed explanation on Sinhala writing system, and our definition of a character is given in Section 2.1.1.

Nonstandard fonts

Implementation of "*Nonstandard fonts*" is a widely used technique to encode and render Sinhala characters. During the past one or two decades, hundreds of nonstandard Sinhala fonts have been developed. *Kaputadotcom* explained in Section 2.3.2 is a typical example. For example, in this font "අ"(=a) is encoded into 0x61 (=ASCII 'a'). Therefore by pressing key: $\boxed{\text{A}}$ user can get "අ" on the screen. In this sense these fonts are not mere fonts, they themselves are input systems.

However they have their own weaknesses. The major problem is none of them are standard encoding schemes, where they use code points which overlap with code points of other encoding schemes such as ASCII or Japanese JIS code. As a result, in some cases Sinhala characters cannot be displayed together with foreign characters in the same document. The second problem is that some rare Sinhala characters (such as ඏ,ඐ) are missing in most of the fonts.

Unicode support for Sinhala [9] was expected to be a solution to these problems.

Unicode Support for Sinhala

Even though Unicode support for Sinhala is a standard scheme which includes all the basic characters and all diacritics, and assign them code points in an universal code space, it still suffers from the some rendering problems, where revisions are required. For example, in most of the operating systems, the default fonts for Sinhala incorrectly display "ක්‍රූ" (=krū)(=U+0D9A U+0DCA U+200D U+0DBB U+0DD6) as "ක්‍රූ."

Even though Unicode support for Sinhala has these kind of problems, it was able to provide a solid foundation for computing in Sinhala, and several input systems have been

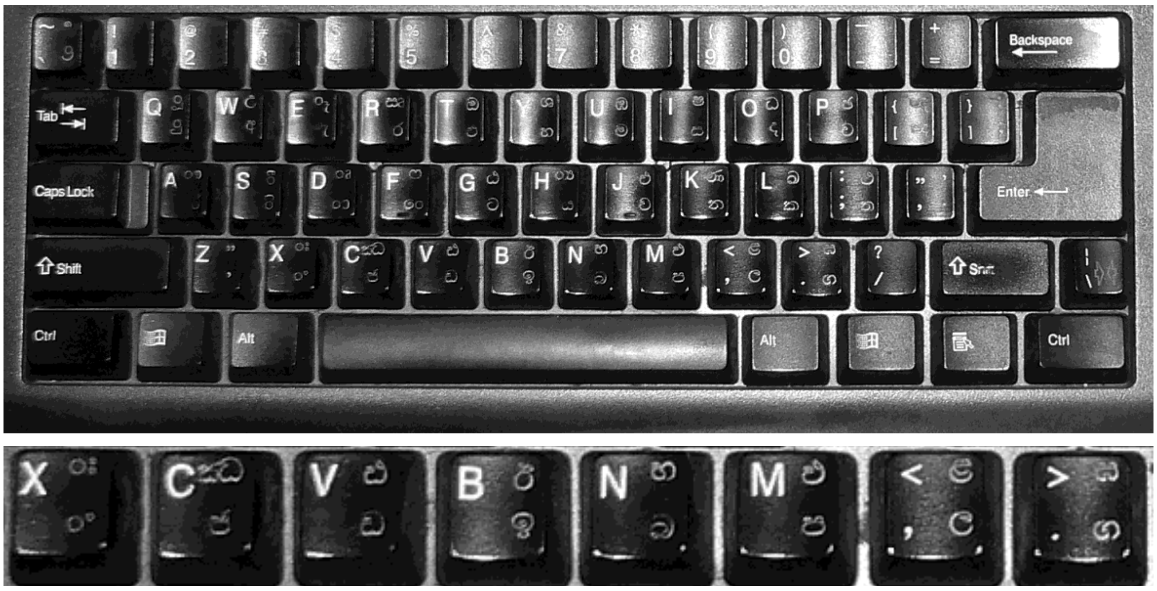


Figure 1.2: Sinhala Keyboard

proposed on this. For example some keyboard layouts have been introduced as shown in Figure 1.2.

Sinhala Input Systems

In the era of typewriters Sinhala typewriters are designed with an independent keyboard layout as shown in Figure 1.3. One of the most popular keyboard layouts was *Wijesekara.*

These layouts were very efficient to type Sinhala as far as the machineries are only used for the typing purpose. However, the situation is completely different if the input machinery carries more roles as with computers. Most of the operating systems used in Sri Lanka are English operating systems. In such a situation nobody can use a computer without practicing a Roman character keyboard layout, most probably a layout such as QWERTY or Dvorak [10, 11]. Those Sri Lankan computer users have to practice another keyboard layout in order to input Sinhala, which is not an easy task. Modern text input machineries for Sinhala therefore should be based on these keyboard layouts, as far as the target users are general users rather than professional typists.

Figure 1.3: Sinhala Typewriter

1.3 Research Motivation

Based on the previous discussions, the prime objective of this research is *to realize a Sinhala input system that is targeted to general Sinhala computer users.* To pursue this goal, we need to establish a technical architecture which is built upon careful considerations on innate characteristics of Sinhala language and preferences of the target users.

More specifically, we should pay considerable attention to: (1) Sinhala has a large set of syllabic characters and there are no standardized ways of transliteration, and (2) possible transliteration of Sinhala words can carry rather rich information that can narrow down possible word candidates. These innate characteristics of Sinhala may impact the design of a Sinhala input system which could be substantially different from Japanese *Kanakanji nyuryoku* input systems. With respect to the point (1), Japanese has rather standardized transliteration schemes [12] and far smaller set of characters, making the initial step of the input (*romaji nyuryoku*) more deterministic. On the other hand, with respect to the point (2), Japanese input system should utilize rich contextual information and/or user preferences to choose among possible Japanese ideograms (*Kanji*) candidates, which are less required in Sinhala. In summary, the technical solution to Sinhala input may substantially different from the one for Japanese input, and this provides us an

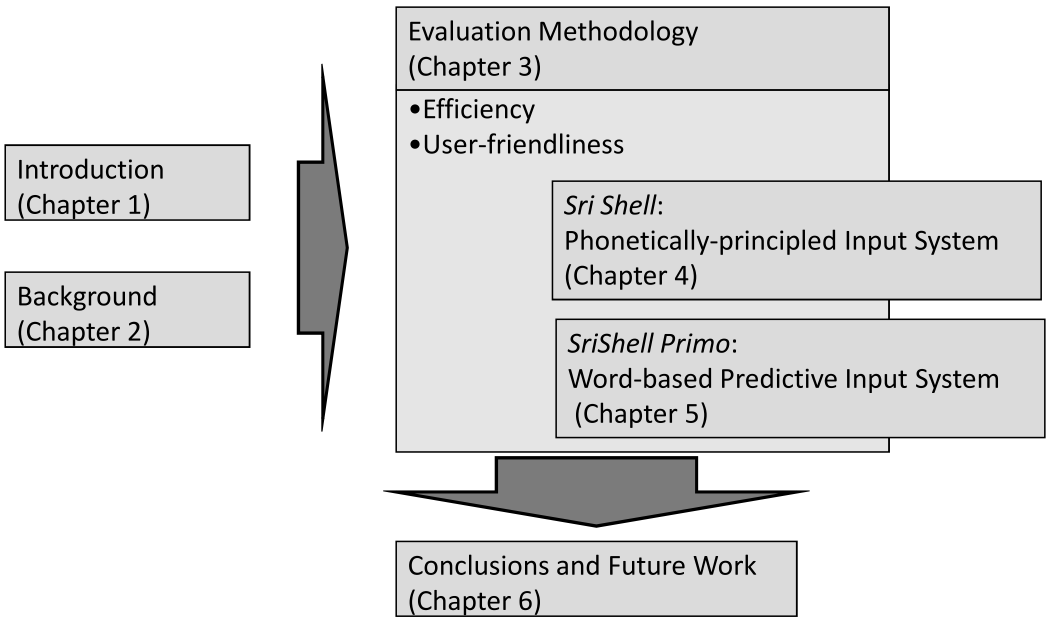


Figure 1.4: Organization of the Thesis

opportunity to develop a best-fit technology for Sinhala.

Once we come up with a technical solution to the above mentioned objective, we need to evaluate it and compare it with other competing solutions and existing technologies. However the evaluation/comparison measures should be carefully prepared by considering characteristics of Sinhala language as well as the nature of the targeted users. In this regard, our secondary objective of this research is *to establish a proper set of measures to evaluate Sinhala input systems*, especially by considering general Sinhala users who make use of Roman character keyboard.

1.4 Organization of this Thesis

This thesis has six chapters, and the overall organization is summarized in Figure 1.4. The descriptions below are quick summaries of the chapters.

Chapter 2 first introduces the basic characteristics of Sinhala language, as these are required to develop the succeeding discussions. Then it categorizes text input systems available for various languages, and reviews representative input systems available for Sinhala. Finally it summarizes the desiderata for realizing an effective Sinhala input system.

Chapter 3 discusses various measures used for evaluating input systems. Then it proposes new measures which are essential for evaluating the user-friendliness and efficiency of an input system. This chapter also evaluates the existing Sinhala input systems using those new measures. Finally it presents experimental evidences that validate the proposed measures.

One of the ways to improve the user-friendliness, is to provide a principled key assignment. Chapter 4 proposes a phonetically-principled Sinhala input system: *Sri Shell*, and evaluate its performances using the new measures proposed in Chapter 3.

In order to further improve the user-friendliness of the input system, Chapter 5 proposes a word based predictive input system: *SriShell Primo*, and evaluates how the user-friendliness has been improved while maintaining the efficiency.

Finally Chapter 6 concludes the achievement of this research, while discussing the future work.

Note that, Chapter 3 and 4 describe the results of the papers published in [13, 14]. Chapter 5 describes the results of the papers published in [15, 16].

Chapter 2

Background

This chapter provides necessary background information to understand the presented research: linguistic nature of Sinhala language and classification of text input systems.

2.1 Characteristics of Sinhala Language

This section explains the characteristics of Sinhala language on character level, and word level. The former is especially required to understand the discussions given in Chapter 4, and the latter is vital to understand the technical details given in Chapter 5.

2.1.1 Sinhala Characters

This section discusses the origin of the Sinhala writing system, Sinhala alphabets[1] and composition of compound Sinhala characters.

Origin of Sinhala Writing System

Brāhmī [17] script is the origin of Sinhala writing system. Table 2.1 shows the Brāhmī character set. As shown in Figure 2.1, Brāhmī has a number of descendant scripts such as Punjabi, Devanagari, Gujarati, Bengali, Oriya, Telugu, Kannada, Tamil, Malayalam, and many more. Sinhala is one of the descendants of the Brāhmī script, and is classified as *South Indic Scripts*. Although the Brāhmī script spread through India and Asia, the organizing principle remained intact. Each country/region however created its own set of

[1]Here alphabet means a character set used in a language.

Table 2.1: Brāhmī Characters

𑀅 a	𑀆 ā	𑀇 i	𑀈 ī	𑀉 u	𑀊 ū	𑀏 e		𑀐 ai	𑀑 o	𑀁 -m̥
𑀓 ka	𑀔 kha	𑀕 ga	𑀖 gha	𑀗 ŋa	𑀘 c	𑀙 cha	𑀚 ja	𑀛 jha	𑀜 ña	
𑀝 t̥a	𑀞 t̥ha	𑀟 d̥a	𑀠 d̥ha	𑀡 n̥a	𑀢 ta	𑀣 tha	𑀤 da	𑀥 dha	𑀦 na	
𑀧 pa	𑀨 pha	𑀩 ba	𑀪 bha	𑀫 ma						
𑀬 ya	𑀭 ra	𑀮 la	𑀴 l̥a	𑀯 va	𑀰 śa	𑀱 s̥a	𑀲 sa	𑀳 ha		

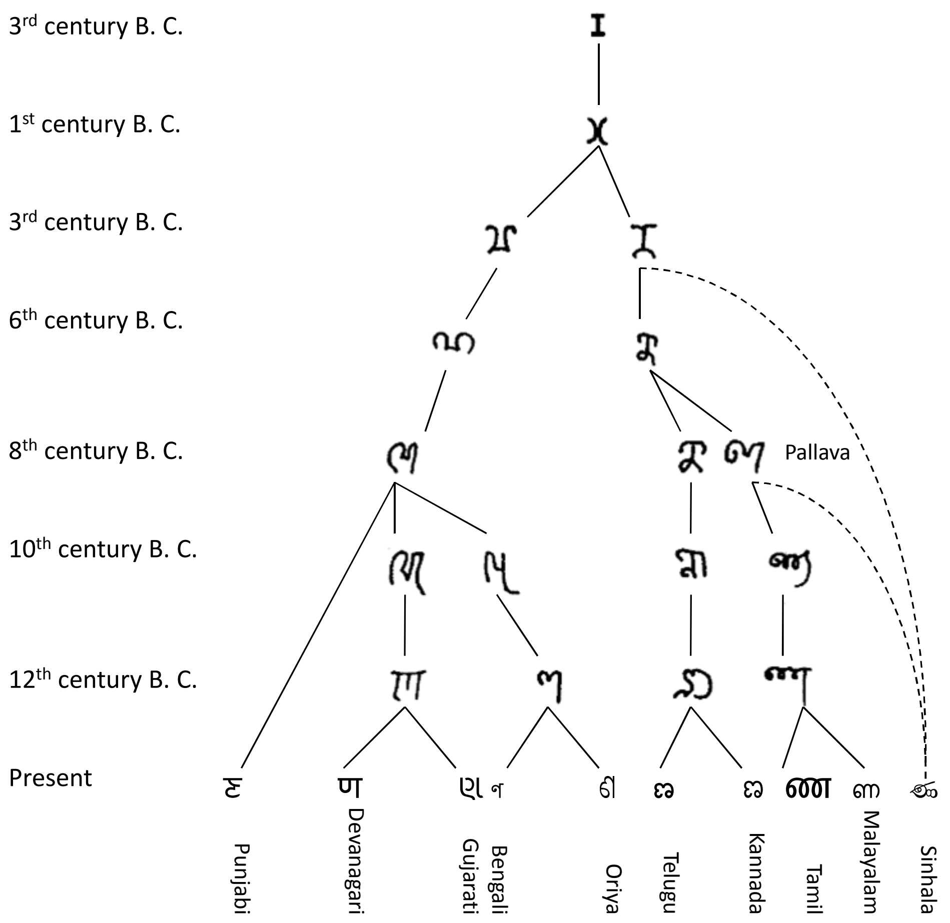


Figure 2.1: Descendants of Brāhmī (Taking ṇa Syllabic as an Example)

Table 2.2: Śuddha Siṃhala Hōḍiya (Pure Sinhala Alphabet)

Vowels	අ a	ආ ā	ඇ æ	ඈ ǣ	ඉ i	ඊ ī	උ u	ඌ ū	එ e	ඒ ē	ඔ o	ඕ ō
Consonants	ක ka	ග ga	ජ ja	ට ṭa	ඩ ḍa	ණ ṇa	ත ta	ද da	න na	ප pa	බ ba	ම ma
	ය ya	ර ra	ල la	ව va	ස sa	හ ha	ළ ḷa					
Diacritics	ං ṃ											
Nasals+ Voiced Consonants	ඟ ňga	ඦ ňja	ඬ ňḍa	ඳ ňda	ඹ m̌ba							

symbols depending on the material used for writing. In north India, where a reed pen was used for writing, the scripts have distinctive horizontal lines. While in south India, Sri Lanka, and Southeast Asia, where stylus was used to write on palm leaves, the script had to be more rounded [18]. So, different languages have mapped different symbols onto this inventory [19].

Sinhala Hōḍiya (Sinhala alphabet)

Hōḍiya is a list of characters that defines all the basic characters of Sinhala. It emerges into three variants according to the historical development.

The "Śuddha Siṃhala Hōḍiya" (pure Sinhala alphabet) has thirty-seven characters (twelve vowels, one diacritic and five nasals+consonants), as shown in Table 2.2. Most of the Sinhala words can be written using only these thirty-seven characters. After the thirteenth century [20] Sinhala language was very strongly influenced by Sanskrit and Pāli languages. As a result, many Sanskrit characters were incorporated into the Sinhala alphabet. The revised alphabet is called the "Miśra Siṃhala Hōḍiya" (Mixed Sinhala Alphabet). The "Miśra Siṃhala Hōḍiya" consists of fifty-nine characters (eighteen vowels and forty-one consonants), as shown in Table 2.3. The occurrence probability of these newly added twenty-two characters is lower than the original thirty-seven pure Sinhala characters. However, these new characters are frequently used in formal sentences. Thus

Table 2.3: Miśra Siṃhala Hōḍiya (Mixed Sinhala Alphabet)

Vowels	අ a	ආ ā	ඇ æ	ඈ ǣ	ඉ i	ඊ ī	උ u	ඌ ū	ඍ ṛ	ඎ ṝ
	ඏ ḷ	ඐ ḹ	එ e	ඒ ē	ඓ ai	ඔ o	ඕ ō	ඖ au		
Consonants	ක ka	ඛ kha	ග ga	ඝ gha	ඞ ṅa					
	ච ca	ඡ cha	ජ ja	ඣ jha	ඤ ña					
	ට ṭa	ඨ ṭha	ඩ ḍa	ඪ ḍha	ණ ṇa					
	ත ta	ථ tha	ද da	ධ dha	න na					
	ප pa	ඵ pha	බ ba	භ bha	ම ma					
	ය ya	ර ra	ල la	ව va	ශ śa	ෂ ṣa	ස sa	හ ha	ළ ḷa	
Diacritics	ං ṃ	ඃ ḥ								
Nasals+ Voiced Consonants	ඟ n̆ga	ඦ n̆ja	ඬ n̆ḍa	ඳ n̆da	ඹ m̆ba					

they are also an indispensable part of the Sinhala alphabet. In the nineteenth and twentieth centuries, Sinhala language was strongly influenced by Portuguese, Dutch and English languages. Consequently the modern Sinhala alphabet also includes the 'f' sound. The modern "Sammata Siṃhala Hōḍiya" (standard Sinhala alphabet) consists of eighteen vowels and forty-two consonants (altogether sixty characters), as shown in Table 2.4.

Conjunct Characters

The basic characters (the characters listed in Siṃhala Hōḍiya) are modified to produce hundreds of conjunct characters that are also known as *grapheme clusters* [21], by adding

Table 2.4: Sammata Siṃhala Hōḍiya (Standard Sinhala Alphabet)

Vowels	අ a	ආ ā	ඇ æ	ඈ ǣ	ඉ i	ඊ ī	උ u	ඌ ū	ඍ ṛ	ඎ ṝ
	ඏ ḷ	ඐ ḹ	එ e	ඒ ē	ඓ ai	ඔ o	ඕ ō	ඖ au		
Diacritics	ං ṃ	ඃ ḥ								
Consonants	ක ka	ඛ kha	ග ga	ඝ gha	ඞ ṅa	ඟ n̆ga				
	ච ca	ඡ cha	ජ ja	ඣ jha	ඤ ña	ඦ n̆ja				
	ට ṭa	ඨ ṭha	ඩ ḍa	ඪ ḍha	ණ ṇa	ඬ n̆ḍa				
	ත ta	ථ tha	ද da	ධ dha	න na	ඳ n̆da				
	ප pa	ඵ pha	බ ba	භ bha	ම ma	ඹ m̆ba				
	ය ya	ර ra	ල la	ව va	ශ śa	ෂ ṣa	ස sa	හ ha	ළ ḷa	ෆ fa

various components (such as *vowel signs*, *devowelizers* and *consonant signs*). Because of this, the definition of a “character” may vary from person to person. As discussed later, to evaluate the “user-friendliness” and “efficiency” of an input system, it is vital to know the occurrence probability of each character. To this end, we need to define a “Sinhala character.” Before giving the definition, we will discuss how the conjunct characters are created. We use Mikami’s notation [22] to explain the structure of Sinhala characters. According to Mikami, Sinhala script is classified as *combining syllabics*, which is a subset of the *syllabary.* Sinhala script is then further categorized as “a-Vowel Inherent Combining Syllabics” [22]. A number of relevant concepts are described as follows.

Basic characters in Sinhala can be classified into three classes.

1. **Vowel syllabics** The first eighteen characters (අ(a) to ඖ(au)) shown in Table 2.4

are vowel syllabics. The shapes of these characters never change. Thus these vowel syllabics are *atomic characters* [21]. Mikami uses the symbol V for this kind of character, and the pronunciation is represented by v.

2. Diacritics There are two diacritics in Sinhala, which are the *anusvaraya* (ං=ṃ) and the *visargaya* (ඃ=ḥ). These two characters can appear after any other vowel syllabic or a consonant syllabic. Mikami uses the symbol $\underline{D}$ for them.

3. Consonant syllabics The Sammata Sinhala Hōḍiya has forty-two consonant syllabics as shown in the Table 2.4 Consonant section. All these consonant syllabics include the vowel sound අ(=a) which is called the *inherent vowel.* Mikami uses C to represent these consonant syllabics and the pronunciation is denoted by cv_0, where v_0=“a.”

Sinhala grapheme clusters can have the following constructions. A grapheme cluster is described as “what end users usually think of as characters” [23].

Consonant-vowel combinations *Vowel signs* are used to change the inherent vowel අ(=a) of a consonant syllabics into another vowel. Mikami uses $\underline{V}$ to represent vowel signs, and the consonant-vowel combining characters are represented by $C\underline{V}$. These vowel signs are called *pilla*(පිල්ල) or *pili*(පිලි) in Sinhala. Table 2.5 shows a few examples of consonant-vowel combinations. The first line (with <*null*> vowel sign) indicates a-Vowel inherited consonant syllabics, which were also listed in Table 2.5. Most of the vowel signs do not take different shapes corresponding to the consonant except the vowel sign for u (*pāpilla*), which takes various shapes depending on the consonant.

Removing the inherent vowel In Sinhala pure consonants are also used in Sinhala scripts, not only at the end of a word but also in the middle of a word and at the beginning of a word. There are four ways to remove the inherent vowel අ(=a).

- **Devowelizer** A devowelizer is added to consonant syllabics in order to remove the inherent vowel sound. This is the most general way to remove the inherent vowel, but it has a lower priority compared to other specific inherent vowel removers. In Sinhala this devowelizer is called the *hal-lakuṇa.* There are two shapes for hal-lakuṇa and one of them is selected depending on the shape of

Table 2.5: Examples of Consonant Vowel Combinations

v	Vowel Signs	V	ṭ		p		k		ḷ	
a	*<null>*		ට	ṭa	ප	pa	ක	ka	ළ	ḷa
ā	ælapilla	◌ා	ටා	ṭā	පා	pā	කා	kā	ළා	ḷā
æ	keṭi ædaya	◌ැ	ටැ	ṭæ	පැ	pæ	කැ	kæ	ළැ	ḷæ
ǣ	diga ædaya	◌ෑ	ටෑ	ṭǣ	පෑ	pǣ	කෑ	kǣ	ළෑ	ḷǣ
i	keṭi ispilla	◌ි	ටි	ṭi	පි	pi	කි	ki	ළි	ḷi
ī	diga ispilla	◌ී	ටී	ṭī	පී	pī	කී	kī	ළී	ḷī
u	keṭi pāpilla	◌ු	ටු	ṭu	පු	pu	කු	ku	ළු	ḷu
ū	diga pāpilla	◌ූ	ටූ	ṭū	පූ	pū	කූ	kū	ළූ	ḷū
ṛ	gætapilla	◌ෘ	ටෘ	ṭṛ	පෘ	pṛ	කෘ	kṛ	ළෘ	ḷṛ
ṝ	diga gætapilla	◌ෲ	ටෲ	ṭṝ	පෲ	pṝ	කෲ	kṝ	ළෲ	ḷṝ
ḷ	gayanukitta	◌ෟ	ටෟ	ṭḷ	පෟ	pḷ	කෟ	kḷ	ළෟ	ḷḷ
ḹ	diga gayanukitta	◌ෳ	ටෳ	ṭḹ	පෳ	pḹ	කෳ	kḹ	ළෳ	ḷḹ
e	kombuva	ෙ◌	ටෙ	ṭe	පෙ	pe	කෙ	ke	ළෙ	ḷe
ē	kombuva & hal-lakuṇa	◌ේ	ටේ	ṭē	පේ	pē	කේ	kē	ළේ	ḷē
ai	kombu deka	ෛ◌	ටෛ	ṭai	පෛ	pai	කෛ	kai	ළෛ	ḷai
o	kombuva & ælapilla	◌ො	ටො	ṭo	පො	po	කො	ko	ළො	ḷo
ō	kombuva & ælapilla & al-lakuṇa	◌ෝ	ටෝ	ṭō	පෝ	pō	කෝ	kō	ළෝ	ḷō
au	kombuva & gayanukitta	◌ෞ	ටෞ	ṭau	පෞ	pau	කෞ	kau	ළෞ	ḷau

Table 2.6: Examples of Devowelizers (Two Shapes)

	Shape 1					*Shape 2*				
C	ක ka	ග ga	ජ ja	න na	ය ya	ච ca	ට ta	ඩ da	ව va	ම ma
CX	ක් k	ග් g	ජ් j	න් n	ය් y	ච් c	ට් t	ඩ් d	ව් v	ම් m

Table 2.7: Examples of Consonant Signs

	Yaṃsaya					Rakarāṃśaya				
Incorrect Correct	ම්ය ම්‍ය mya	ත්ය ත්‍ය tya	ක්ය ක්‍ය kya	ෂ්ය ෂ්‍ය ṣya	න්ය න්‍ය nya	ප්ර ප්‍ර pra	ත්ර ත්‍ර tra	ක්ර ක්‍ර kra	ජ්ර ජ්‍ර jra	ශ්ර ශ්‍ර śra

the consonant syllabic. Mikami uses X to represent this devowelizer. A few examples are shown in Table 2.6.

In *Shape 1* a flag-like symbol is added at the end of the character, and in *Shape 2* the top ending line is doubled by reversing it.

- **Consonant signs** In some cases consonant signs are used to devowelize the inherent vowel. There are three consonant signs: *yaṃsaya*, *rakarāṃśaya* and *rēphaya*. If the consonant next to the devowelized consonant is ය(=ya) then ‍්‍ය(yaṃsaya) is used. If the consonant next to the devowelized consonant is ර(=ra), then rakarāṃśaya is used. These two consonant signs have a higher priority compared to the devowelizer. A few examples are shown in Table 2.7.

  The third consonant sign is called rēphaya and it is exactly equivalent to ර්(=r). As this rēphaya is extremely rare in modern Sinhala text, we do not take this into account in our evaluations. This consonant sign is optional in modern Sinhala. Mikami uses $\underline{C}$ to represent consonant signs.

- **Half-letters** Half letters can be used instead of devowelizers. However this is also optional. Nowadays these half letters are also very rare, thus we exclude them in our evaluations. A few examples are shown in Table 2.8.

- **Special characters (or Conjunct consonants)** Traditionally there were

Table 2.8: Examples of Half Letters

Modern Writing	න්ද	න්ධ	ත්ථ
Traditional Writing	න්‍ද	න්‍ධ	ත්‍ථ
	nda	ndha	ttha

many special characters in use, but currently only one special character remains. This is ඥ=ජ්(=j)+ඤ(=ña). In the Sinhala Unicode character set, this is considered an independent character. In our evaluation we also consider it an independent Sinhala character.

Definition of Character

We now give a definition of a Sinhala character.

Let T be an arbitrary Sinhala text and $f_0 ... f_n$ be the phonetic notation of T. This phonetic notation can be NLAC (*National Library at Calcutta Romanization*) [24] or IPA (*International Phonetic Alphabet*) [25, 26] or an input string of any phonetic based Sinhala input system. Then we can define a function such that, $T = phonetic\_to\_Sinhala(f_0 ... f_n)$.

$$\begin{aligned} &\exists i, j, \text{ and } i \leq j \\ &\quad T = phonetic\_to\_Sinhala(f_0 ... f_{i-1}) \\ &\qquad + phonetic\_to\_Sinhala(f_i ... f_j) \\ &\qquad + phonetic\_to\_Sinhala(f_{j+1} ... f_n) \end{aligned} \tag{2.1}$$

$$\begin{aligned} &\text{and } \forall k, i \leq k < j \\ &\quad T \neq phonetic\_to\_Sinhala(f_0 ... f_k) \\ &\qquad + phonetic\_to\_Sinhala(f_{k+1} ... f_n), \end{aligned} \tag{2.2}$$

where + means to simply concatenate the two strings.
Then, $phonetic\_to\_Sinhala(f_i ... f_j)$ is defined as a single Sinhala character.

According to Mikami's notation a Sinhala character can be represented by the following combinations.

$$S := V|C|C\underline{V}|CX|C\underline{C}|C\underline{CV}|\underline{D} \tag{2.3}$$

Table 2.9: Conjunct Consonants Derived from ක(=ka)

ක ka	කා kā	කැ kæ	කෑ kǣ	කි ki	කී kī	කු ku	කූ kū	කෙ ke	කේ kē	කෛ kai	කො ko	කෝ kō	කෞ kau
ක්‍ර kra	ක්‍රා krā	ක්‍රැ kræ	ක්‍රෑ krǣ	ක්‍රි kri	ක්‍රී krī	ක්‍රු kru	ක්‍රූ krū	ක්‍රෙ kre	ක්‍රේ krē	ක්‍රෛ krai	ක්‍රො kro	ක්‍රෝ krō	ක්‍රෞ krau
ක්‍ය kya	ක්‍යා kyā					ක්‍යු kyu	ක්‍යූ kyū	ක්‍යෙ kye	ක්‍යේ kyē		ක්‍යො kyo	ක්‍යෝ kyō	
ක් k	කෘ kṛ	කෲ kṝ	කෟ kḷ	කෳ kḹ									

Table 2.10: Noun Declensions

Singular			Plural		
ගස	(=gasa)	tree	ගස්	(=gas)	trees
ගසට	(=gasaṭa)	to tree	ගස්වලට	(=gasvalaṭa)	to trees
ගසේ	(=gasē)	in tree	ගස්වල	(=gasvala)	in trees
ගසෙන්	(=gasen)	from tree	ගස්වලින්	(=gasvalin)	from trees

Table 2.9 shows all the characters derived from Sinhala character ක(=ka). All other consonants also produce derivatives similarly. As a result Sinhala language has hundreds of characters.

2.1.2 Sinhala Words

Sinhala words can be divided into three grammatical categories: nouns, verbs and prepositions/postpositions.

Nouns A noun in Sinhala changes its form depending on the case[2] it carries. Table 2.10 shows the derivation of the noun ගස(=gasa: a tree). Note that case only changes the word ending. Sometimes, the same noun takes a completely different form in

[2]case: 格 විභක්ති

Table 2.11: Gender Changes of Nouns

	Pattern 1			Pattern 2		Pattern 3	
Male	parrot	cat	monkey	deer	cow	horse	peacock
	ගිරවා	බළලා	වඳුරා	මුවා	ගවයා	අශ්වයා	මොනරා
	giravā	baḷalā	vaňdurā	muvā	gavayā	aśvayā	monarā
Female	ගිරවි	බැළලි	වැඳිරි	මුව දෙන	ගව දෙන	වෙළඹ	සෙබඩ
	giravi	bæḷali	væňdiri	muva dena	gava dena	veḷaňba	sebaḍa

spoken Sinhala. For example, ගස(=gasa) becomes ගහ(=gaha). Sinhala nouns also change their forms with the gender. In this case not only the ending of the noun but the whole word changes as shown in Table 2.11. There are three patterns to construct the feminine form of the noun. Pattern 1 is to change the vowels of the masculine noun, pattern 2 is to add the word "දෙන(=dena)" and pattern 3 is to use a completely different word for the feminine noun.

Another interesting feature of Sinhala nouns is, two or more nouns conjoin together to produce a new compound noun:

පුරාණ(=purāṇa) + ඉක(=ika) + ත්ව(=tva)	→ පෞරාණිකත්ව(=paurāṇikatva) ancientry
රාජ(=rāja) + අභිෂේකය(=abhiṣēkaya)	→ රාජාභිෂේකය(=rājābhiṣēkaya) becoming king
අන්‍යෝන්‍ය(=anyōnya) + ආධාර(=ādhāra)	→ අන්‍යෝන්‍යාධාර(=anyōnyādhāra) mutual cooperation

Verbs The conjugations of verbs of spoken Sinhala differ from written Sinhala. As shown in Table 2.12, written Sinhala has very complicated grammar compared to spoken Sinhala, where the verb word form of written Sinhala depends on the tense, gender, and number, but in spoken Sinhala the verb word form depends only on the tense. In addition to these conjugations, passive voice forms, and agentive nouns can also be derived from a verb.

prepositions/postpositions In Sinhala, prepositions/postpositions have no derivations. Some prepositions/postpositions are written together with nouns and verbs with-

Table 2.12: Conjugations of Verb බලනවා(=balanavā: to see)

	Tense	Non-past		Past	
	Number	Singular	Plural	Singular	Plural
Written	1st *Person*	බලමි (=balami)	බලමු (=balamu)	බැලුවෙමි (=bæluvemi)	බැලුවෙමු (=bæluvemu)
Written	2nd *Person*	බලහි (=balahi)	බලහු (=balahu)	බැලුවෙහි (=bæluvehi)	බැලුවෙහු (=bæluvehu)
Written	3rd *Person* Male	බලයි (=balayi)	බලති (=balati)	බැලුවේය (=bæluvēya)	බැලුවෝය (=bæluvōya)
Written	3rd *Person* Female	බලන්නීය (=balannīya)	බලති (=balati)	බැලුවාය (=bæluvāya)	බැලුවෝය (=bæluvōya)
Spoken		බලනවා(=balanavā)		බැලුවා(=bæluvā)	

out white spaces. For example, -ත්(=-t: and), නො-(=no-: not), etc. are written with the corresponding noun or the verb. On the other hand, some postpositions are written as a separate word. For example නිසා(=nisā: because), පිණිස(=piṇisa: for), etc. are written as a separate word.

Sinhala text is written with white spaces; a white space usually indicates a word boundary. This may lead us to develop a word-based text input system which might make use of a word list. However we need to consider two possible problems:

1. word boundary problem:
   There are many cases where two or more words are written without any white spaces. This indicates that the definition of a Sinhala word is not strictly demarcated, at least not clearly recognized by ordinary Sinhala people [27]. Sometimes even the professionals' opinions become divided over this matter. To address this problem with a dictionary-based solution, we may need to have a word list which exhaustively list plausible word combinations.

2. word form variation problem:
   Sinhala nouns and verbs have a lot of derivatives which cannot be generated mechanically, given a situation where a comprehensive set of composition rules are not

elucidated by a proper research. In order to solve this problem, we basically have to enumerate all the derivations in a word list.

2.2 Classification of Input Systems

Text input systems can be mainly categorized into two categories: *direct input systems* and *predictive input systems*. A direct input system associates a key sequence into a unique character sequence, toward which the users do not have to choose from a set of candidates. Direct input systems are further classified according to: associative/non-associative and conversion/non-conversion. A predictive input system, on the other hand, provides a list of candidates in response to the user's key sequence; the users have to select their intended candidate from the menu. Predictive input systems are further divided depending on the linguistic unit on which the system relies.

2.2.1 Direct Input Systems

One of the dimensions which classifies the direct input systems is whether the system converts or not. Conversion systems convert a combination of keystrokes into a character or a part of a character, whereas non-conversion systems associate a single keystroke with a character or a part of a character. Non-conversion systems are feasible only for the languages which have only a very limited number of characters. For example QWERTY keyboard associates one keystroke with one Roman character.

Another dimension for the classification is associativity between keystrokes and entered character. Associative systems maintain a relationship between keystrokes and the intended character by means of geometric, phonetic or any other association. You can experience a non-associative text input by setting your computers keyboard settings into Dvorak keyboard layout if you have a physical QWERTY keyboard or vice versa. Then the characters printed on each key in the keyboard will produce a different character on the computer screen, where there is no phonetical or geometrical association between the key you press and the character produced on the computer.

Depending on being associative or non-associative and necessity of conversion, all the direct input systems can be categorized into four classes.

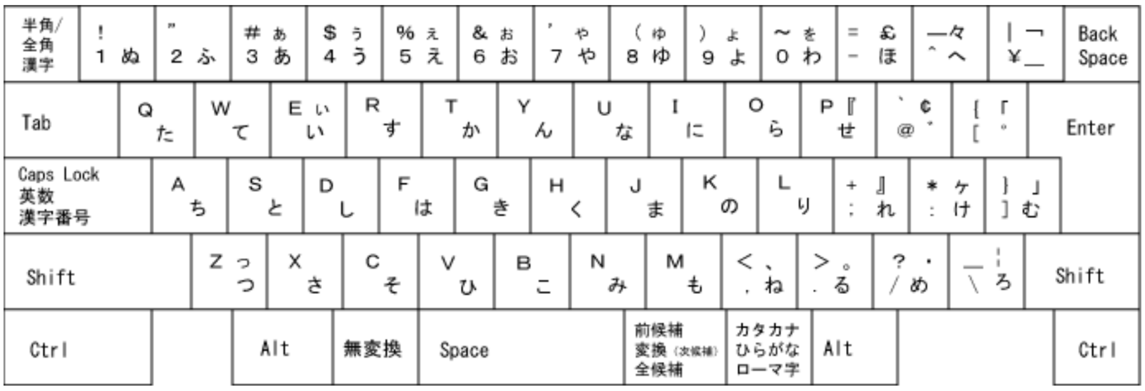


Figure 2.2: Japanese “kana” Keyboard Layout

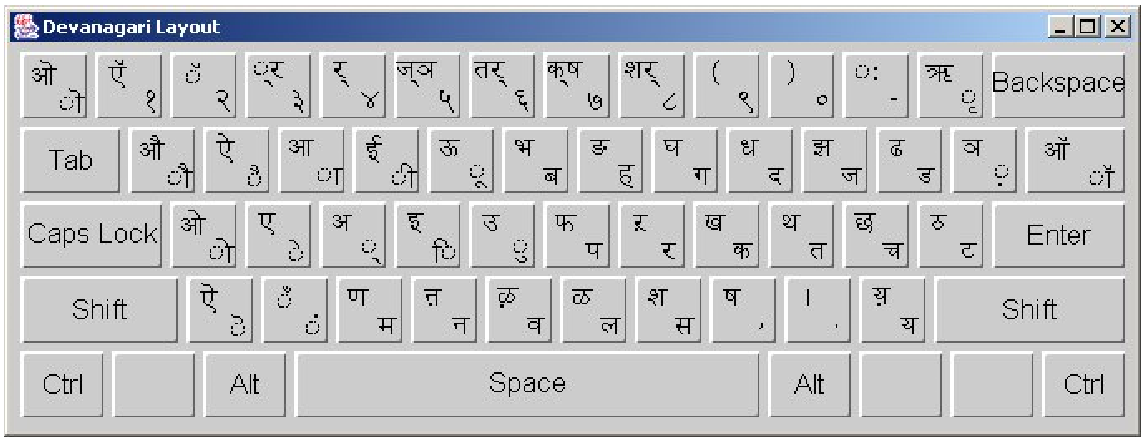


Figure 2.3: Inscript Keyboard Layout for Devanagari

1. Associative non-conversion systems

The simplest example of this kind of input system is Roman character text input using a QWERTY keyboard. Roman characters are allocated to a key in the QWERTY keyboard. The associations between the keys and the characters are brought about by printing the characters themselves on the keys.

When we consider the Japanese language, there are approximately fifty *Hiragana* characters. Using the Japanese keyboard these characters can be input by a single keystroke as shown in Figure 2.2. This system is called “*kana nyuryoku*,” representing an example of associative and non-conversion direct input.

Inscript is a common keyboard designed to input Indic scripts such as Bengali, De-

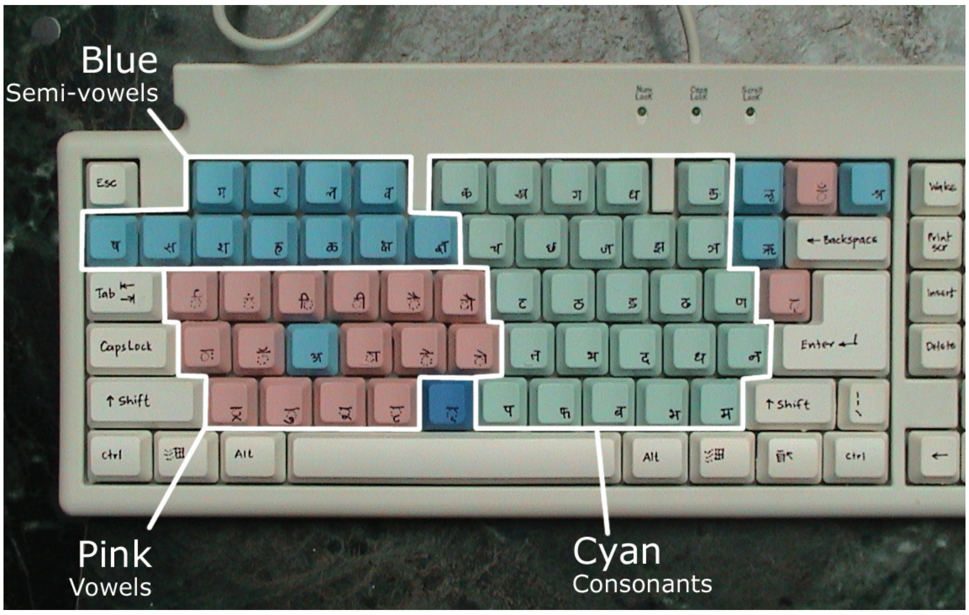


Figure 2.4: Keylekh Keyboard Layout for Devanagari

vanagari, Gujarati, Gurmuki, Kannada, Malayalam, Oriya, Tamil, and Telugu. Figure 2.3 shows the Inscript keyboard layout for Devanagari[3].

Figure 2.4 shows *Keylekh* [28] keyboard layout which is also designed to input Devanagari script. The specialty of this keyboard layout is that, the characters are placed in alphabetical order of the Devanagari script. Therefore even the characters are not printed on the keyboard, still the system is associative; because the user could know what are the characters produced by each key even without any training.

In addition Hangul keyboard for Korean, and Thai keyboards also assign parts of characters into keys, and the symbols are printed on the keyboard.

2. Non-associative non-conversion systems

All the keyboard layouts classified as “associative non-conversion systems,” except *Keylekh* keyboard for Devanagari, are associative if and only if the corresponding characters are printed on the key. Most of the keyboards used all over the world, and especially in Japan, Sri Lanka, and India have the Roman characters printed on them in QWERTY order [29]. Thus there is no problem of typing Roman characters using them. However,

[3]Devanagari is a very popular Indic script used to write languages such as Sanskrit, Prakrit, Hindi, Nepali, Marathi, Bhili, Konkani, Bhojpuri, Magahi, Maithili, Newari, etc.

Table 2.13: Japanese Roman Transliteration

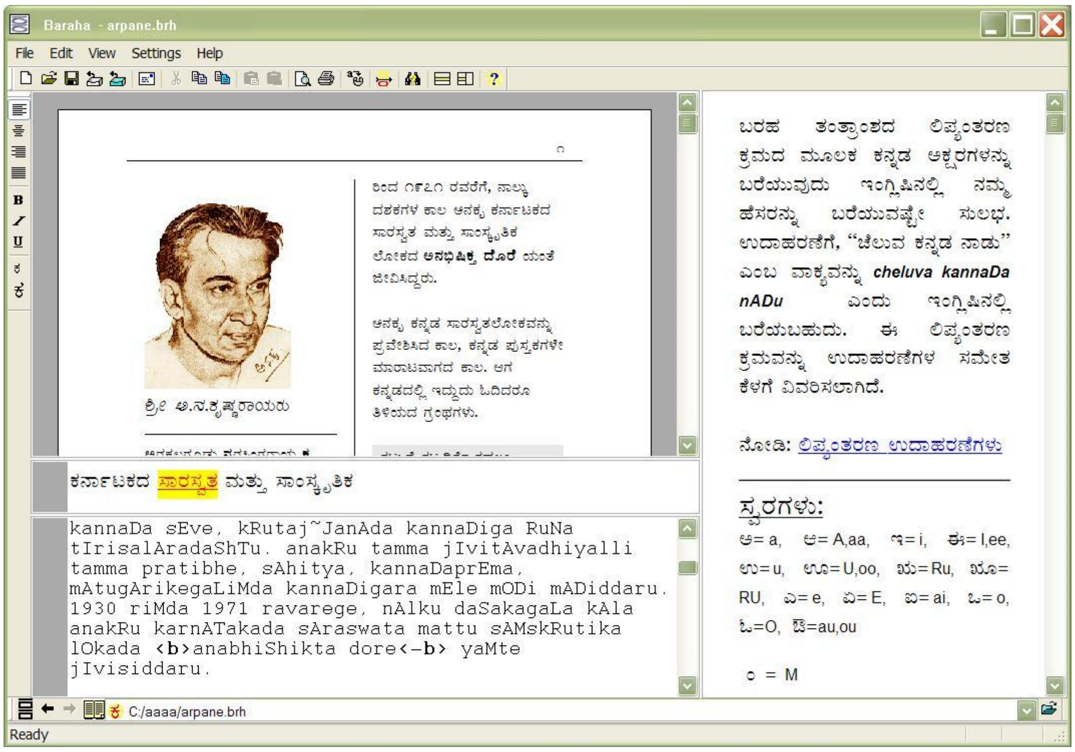
あ a	か ka	が ga	さ sa	ざ za	た ta	だ da	な na	は ha	ば ba	ぱ pa	ま ma	や ya	ら ra	わ wa
い i	き ki	ぎ gi	し si shi	じ zi ji	ち ti chi	ぢ di	に ni	ひ hi	び bi	ぴ pi	み mi		り ri	
う u	く ku	ぐ gu	す su	ず zu	つ tu tsu	づ du	ぬ nu	ふ hu	ぶ bu	ぷ pu	む mu	ゆ yu	る ru	
え e	け ke	げ ge	せ se	ぜ ze	て te	で de	ね ne	へ he	べ be	ぺ pe	め me		れ re	
お o	こ ko	ご go	そ so	ぞ zo	と to	ど do	の no	ほ ho	ぼ bo	ぽ po	も mo	よ yo	ろ ro	を wo
	きゃ kya	ぎゃ gya	しゃ sya sha	じゃ zya ja	ちゃ tya cha	ぢゃ dya	にゃ nya	ひゃ hya	びゃ bya	ぴゃ pya	みゃ mya		りゃ rya	ん nn n’
	きゅ kyu	ぎゅ gyu	しゅ syu shu	じゅ zyu ju	ちゅ tyu chu	ぢゅ dyu	にゅ nyu	ひゅ hyu	びゅ byu	ぴゅ pyu	みゅ myu		りゅ ryu	
	きょ kyo	ぎょ gyo	しょ syo sho	じょ zyo jo	ちょ tyo cho	ぢょ dyo	にょ nyo	ひょ hyo	びょ byo	ぴょ pyo	みょ myo		りょ ryo	

when you want to input your local language to a computer outside, or in a foreign country, it is very troublesome unless you have a good practice on touch typing. This may be one reason why the Japanese *kana nyuryoku* and the Devanagari *Inscript* keyboard are not so popular.

3. Associative conversion systems

Compared to *kana nyuryoku*, Japanese Roman transliteration input system which is also known as Japanese *romaji nyuryoku* is very popular input system used to input Japanese *Hiragana*. Table 2.13 shows the transliteration scheme.

Baraha [30, 31] is a transliteration scheme available for Indic scripts. Baraha supports Kannada, Devanagari, Tamil, Telugu, Malayalam, Gujarati, etc. Figure 2.5 shows some screenshots of Baraha input system.



೧

ರಿಂದ ೧೯೭೧ ರವರೆಗೆ, ನಾಲ್ಕು ದಶಕಗಳ ಕಾಲ ಅನಕೃ ಕರ್ನಾಟಕದ ಸಾರಸ್ವತ ಮತ್ತು ಸಾಂಸ್ಕೃತಿಕ ಲೋಕದ **ಅನಭಿಷಿಕ್ತ ದೊರೆ** ಯಂತೆ ಜೀವಿಸಿದ್ದರು.

ಅನಕೃ ಕನ್ನಡ ಸಾರಸ್ವತಲೋಕವನ್ನು ಪ್ರವೇಶಿಸಿದ ಕಾಲ, ಕನ್ನಡ ಪುಸ್ತಕಗಳೇ ಮಾರಾಟವಾಗದ ಕಾಲ. ಆಗ ಕನ್ನಡದಲ್ಲಿ ಇದ್ದುದು ಓದಿದರೂ ತಿಳಿಯದ ಗ್ರಂಥಗಳು.

ಶ್ರೀ ಅ.ನ.ಕೃಷ್ಣರಾಯರು

ಕರ್ನಾಟಕದ ಸಾರಸ್ವತ ಮತ್ತು ಸಾಂಸ್ಕೃತಿಕ

```
kannaDa sEve, kRutaj~JanAda kannaDiga RuNa
tIrisalAradaShTu. anakRu tamma jIvitAvadhiyalli
tamma pratibhe, sAhitya, kannaDaprEma,
mAtugArikegaLiMda kannaDigara mEle mODi mADiddaru.
1930 riMda 1971 ravarege, nAlku daSakagaLa kAla
anakRu karnATakada sAraswata mattu sAMskRutika
lOkada <b>anabhiShikta dore<-b> yaMte
jIvisiddaru.
```

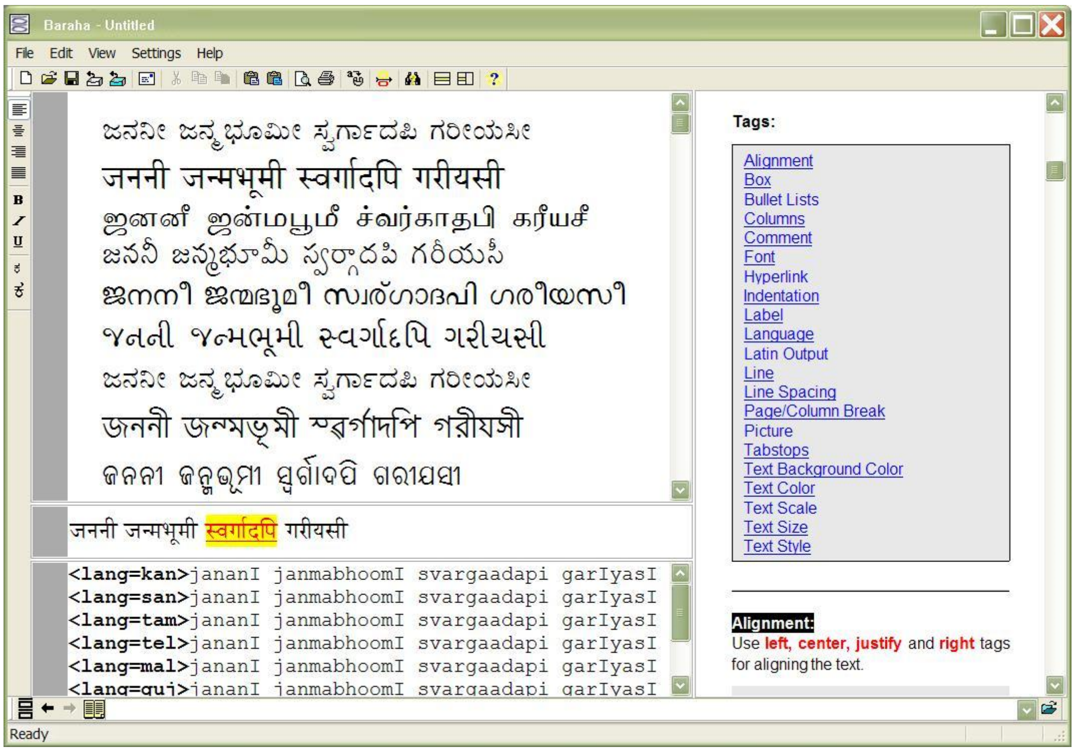
ಬರಹ ತಂತ್ರಾಂಶದ ಲಿಪ್ಯಂತರಣ ಕ್ರಮದ ಮೂಲಕ ಕನ್ನಡ ಅಕ್ಷರಗಳನ್ನು ಬರೆಯುವುದು ಇಂಗ್ಲಿಷಿನಲ್ಲಿ ನಮ್ಮ ಹೆಸರನ್ನು ಬರೆಯುವಷ್ಟೇ ಸುಲಭ. ಉದಾಹರಣೆಗೆ, "ಚೆಲುವ ಕನ್ನಡ ನಾಡು" ಎಂಬ ವಾಕ್ಯವನ್ನು ***cheluva kannaDa nADu*** ಎಂದು ಇಂಗ್ಲಿಷಿನಲ್ಲಿ ಬರೆಯಬಹುದು. ಈ ಲಿಪ್ಯಂತರಣ ಕ್ರಮವನ್ನು ಉದಾಹರಣೆಗಳ ಸಮೇತ ಕೆಳಗೆ ವಿವರಿಸಲಾಗಿದೆ.

ನೋಡಿ: ಲಿಪ್ಯಂತರಣ ಉದಾಹರಣೆಗಳು

ಸ್ವರಗಳು:

ಅ= a, ಆ= A,aa, ಇ= i, ಈ= I,ee, ಉ= u, ಊ= U,oo, ಋ= Ru, ೠ= RU, ಎ= e, ಏ= E, ಐ= ai, ಒ= o, ಓ=O, ಔ=au,ou

ಂ = M



ಜನನೀ ಜನ್ಮಭೂಮೀ ಸ್ವರ್ಗಾದಪಿ ಗರೀಯಸೀ
जननी जन्मभूमी स्वर्गादपि गरीयसी
ஜனனீ ஜன்மபூமீ ச்வர்காதபி கரீயசீ
జననీ జన్మభూమీ స్వర్గాదపి గరీయసీ
ജനനീ ജന്മഭൂമീ സ്വര്ഗാദപി ഗരീയസീ
જનની જન્મભૂમી સ્વર્ગાદપિ ગરીયસી
ಜನನೀ ಜನ್ಮಭೂಮೀ ಸ್ವರ್ಗಾದಪಿ ಗರೀಯಸೀ
জননী জন্মভূমী স্বর্গাদপি গরীযসী
ଜନନୀ ଜନ୍ମଭୂମୀ ସ୍ୱର୍ଗାଦପି ଗରୀଯସୀ

जननी जन्मभूमी स्वर्गादपि गरीयसी

```
<lang=kan>jananI janmabhoomI svargaadapi garIyasI
<lang=san>jananI janmabhoomI svargaadapi garIyasI
<lang=tam>jananI janmabhoomI svargaadapi garIyasI
<lang=tel>jananI janmabhoomI svargaadapi garIyasI
<lang=mal>jananI janmabhoomI svargaadapi garIyasI
<lang=guj>jananI janmabhoomI svargaadapi garIyasI
```

Tags:

Alignment
Box
Bullet Lists
Columns
Comment
Font
Hyperlink
Indentation
Label
Language
Latin Output
Line
Line Spacing
Page/Column Break
Picture
Tabstops
Text Background Color
Text Color
Text Scale
Text Size
Text Style

Alignment:
Use **left, center, justify** and **right** tags for aligning the text.

Figure 2.5: Screenshots of Baraha Indian Language Software

Table 2.14: Japanese Associative Type Kanji Direct Input Systems

漢字	KIS		KANTEC	
	keystrokes	Clue	keystrokes	Clue
漢	チナ	漢の国→ China	カフ	漢文
字	レタ	letter	シタ	字体
直	ナオ		スク	直ぐ
接	セツ		メセ	面接
入	イリ		ソニ	挿入
力	リキ		リキ	

Japanese language uses not only the “kana” (*Hiragana* and *Katakana*) characters (phonograms) but also *Kanji* (Chinese) characters (ideograms) [32]. Written Japanese uses about 50,000 Kanji characters. In 1981, in an effort to make it easier to read and write Japanese, the Japanese government introduced the *Joyo Kanji Hyo* (List of Chinese Characters for General Use), which includes 1,945 regular characters, plus 166 special characters used only for people's names. All government documents, newspapers, textbooks and other publications for non-specialists use only these Kanji characters [33].

Some Japanese typing professionals use direct input systems to enter Kanji characters into computers. KIS input system [34] and NE–KANTEX (KANTEC) [35] input system are two popular input systems of this kind. With these systems, each Kanji character can be input using two keystrokes. These two letters have some relationship to the character, indicating that these are associative systems. For example “漢” means China, so in KIS this character can be input using two keystrokes: “チ”(=chi) and “ナ”(=na). Similarly a Japanese phrase: “漢英学習字典” can be typed as “チナエイマナナウレタノリ.” Table 2.14 shows a few examples of key assignments in KIS and KANTEC input systems. These systems are mainly used by professionals because of its efficiency: the users do not have to check the results of Kanji conversions.

4. Non-associative conversion systems

A more efficient way to input Kanji characters is to use non-associative conversion system like T-code [36, 37]. In this system the QWERTY keyboard is divided into two areas: *Left* and *Right*, as shown in Table 2.15. Each area has four rows and five lines. In this system by striking two key strokes, Kanji characters can be input.

There are four patterns to select those two keystrokes:

Table 2.15: QWERTY Keyboard Used in T-Code

	Left					Right				
	1	2	3	4	5	1	2	3	4	5
1	1	2	3	4	5	6	7	8	9	0
2	Q	W	E	R	T	Y	U	I	O	P
3	A	S	D	F	G	H	J	K	L	;
4	Z	X	C	V	B	N	M	,	.	/

Table 2.16: T-Code RL Characters Table

 請境系探象 盛革突温捕 依織借須訳	 尚賀岸責漁 益援周域荒 織父枚乱香	 舎喜幹丘糖 康徒景処ぜ 譲へ模降走	 布苦圧恵固 邦舞雑漢緊 激干彦均又	 姿絶密秘押 衆節杉肉除 測血散笑弁
酸昼炭稲湯 歩回務島開 給員ど代レ 相家的対歴	貿捜異隣旧 キせ区百木 分よル千ア 付プばュ作	攻焼闘奈夕 や出タ手保 7か(トれ 内工八テ見	盤帯易速拡 コ山者発立 きっ日国二 九名川機チ	汽換延雪互 ナ金マ和女 上く8え年 サ建パ第入
桜瀬鳥催障 中スもお定 3と○てる ッ人三京ち	典博筋忠乳 わラ東生ろ ーした一が ロク万方フ	採謡希仏察 う4)十リ い、の51 んまンつ四	君純副盟標 あこ6学月 。*0・2 けイす電地	犯余堀肩療 本さら高シ ではになを 業時「長み
呼幅歓功盗 店持町所ほ 行ド円小ジ 海道ず西げ	紀破郡抗幡 全じ自議明 通カ社野同 当理メウグ	房績識属衣 バ部六経動 だり一め大 不合面政オ	去疑ぢ綿離 後間場ニ産 新」9子五 委化ビ目市	秒範核影麻 問ム七住北 事田会前そ 気売下都株

1. *LR* One keystroke from the *Left* area and one from the *Right*
2. *RL* One keystroke from the *Right* area and one from the *Left*
3. *LL* Both keystrokes from the *Left* area
4. *RR* Both keystrokes from the *Right* area

For each pattern a table of characters is assigned. Table 2.16 shows the characters assigned for Right Left (*RL*) key combination. For example if you want to input the Kanji character "国," first you have to find where this character is placed on Table 2.16. The coordinates of the character "国" is 3rd row, 4th column, in the *Box* located at 2nd row, 4th column. Now you have to strike the keys at the same coordinates it the QWERTY keyboard: coordinates of the character from right area, and coordinates of the *Box* from the left area. Therefore by striking the two keys: $\boxed{\text{L}}$ and $\boxed{\text{R}}$, you can input the character "国." This system covers most frequent 1600(=4 patterns×4 rows×5 columns×4 rows×5 columns) Kanji characters.

Using T-Code a very high typing speed is achievable, but it is not user-friendly. Therefore only the Japanese typing professionals use these kinds of input systems.

As discussed, direct text input systems can be efficient, hence are suitable for professional users. Especially, non-associative systems can be designed so as to maximize the efficiency at the cost of user-friendliness. On the other hand, general computer users may prefer more memorable and learnable systems, especially if his/her language has a huge number of characters.

```
>ぶんしょ  (typed as: "bunsho")
  1.文書  2.分署  3.ぶんしょ  4.ブンショ
>1
文書>にゅうりょく  (typed as: "nyuuryoku")
  1.入力  2.ニュウリョク  3.にゅうりょく
文書>1
文書入力>の  (typed as: "no")
  1.の  2.野  3.乃  4.之  5.埜  6.ノ
文書入力>1
文書入力の>けんきゅう  (typed as: "kenkyuu")
  1.研究  2.建久  3.けんきゅう  4.ケンキュウ
文書入力の>1
文書入力の研究
```

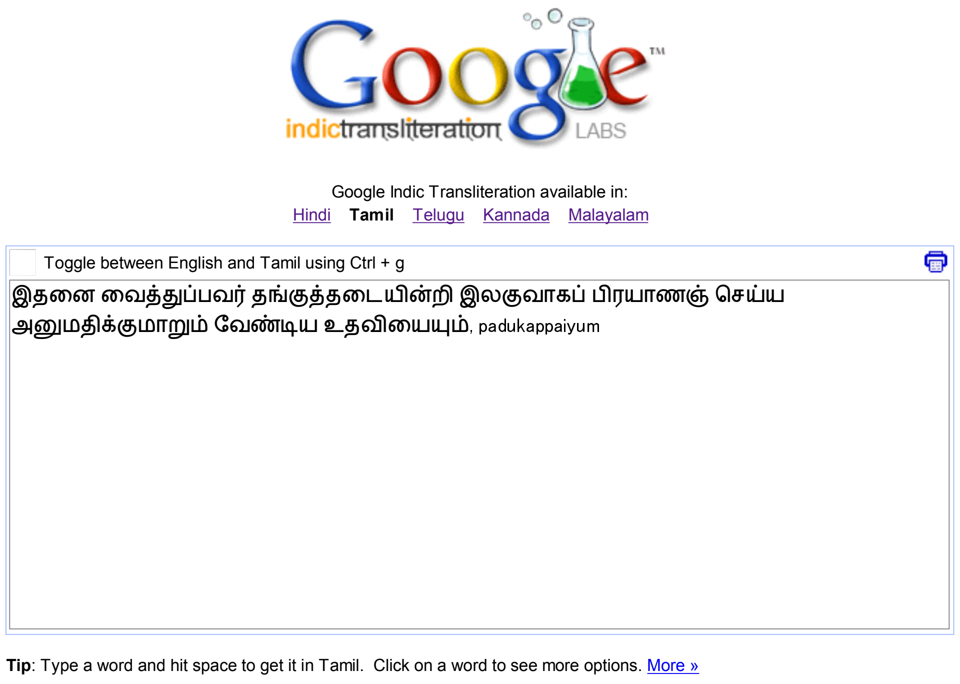
Figure 2.6: Japanese Word-Based Text Entry Example

2.2.2 Predictive Systems

Predictive systems are preferred or sometimes highly required by general users when the language has a large set of characters. Japanese is a typical language of such kind. Earliest studies regarding predictive input system have been started in 1960s [38]. The first Japanese word processor was commercialized in 1970s [39].

Therefore, it is not an exaggeration to say that most advanced predictive input systems are Japanese input systems. As mentioned above, Japanese writing uses Kanji characters which are ideograms. Each Kanji character has several readings. For example "生" can be read as: "*sei*, *shō*, *i*(kiru), *u*(mu), *o*(u), *ki*, *nama*, *ha*(eru)." On the other hand, different Kanji characters have the same reading. For example all the followings have a common reading "*kai*": 会, 貝, 買い, 解, 回, 界, 快, 戒, 怪, etc.

Because of the bi-directional ambiguity, character level conversion systems are not efficient enough; more linguistically rich context is necessary to narrow down the conversion



Google Indic Transliteration available in:
Hindi **Tamil** Telugu Kannada Malayalam

Toggle between English and Tamil using Ctrl + g

இதனை வைத்துப்பவர் தங்குத்தடையின்றி இலகுவாகப் பிரயாணஞ் செய்ய அனுமதிக்குமாறும் வேண்டிய உதவியையும், padukappaiyum

Tip: Type a word and hit space to get it in Tamil. Click on a word to see more options. More »

Figure 2.7: Screenshot of Google Indic Transliteration (Tamil)

candidates. Therefore, word-based and phrase-based predictive input systems became popular.

Word-based Predictive Systems

Figure 2.6 shows an example of word-based predictive Japanese input system [40]. First the user inputs the reading of the intended word using Roman transliteration method (or using Kana non-conversion input), as soon as the user presses the spacebar the candidate words appear in a menu. Then the user can select the menu item by pressing the number of the item. In the example by typing "*bunsho*," first the user gets the *Hiragana* representation of it: "ぶんしょ." When the user presses the spacebar the system displays four menu items: "1. 文書 2. 分署 3. ぶんしょ 4. ブンショ," where user can select the intended word "文書" by pressing the numeric key: $\boxed{1}$.

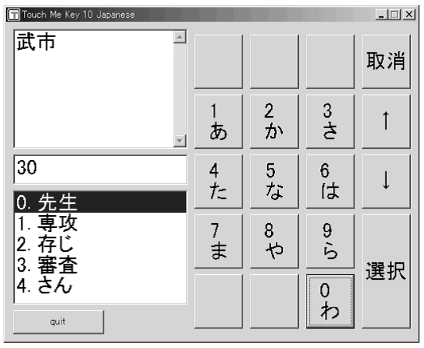
Google Indic Transliteration is a similar word-based input system available for Hindi, Tamil, Telugu, Kannada, and Malayalam [41]. Figure 2.7 shows a screenshot of Google Indic transliteration for Tamil.

Figure 2.8 shows an example of Japanese multitap input on a phone keypad. Here by

1 あ (*ϕ*)	2 か (k)	3 さ (s)
4 た (t)	5 な (n)	6 は (h)
7 ま (m)	8 や (y)	9 ら (r)
∗ ゛	0 わ (w/ŋ)	♯

3333000333311
↓
せんせい (sensei)
↓
先生 (teacher)

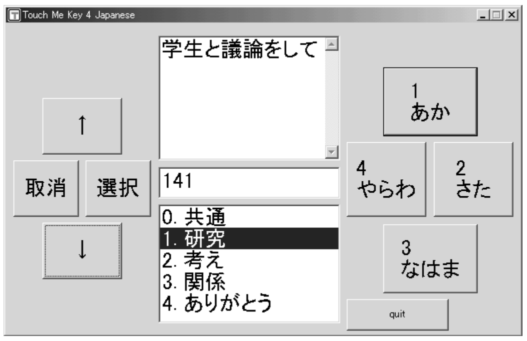
Figure 2.8: Example of Japanese Multitap Text Entry on a Phone Keypad



武市

30

0. 先生
1. 専攻
2. 存じ
3. 審査
4. さん



学生と議論をして

141

0. 共通
1. 研究
2. 考え
3. 関係
4. ありがとう

Figure 2.9: Screenshot of "Touch Me Key 10 Japanese" System

Figure 2.10: Screenshot of "Touch Me Key 4 Japanese" System

tapping key: $\boxed{3}$ for four times user can get "せ"(=se), then key: $\boxed{0}$ for three times to get "ん"(=ŋ), "せ" again, and finally two taps on key: $\boxed{1}$ to produce a "い"(=i). After typing the word in *Hiragana* phonograms, user can convert the word into Kanji ideograms. Even though this system is widely used in Japanese mobile phones, it is not very efficient.

In order to improve the efficiency "Touch Me Key 10 Japanese" [42] has been proposed. In this system, instead of keep tapping on the same key, users hit each key only once. Using this highly ambiguous input sequence, the system produces a list of possible candidates. Figure 2.9 shows an example of this system. Essentially the user has to strike four keys: $\boxed{3}\boxed{0}\boxed{3}\boxed{1}$ to produce "先生"(=せんせい: sensei), but in this system, with the support of the *auto completion techniques* [43], the user can get the intended word "先生" using only the first two keystrokes: $\boxed{3}\boxed{0}$.

Later "Touch Me Key" was able to reduce the number of keys to the utmost limit. As shown in Figure 2.10 "Touch Me Key 4 Japanese" [44] system uses only four keys to enter text, and there are four control keys.

Phrase-based Predictive Systems

Table 2.17: Example of Phrase-Based Kana-Kanji Conversions

(a)	(b)
わたしは <u>かきを</u> たべる。	<u>かきを</u> およみください。
(typed as: "watashiha<u>kakiwo</u>taberu.")	(typed as: "<u>kakiwo</u>oyomikudasai.")
↓	↓
私は <u>柿を</u> 食べる。	<u>下記を</u> お読みください。
I eat <u>a persimmon</u>.	Please read <u>the following</u>.

Currently phrase-based predictive input systems are widely used to input Japanese [45] and Chinese [46]–[48]. Table 2.17 shows a pair of examples of Japanese phrase-based predictive input system. Note that the same *Hiragana* presentation "かきを"(=kakiwo) has been converted into two different Kanji presentations "柿"(: a persimmon) and "下記"(: the following). This was possible because the language model used here was able to decide that "*persimmon*" is strongly connected with "*eating*" and, "*the following*" with "*reading*." Using similar language models, "context-based auto completion systems" was proposed [49]. Note that the term *auto completion* is introduced to denote the system's function to foresee or look-ahead keystrokes that have not been entered, while *prediction* simply denotes system's behavior to produce a list of candidates.

2.3 Review of Existing Sinhala Input Systems

This section reviews three representative Sinhala input systems proposed so far: *Wijesekara*, *Kaputadotcom* and *Natural SinGlish*. All of them have their own shortcomings.

Wijesekara is a keyboard layout which is used in old Sinhala typewriters. Therefore it is very efficient, where it assigns more rare characters into shifted keys. However, has very poor user-friendliness, because the key layout has no phonetical or geometrical connection with the Roman character keyboard layout; which is known as non-associative input systems.

Kaputadotcom keyboard layout assigns Roman character keys to Sinhala character components, considering phonetical and geometrical relationships in between. Therefore it was able to improve the user-friendliness, but at a cost of efficiency.

Natural SinGlish is a converter which has improved the user-friendliness further, by introducing a transliteration scheme. Of cause this system is less efficient compared to the other two non-conversion systems: *Wijesekara* and *Kaputadotcom*.

Considering good and bad point of these existing systems, we propose a better phonetically-principled conversion system in Chapter 4. Then, Chapter 5 proposes the first predictive input system for Sinhala, which has the maximum user-friendliness.

2.3.1 *Wijesekara*

The *Wijesekara* is a direct input system with non-conversion key assignment, which was originally used in Sinhala typewriters. In this system each Sinhala character component (that is, V, C, $\underline{D}$, $\underline{V}$, X or $\underline{C}$ in Equation (2.3)) is assigned to a key. This system has been designed to maximize Sinhala typing efficiency by assigning frequently used Sinhala character components to unshifted keys and less frequent ones to shifted keys. Table 2.18 demonstrates a text entry example using *Wijesekara*. Note that in this example, most of the keystrokes are unshifted keystrokes.

Even though this layout is highly efficient, and supported by most of the operating systems as their default Sinhala input system, still it is not widely spread among novice Sri Lankan computer users. The main reason for that is the lack of user-friendliness, where there is no phonetical or geometric association between a key and the corresponding Sinhala character component.

Figure 2.11 shows complete key layout of *Wijesekara*.

2.3.2 *Kaputadotcom*

Kaputadotcom [50] is an associative non-conversion system, which was a popular Sinhala keyboard layout: Figure 2.12, before Unicode support for Sinhala [9] was introduced. *Kaputadotcom* provides a set of key assignments by considering phonetic and/or geometric relationships between a key and the corresponding Sinhala character. For example, Sinhala අ(=a) is assigned to 0x61 (=ASCII 'a'), where there is a phonetic relationship. On the other hand, Sinhala vowel sign 'ෙ' is assigned to '@' where there is a geometric relationship. For example, "ආයුබෝවන්"(=āyubōvan: welcome) can be typed as '*a`yE@b`~vn~*', here (අ,a), (ය,y), (බ,b) can be considered phonetically related, and (ෙ,@), (ා,`) can be

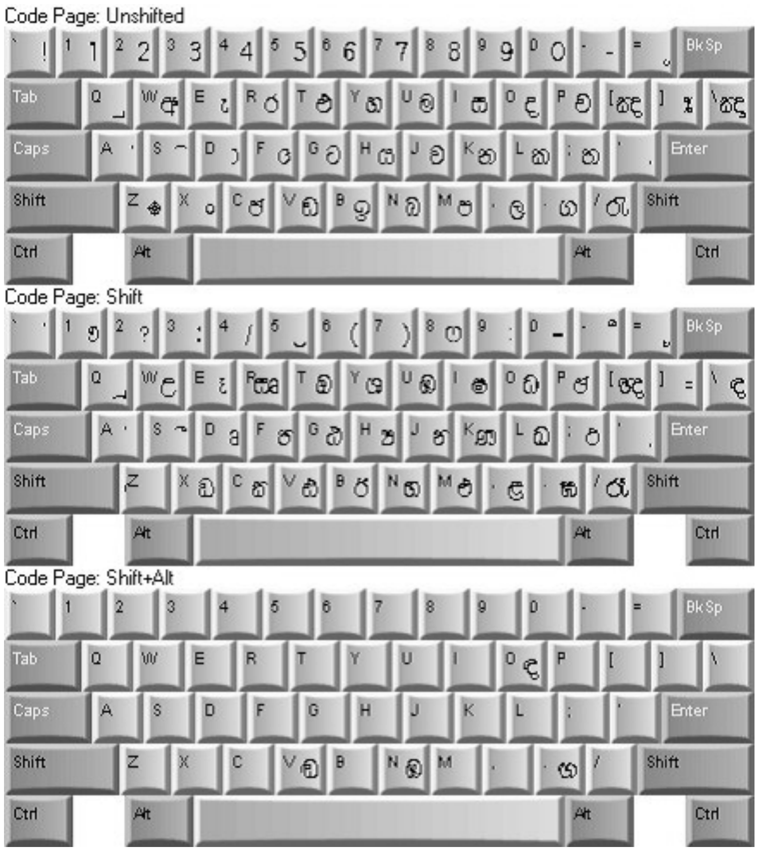


Figure 2.11: *Wijesekara* Keyboard Layout

Table 2.18: Text Entry Example of *Wijesekara*

Output text	මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.
Input key sequence	*fuh orkakdg wjysr ndOdj,ska f;drj ksoyfia .uka lsrSug iy wjYHjk wdOdr o wdrlaIdj o i,iajd fok f,i;a wod, j.lSua ork ish,q fokdf.kau Y`S ,xld m`cd;dka;`sl iudcjdoS ckrcfha ckdOsm;s b,a,qualr o wfmalaIdlr oisgS'*

Table 2.19: Text Entry Example of *Kaputadotcom*

Output text	මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.
Input key sequence	*@my qrn˜n`t avhQr b`{`vlQn˜ @w`rv nQqh@s˜ gmn˜ kQrWmt sh avX vn a`{`r q a`rk˜;`v q sls˜v` @qn @lsw˜ aq`l vgkWm| qrn sQylE @qn`@gn˜m XYW lAk` pYj`w`n˜wYQk sm`jv`qW jnrj@y˜ jn`{QpwQ il˜lEm|kr q a@p˜k˜;`kr q sQtW.*

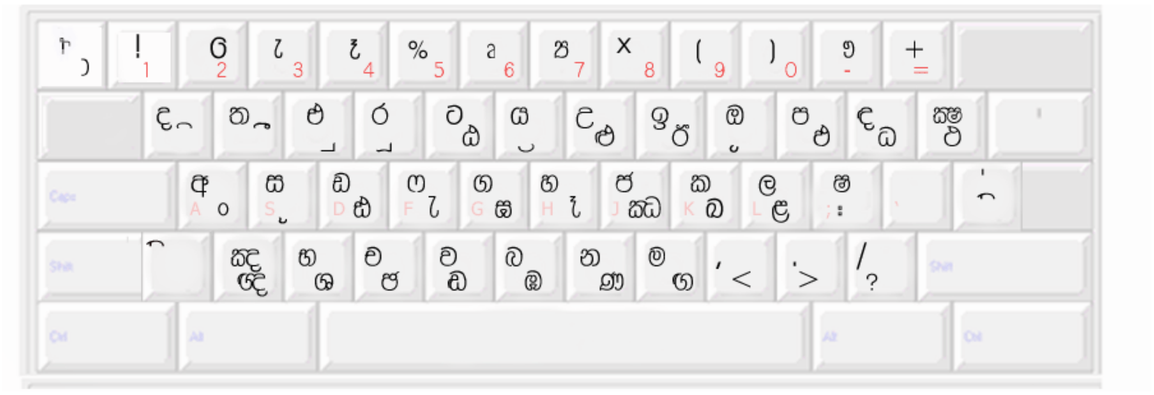
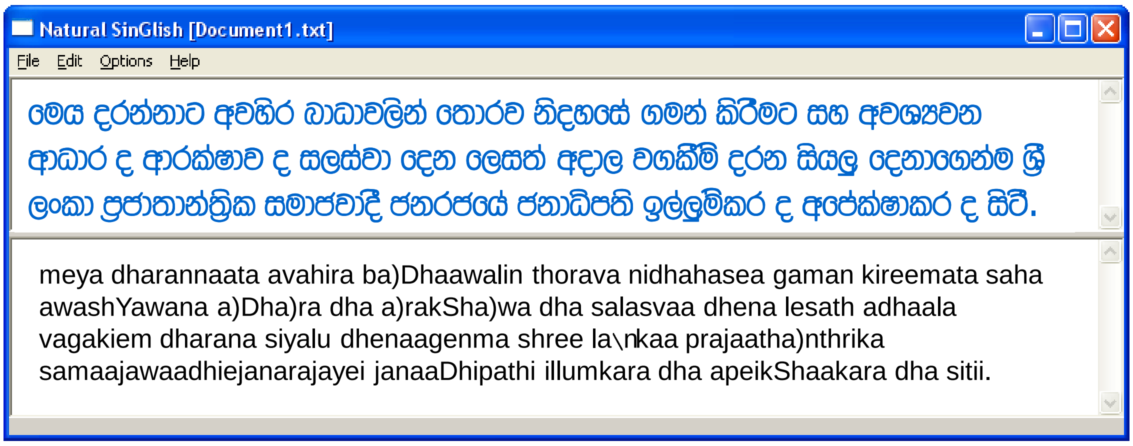


Figure 2.12: *Kaputadotcom* Keyboard Layout

considered geometrically related. Based on these phonetic and geometric relationships user-friendliness has been slightly improved compared to *Wijesekara*.

However *Kaputadotcom* is an incomplete system, where it has no key assignments for rare Sinhala characters: ඏ(=ḷ), ඐ(=ḹ), ඍ(=ṛ), ඎ(=ṝ) and ඞ(=ṅa). The example in Table 2.19 shows that this system uses a lot of shifted keystrokes where the efficiency can go down. In spite of this problem, *Kaputadotcom* was very popular not only among the novice Sinhala users but also Sinhala typing experts, where several Sinhala newspapers published on the internet still use it.



මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.

meya dharannaata avahira ba)Dhaawalin thorava nidhahasea gaman kireemata saha awashYawana a)Dha)ra dha a)rakSha)wa dha salasvaa dhena lesath adhaala vagakiem dharana siyalu dhenaagenma shree la\nkaa prajaatha)nthrika samaajawaadhiejanarajayei janaaDhipathi illumkara dha apeikShaakara dha sitii.

Figure 2.13: Screenshot of *Natural SinGlish* Text Entry Application

2.3.3 *Natural SinGlish*

Natural SinGlish [51] is a conversion-based direct input system that was proposed to solve the problems with non-conversion input systems: a key may not have phonetical association with the corresponding Sinhala character component. *Natural SinGlish* was introduced by A. D. R. Sasanka as an application rather than an application independent input system. It converts the input sequence into Sinhala characters that are more natural for users. English spellings and pronunciations are considered in this system. Figure 2.13 shows a text entry example of *Natural SinGlish*. Since the Sinhala language has many more characters than Roman characters, a simple Roman character transliteration of Sinhala text is likely to be highly ambiguous. To overcome this problem, this system has introduced the following techniques:

- 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, therefore it is quite complex. Characters with phonetic similarities cannot be typed in a similar manner:

Ex. 1)	ka	$\rightarrow$	ක(=ka)	and	kha	$\rightarrow$	ඛ(=kha)
	ta	$\rightarrow$	ට(=ṭa)	but	tha	$\not\rightarrow$	ඨ(=ṭha)
Ex. 2)	da	$\rightarrow$	ඩ(=da)	and	nnda	$\rightarrow$	ඬ(= ňḍa)
	ba	$\rightarrow$	බ(=ba)	but	nnba	$\not\rightarrow$	ඹ(= m̌ba)

In some cases, this system is not very efficient because it uses many upper case letters in the middle of the words, forcing users to press and release the shift key frequently.

2.4 Desiderata for Sinhala Input System

Based on the discussions given in the previous chapter as well as the sections in this chapter, the desiderata for an effective Sinhala input system, targeted to general Sinhala computer users, are summarized as follows.

Given the target users who are familiar with English operating systems, the input system should be implemented assuming the use of a Roman character keyboard, rather than specially designed Sinhala keyboards.

Given the most significant feature of Sinhala, having a large syllabic character set, the input system is inevitably conversion-based, where the input sequence should be as much as *user-friendly*; the required/acceptable input sequence should be user-intuitive and/or principled in some way.

Given another prominent feature of Sinhala, having no standardized ways of Sinhala transliteration, also requires that the input sequences should be user-friendly; they should cover a range of transliterations given by various users.

To fulfill the preceding requirement, the input system should be able to handle possibly ambiguous input sequences; yet achieving certain level of *efficiency* is highly desirable, given the fact that efficiency has been considered very important dimension of text input.

In summary, we should explore an adequate technical architecture and pursue its effective implementation in order to achieve these desiderata.

Chapter 3

Evaluation Methodology

This chapter proposes a new methodology to evaluate Sinhala input systems. First we discuss perspectives for evaluation in Section 3.1, and the existing measures available for evaluating input systems in Section 3.2. These measures mainly concern efficiency, sometimes including correctness, by measuring input speeds and error rates. As stated in Chapter 1, text input systems should be evaluated not only by the efficiency but the user-friendliness, especially when the users are not professionals. Given the requirement to evaluate Sinhala input systems, Section 3.3 argues that the existing efficiency measures can be improved, and proposes a modified measure. This section also proposes a novel measure to assess the user-friendliness of Sinhala input systems, which were not discussed in previous studies. Based on the proposed set of measures, Section 3.4 illustrates the evaluation results of the existing Sinhala input systems. Finally, Section 3.5 gives an evidential proof for the validity of the user-friendliness measure.

3.1 Perspectives for Evaluation

In the field of human interface, *usability* has been a quite important concept not only in the design but also in the evaluation of systems. Roughly speaking, usability is a qualitative measure of “ease-of-use”; it tries to assess how easy a user can use the system at hand. Nielsen [52] presents the dimensions of usability as follows.

- Learnability

- Efficiency

- Memorability
- Errors
- Satisfaction

As text input systems form a class of human interface system, the evaluation perspectives should appreciate these dimensions.

Traditionally text input systems have been evaluated primarily focusing on input speed and the correctness; these are obviously associated with *Efficiency* and *Errors* in the Nielsen's list. We will review the existing quantified measures for efficiency, and propose their improvement in this chapter.

The remaining dimensions, *Learnability*, *Memorability*, and *Satisfaction*, in the list should also be considered in evaluation. In other words, in the context of this research, a Sinhala input system should be learnable, memorable, and satisfactory. As the degree of satisfaction cannot be directly measured, we will focus on the learnability and the memorability. With regard to these dimensions, the required/acceptable input key sequences by an input system are relevant, hence should be considered in the evaluation. Then how can be the input key sequences learnable and/or memorable? We assume that if the key sequences are regulated by some general rules, they are learnable. We also suppose that if the key sequences are intuitive, they are memorable; or more precisely, they are free from remembering. In this research, we use the term *user-friendliness* to denote these two dimensions. As will be discussed in this chapter, we propose a measure for the user-intuitiveness of input key sequences, given the situation where ways of standardized transliteration for Sinhala are not present.

3.2 Existing Measures for Text Input Efficiency

There are several measures used to evaluate efficiency of input systems. Speed is a very important aspect among them. Since early stages of typewriters, typing speed is used to compare the speed of each input system. For example, Masui measured the average input time that was necessary to input a prepared text with 53 characters [53].

Table 3.1: Focus of Attention (FOA)

Task	Direct Input		Predictive Input	
	Expert	Novice User	Expert	Novice User
Text creation task	0	1	1	2
text copying task	1	2	2	3

3.2.1 Entry Rates

Calculating the entry rate using Words per Minute (WPM) [54] is most widely used. Here, a word is standardized to 5 characters.

Words per Minute (WPM)

Words per Minute (WPM) is calculated as follows:

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5}, \tag{3.1}$$

where

S = time taken to enter text in seconds and

$|T|$ = number of characters in the text.

However, depending on the task that the test subjects are requested to perform, the entry rates vary. Text entry tasks can be classified into either text creation or text copying. These task types require different number of *focus of attentions (FOA)* [55] as shown in Table 3.1. The number of FOA is the number of places where the user has to keep his/her eyes on. Always the number of FOA of experts is lower by one because they can type without looking at the keyboard. Predictive input systems increase the number of FOA by one, as the users have to look at the display to confirm the candidate. text copying task always has one additional FOA, as it requires to look at the original text. Always high number of FOA depresses the WPM.

Even though text creation task has less FOA and it mimics typical usage, there are several problems why the researchers prefer the text copying task [56]. The problems of text creating task are: test subjects have to spend time wondering “What should I enter next?,” it is difficult to identify errors, it loses the control over the distributions of words.

There are several ways to reduce the number of FOA of the task by making the user to avoid seeing the manuscripts. Some researchers dictate the original text through an audio channel [57]. Some others [58]–[60] force the test subject to memorize the original phrase before starting to type, by hiding the original text as soon as the test subject starts to type.

The next important deal is how to handle the error factor. In text copying task, the target output is defined at the beginning. Therefore, some typing speed measuring systems do not accept any incorrect input sequences [61]–[64]. In contrast, some researchers do not allow any error corrections [65, 66]. Both of these two extremes do not reflect the real data entry process. Thus, the *unconstrained text entry evaluation paradigm* [67]–[69] is said to be a fare procedure to handle the error rates. In such a case, *Adjusted Words per Minute (AdjWPM)* [70] can be used to evaluate the system.

Adjusted Words per Minute (AdjWPM)

Adjusted Words per Minute (AdjWPM) can be defined as follows:

$$AdjWPM \quad = \quad WPM \times (1-U)^a, \tag{3.2}$$

where

$$\begin{aligned} WPM &= \text{Words per Minute,} \\ U &= \text{uncorrected error rate, and} \\ a &= \text{penalty exponent, usually set to one.} \end{aligned}$$

Because of the arbitrary nature of this measure, some researchers force the test subjects to correct all the errors [71].

In order to compare performances of two or more input systems, practically WPM or AdjWPM measures are not very suitable. The reason is one test subject may be familiar with one input system but not with the others. In such a situation the researchers have to find a quite big number of subjects for each input system who are familiar with it. It is quite difficult to fulfill this requirement especially when we consider a language like Sinhala; the number of users who have sufficient experience with a particular input system is very limited. In such a situation a theoretical measure of efficiency is required.

3.2.2 Efficiency

When we consider a direct input system such as the use of QWERTY for Roman characters, one keystroke produces one character. Therefore, the only factor that influences the typing speed is the physical arrangement of the keys in the keyboard. Dominic et al. [72] proposed a method for predicting maximum typing speeds with such a key arrangement. Their focus was on the prediction of typing speeds that can reduce the number of actual measurements.

On the other hand, in an input system for a language with many characters, we need a conversion process that maps the input key sequence into a linguistic expression in some representation form in the target language. A typical example of such a method is Japanese *romaji nyuryoku*, with which we get Hiragana characters by inputting the associated transliteration. The efficiency of such a conversion system, can be calculated using the measure: *Keystrokes per Character (KSPC)* [73, 74].

Keystrokes per Character (KSPC)

Keystrokes per Character (KSPC) is defined as follows:

$$KSPC = \sum_{i=1}^{N} P(C_i) \times |K_{C_i}|, \tag{3.3}$$

where

$C_{1..N}$	$\in$	complete character set of a specific language,
$P(C_i)$	$=$	occurrence probability of character C_i, and
$\|K_{C_i}\|$	$=$	number of keystrokes required to input character C_i.

3.3 Proposal of Efficiency and User-friendliness Measures

In this section we propose measures for evaluating the efficiency and the user-friendliness of input systems. For the efficiency measure, we modified KSPC which was given in the previous section. On the other hand, for the user-friendliness measure, we propose a novel method to assess the user-intuitiveness of the input sequence based on edit-distance.

3.3.1 Efficiency

As explained in Section 3.2, the most general way to calculate efficiency is to experimentally compute the maximum typing speed for each input system. However, the method experimentally measures the maximum typing speed is not applicable to Sinhala by the following reasons, suggesting that we need some theoretical measure.

- Sinhala has hundreds of characters with very low occurrence probabilities. Thus, it is not appropriate to take a short paragraph for experimentally calculating the efficiency.
- At most, the novice Sinhala computer users are used to type Sinhala based on only an input system that is his/her preference. Therefore, the experimental results will be innately biased.
- However, since the input sequences of the existing input systems are quite far from the intuition of average Sinhala computer users, it remains difficult to train people to type Sinhala using all the existing input systems for evaluation.

Hence, instead of the actual typing speed we use *typing_cost* which revises *Keystrokes per Character (KSPC)* introduced in the previous subsection. Note that KSPC is a theoretical measure which considers character occurrence probabilities.

In Sinhala, we cannot make use of KSPC as it is, where every key is equally considered. However, as exemplified in Table 2.18 and 2.19, and Figure 2.13, these existing systems force the user to frequently use shifted keys; the non-conversion direct input systems can not be implemented without using the shifted keys as Sinhala has a large set of characters (or character components). The use of shifted key might reduce the efficiency. Therefore we modify KSPC so as to incorporate weights for key classes as shown in Equation (3.4), and experimentally decide the weights. Note that the proposed measure is basically a theoretical measure, yet reflects a nature of actual use which has to be experimentally determined.

$$typing\_cost = \sum_{i=1}^{N} P(c_i) \times (|K_{C_i}| + w_s \times S(K_{C_i}) + w_r \times R(K_{C_i})), \qquad (3.4)$$

$$w_s \quad \text{(weight of shift key)} \quad = \frac{t_{xY} + t_{Xy}}{t_{xy}} - 2, \tag{3.5}$$

$$w_r \quad \text{(weight of repeated key)} \quad = \frac{t_{xx}}{t_{xy}} - 1, \tag{3.6}$$

where

$C_{1..N}$	$\in$	complete character set of a specific language,
$P(C_i)$	$=$	occurrence probability of character C_i,
K_{C_i}	$=$	key sequence require to input character C_i,
$\lvert K_{C_i}\rvert$	$=$	length of K_{C_i},
$S(K_{C_i})$	$=$	number of shift key used in K_{C_i},
$R(K_{C_i})$	$=$	number of repeated key strokes in K_{C_i},
t_{xy}	$=$	average time lapse between two unshifted keystrokes,
t_{xx}	$=$	average time lapse to repeat an unshifted keystroke,
t_{xY}	$=$	average time lapse between unshifted and shifted keys, and
t_{Xy}	$=$	average time lapse between shifted and unshifted keys.

Using this notation, Sinhala typing speed and keystroke typing speed can be defined by using Equations (3.7) and (3.8):

$$\text{keystroke typing speed} = \frac{1}{t_{xy}}, \tag{3.7}$$

$$\text{Sinhala typing speed} = \frac{\text{key stroke typing speed}}{\mathit{typing\_cost}}. \tag{3.8}$$

Experiments 1 and 2 are carried out in order to calculate the weights of shifted keys and repeated keys.

Experiment 1

Test subjects are asked to type a set of character pairs. Some pairs consist of two different characters, and the others consist of two same characters. Then t_{xy} and t_{xx} are calculated by averaging them. This experiment was carried out on a group of 12 subjects (3 females and 9 males, age 18-46 years).

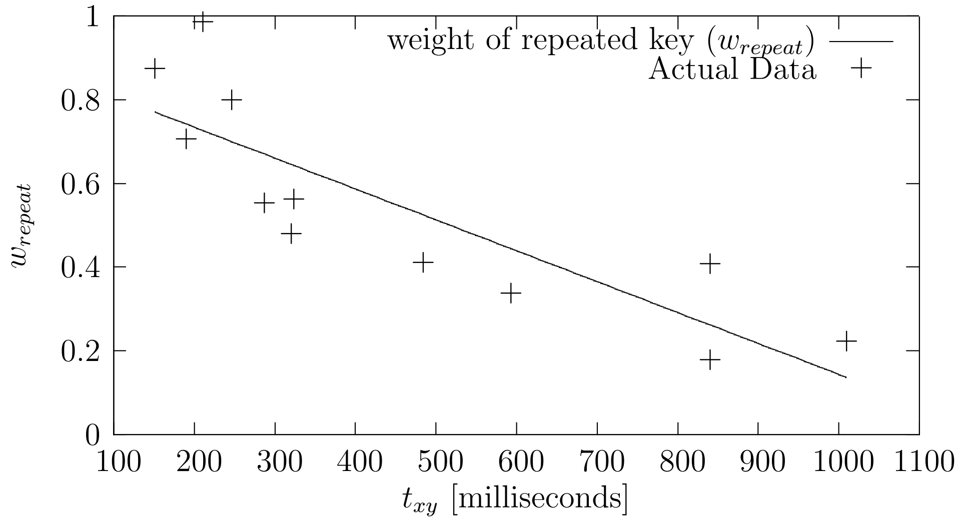


Figure 3.1: Weight of Repeated Keys

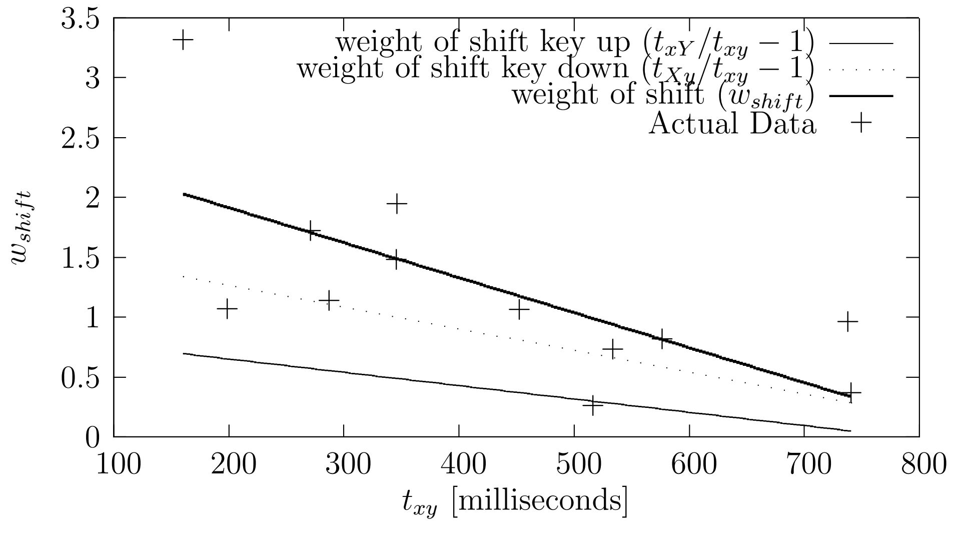


Figure 3.2: Weight of Shift Key

Experiment 2

The test subjects are asked to type a set of common English words, but some characters of the word are designated to use uppercase letters. Then t_{xy}, t_{xY} and t_{Xy} are calculated by averaging them. This experiment was carried out on a group of 11 subjects (7 females and 4 males, age 20-31 years).

Least Square Method

The trend of the above experiment data is estimated using the least square method. The trend is approximated into a line: Equation (3.9). b and m are calculated, which minimize the $\sum(y - actual\_data)^2$ [75, 76].

$$y = mx + b \tag{3.9}$$

$$m = \frac{\sum(x-\overline{x})(y-\overline{y})}{\sum(x-\overline{x})} \tag{3.10}$$

$$b = \overline{y} - m\overline{x} \tag{3.11}$$

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}} \tag{3.12}$$

The experimental results are shown in Figures 3.1 and 3.2. The X-axis shows t_{xy}, the average time lapse between two Roman character key strokes, while the Y-axis shows the weights of repeated keys and the shift key.

The equations of the approximation lines and the correlation coefficients are shown in Equations (3.13) and (3.14).

$$w_{repeat} = 0.87 - 0.73t_{xy}(|r| = 85\%) \tag{3.13}$$

$$w_{shift} = 2.50 - 2.92t_{xy}(|r| = 69\%) \tag{3.14}$$

Then, the Divaina online Sinhala newspaper [77] from January 2005 to May 2006 (about 50MB of *Kaputadotcom* encoded text) was used as a corpus to calculate the occurrence probabilities of each Sinhala character. Table 3.2 and Figures 3.3 and 3.4 show the probability distribution of Sinhala characters. Appendix A gives a more detailed list of it.

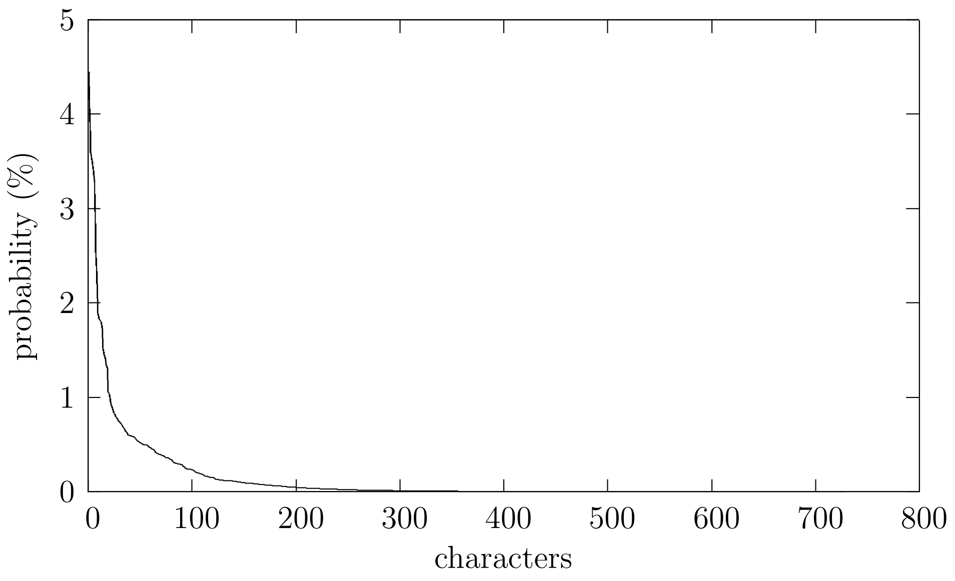


Figure 3.3: Occurrence Probabilities of Characters

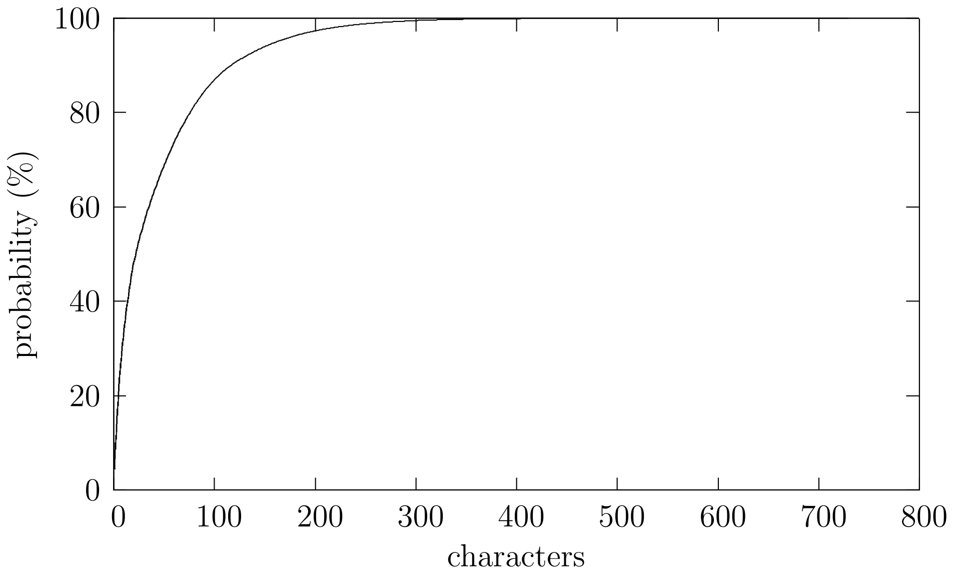


Figure 3.4: Accumulated Probability

Table 3.2: Occurrence Probabilities of Sinhala Characters

#	character	occurrence probability	accumulated probability
1	ය (=ya)	4.44%	4.44%
2	ව (=va)	4.15%	8.60%
3	න් (=n)	3.59%	12.19%
4	ම (=ma)	3.55%	15.74%
5	ක (=ka)	3.47%	19.21%
6	න (=na)	3.35%	22.56%
7	ර (=ra)	3.28%	25.83%
8	ට (=ṭa)	2.53%	28.36%
9	ස (=sa)	2.20%	30.56%
⋮	⋮	⋮	⋮

3.3.2 User-friendliness

As discussed in Section 3.1, we use the term user-friendliness as it indicates how easily a user with ordinary background can make use of a system. It turns out that, while considering text input systems, the acceptable input key sequences are crucial. That is, if a user is forced to use an unintuitive key sequence to input a text, the system is not user-friendly. On the contrary, if a system can accept a reasonable variety of intuitive key sequences, the system is user-friendly.

For example, in Japanese text input, there is no difficulty in inputting Japanese using Roman character key sequences because there is a set of well-known conversion rules for transliterating Japanese. In this regard, Japanese input systems are user-friendly. In India also, there are transliteration systems such as “baraha,” making the conversion-based input system more popular than the non-conversion input systems that force its user to use unintuitive key sequences.

In Sinhala, such a standardized transliteration scheme does not exist; a Sinhala text can be variously transliterated by Roman character sequences depending on the user. Therefore to assess the user-friendliness of a Sinhala input system, we need to have a user-friendliness measure that can evaluate how one of the acceptable key sequences is similar to the actual user key sequence that is generated from a user intuition. Note here that the measures discussed in Section 3.2 are introduced mainly for assessing the efficiency, but not for the user-friendliness as we need here.

Here, we propose to use the average edit distance between the input key sequences of each input system and the user intuitive key sequence, as a measure of the "user-friendliness," as shown in Equation (3.15):

$$avg\_edit\_dist = \frac{1}{M}\sum_{j=1}^{M}\sum_{i=i}^{N} P(C_i)edit\_dist(U_{S_jC_i}, K_{C_i}), \tag{3.15}$$

where

$C_{1..N}$	$\in$	Sinhala characters,
$P(C_i)$	$=$	occurrence probability of character C_i,
$S_{1..M}$	$\in$	test subjects,
$U_{S_jC_i}$	$=$	test subject S_j's intuitive transliteration of character C_i, and
K_{C_i}	$=$	input key sequence assigned for character C_i in a given input system.

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 [78]. Table 3.3 shows an example of edit distance calculation using the word "ආයුබෝවන්"(=āyubovan: welcome). Let's say one user intuitive key sequence to input "ආයුබෝවන්" is "ayubovan." Using *Kaputadotcom* keyboard layout "ආයුබෝවන්" can be typed as "a`yE@b`~vn~." The example shows that the edit distance between these two key sequences will be 7: 4 insertions, 1 deletion, and 2 substitutions.

Transliteration Experiment

In order to find our user intuitive transliteration of each Sinhala character, 275 Sinhala characters were used, and this covers more than 99% of the characters occurred in the corpus, and all the characters have more than a 0.0155% occurrence probability.

In order to produce an experiment more natural for the test subjects, we used a word list that includes all 275 characters mentioned above, instead of using the characters separately. We tried to minimize the number of words in order to reduce the test subjects' load. However, the word list ended up with 106 words. The difference between the input

Table 3.3: Example of Calculating Edit Distance

a		y	u		b	o		v	a	n	
↓	I	↓	S	I	↓	S	I	↓	D	↓	I
a	`	y	E	@	b	`	˜	v		n	˜

Number of Insertions	(I)	=	4
Number of Deletions	(D)	=	1
Number of Substitutions	(S)	=	2
Edit Distance		=	7

sequences and test subjects' transliteration proposals is taken as a measure of how difficult it is to remember the input sequence for each Sinhala character.

Test subjects were asked to transliterate the Sinhala word list. This experiment was carried out on a group of 30 subjects between 14 to 60 years old, which included 14 males and 16 females. The transliterated word lists we got from the subjects were split into characters. Then the difference between the input key sequence of each input system and the proposed transliterations of each test subject was measured by the edit distance between the two strings.

3.4 Evaluation of the Existing Sinhala Input Systems

For our evaluation, the most popular Sinhala input systems, which are the *Wijesekara*, *Kaputadotcom* and *Natural SinGlish* explained in Section 2.3, have been taken into account.

3.4.1 Efficiency

The efficiencies of the existing Sinhala input systems are shown in Figure 3.5. They were calculated using the occurrence probabilities of each Sinhala character in the UCSC Sinhala Corpus BETA [79] provided by the University of Colombo. The X-axis shows the keystroke typing speed in keystrokes per minute, and the Y-axis shows the Sinhala typing speed in Sinhala characters per minute. These results indicate that the most efficient existing Sinhala input system is the *Wijesekara*. Table 3.4 shows the average typing cost

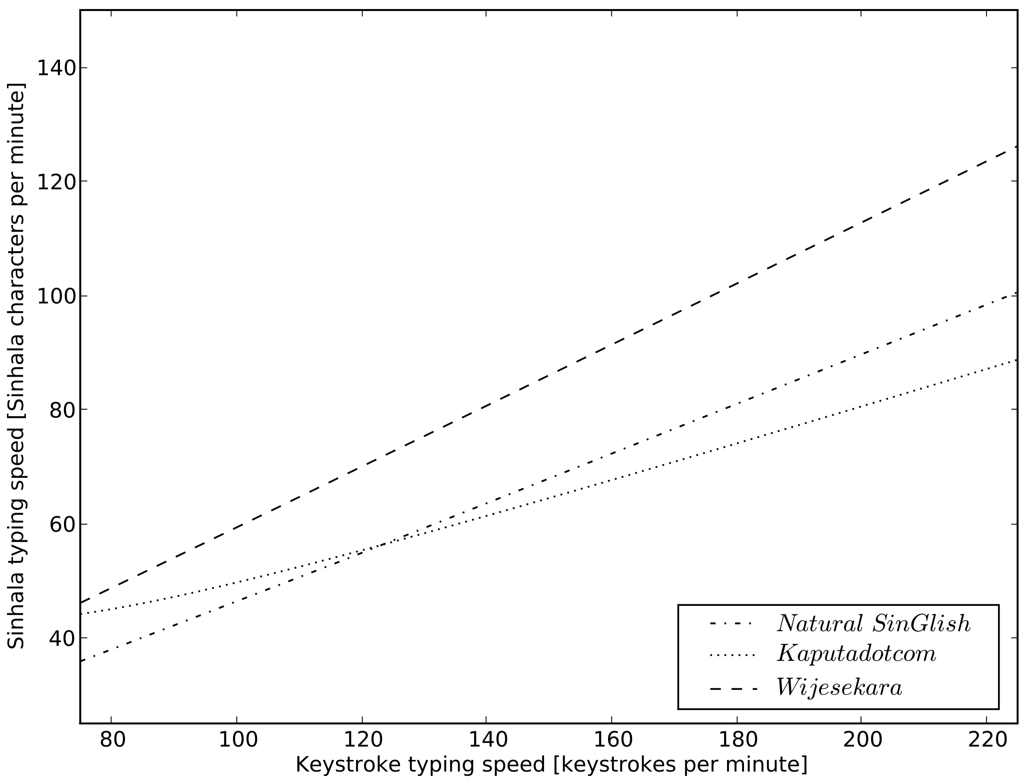


Figure 3.5: Efficiencies of Existing Sinhala Input systems

Table 3.4: Average Typing Cost

Input	Keystroke typing speed [Keystrokes per minute]						
System	75	100	125	150	175	200	225
Kaputadotcom	1.70	2.01	2.20	2.33	2.42	2.48	2.54
Natural SinGlish	2.09	2.16	2.19	2.21	2.22	2.23	2.24
Wijesekara	1.63	1.69	1.72	1.74	1.76	1.77	1.78

Table 3.5: Average Edit Distances

Input system	Average edit distance
Wijesekara	2.06
Kaputadotcom	1.43
Natural SinGlish	0.33

of each input system at different key-stroke typing speed levels.

3.4.2 User-friendliness

As a measurement of user-friendliness, we have calculated the average edit distance between an input key sequence and the proposed transliteration of each character. The average edit distances of each input system are calculated using Equation (3.15) and are shown in Table 3.5. The results show that there is a big difference between the subjects' transliteration proposals and the input sequence proposed by *Kaputadotcom* and *Wijesekara*. Even though *Natural SinGlish* significantly reduced the gap, it is not good enough for novice users because it forces the users to memorize a set of key assignments for entering Sinhala characters. According to the above results we can say that trade-off exists between efficiency and user-friendliness.

3.5 Validity of the User-friendliness Measure

In Japanese *romaji nyuryoku*, the edit distance between the user intuitive key sequence and the required input sequence is zero. This is because Japanese has a well know transliteration scheme as explained in Section 2.2. In this regard, any Japanese input systems are fully user-friendly, meaning that our notion of user-friendliness is not relevant.

On the other hand, in India, whose languages share same characteristics with Sinhala in the sense that there are no standardized transliteration, the input systems that accept more user intuitive key sequence as there input sequence are more popular. This may support that our claim that the user-friendliness, particularly for languages without standardized transliteration, can be measured by the edit distance-based measure.

However, it is not directly proven that our edit distance-based measure is sufficient

Table 3.6: Average Edit Distance vs. Test Subjects' Ratings

Input Systems	Average Edit Distance	Ratings by Test Subjects
Natural SinGlish	0.33	81.6
Kaputadotcom	1.43	32.4
Wijesekara	2.06	25.2
Correlation coefficient	r=-96.9%	

to fully assess the "user-friendliness" of a text input system. We therefore planned an experiment to validate the proposed measure.

3.5.1 Experiment

We conducted a survey in the form of a questionnaire (Appendix B) with 13 subjects (10 females and 3 males). In the questionnaire, we first asked them typing experiences in English or Sinhala. We then gave a simple Sinhala sentence with the key sequences used to input that sentence in each input system. We asked them first to study carefully the sentence and the key sequences required to type, and then to rate each input system from a viewpoint of වඩා පහසු ලිවීමේ ක්‍රමය ("easiest-to-input") on a scale of 1 to 100. Note that the notion of "easiest-to-input" is highly associated with the dimensions of the *usability*: *memorability* and *learnability*. Other dimensions are not relevant here; *efficiency*, *error*, and *satisfaction* should be directly measured by using some input system. On the other hand, *memorability* and *learnability* should be considered when the subjects answer the questionnaire. If the results from this experiment using the questionnaire correlate with the results from our edit distance-based measure, it means that our measure can be employed as a measure for assessing memorability and learnability, hence our notion of user-friendliness.

3.5.2 Experimental Results

Table 3.6 shows the average ratings by the test subjects and average edit distances. A visual representation of the data is given in Figure 3.6. As shown in the table, very high level of correlation (-96.9%) was measured. This proves that the degree of "easiest-to-input"

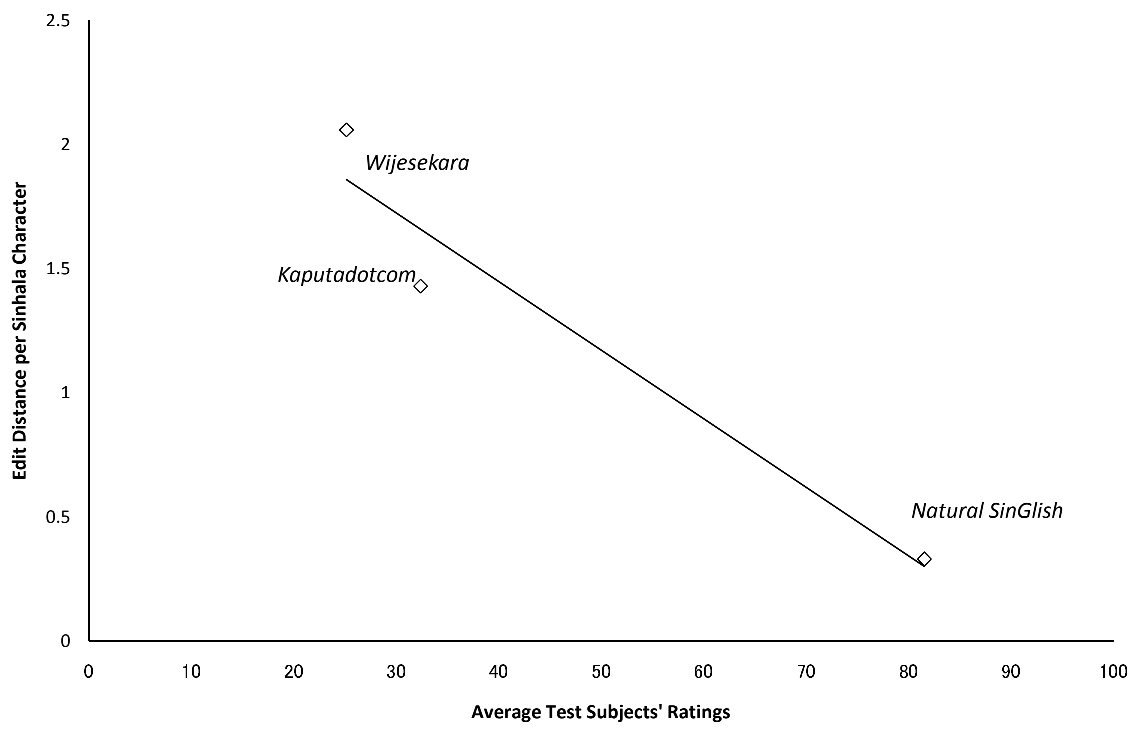


Figure 3.6: Average Edit Distance vs. Test Subjects' Ratings

highly correlates with the edit distance between the required and resulted sequences, suggesting that the user-friendliness of a text input system can be assessed by the proposed measure.

The typing experiences of the test subjects' did not make a significant difference on their ratings; the correlation coefficient between the experienced subjects' ratings and those of non-experienced subjects was 0.99. Therefore, the proposed user-friendliness measure is valid independent of the users' typing experiences.

3.6 Conclusion

In this chapter we have proposed a new methodology to evaluate Sinhala input systems. First we have discussed the general measures used to evaluate input systems. Text input systems should be evaluated not only by the efficiency but the user-friendliness, especially when the users are not professionals. The efficiency is quantified by the average typing cost per Sinhala character, while the user-friendliness is assessed by the average edit distance between a user-intuitive character sequence and the input sequences of an input system.

We have reported the evaluation results of existing Sinhala systems by employing these measures. We finally proved that the proposed user-friendly measure is valid to evaluate the user-friendliness through questionnaire based experiment.

Chapter 4

Sri Shell: Phonetically-principled Input System

The previous chapter argued that a text input system should be user-friendly, especially for non-professionals. One of the strategies to ensure the user-friendliness is to develop a key assignment which is intuitive or principle-based. In this chapter, we propose a phonetically-principled associative conversion-based direct input system. We also intend to implement such a principled system as a light-weighted application independent module that can be realized without any language resources such as corpora or dictionaries. The objective of this system is discussed in Section 4.1. System design is presented in Section 4.2. Overall architecture of the system is discussed in Section 4.3. Finally Section 4.4 concludes the evaluation results.

4.1 Objective

All existing Sinhala input systems use uppercases to cover the variety of characters. Among them *Wijesekara* uses uppercases for the less frequent characters. The other systems use uppercases for various characters. The use of uppercases can be problematic for three reasons. Firstly, the use of uppercases increase the users' load as discussed in Chapter 3. Secondly, assigning special role for shifted key should be avoided especially for English-familiar users; in English, uppercase letters are used for proper nouns and sentence beginnings, and do not exhibit any phonetical differences. Finally, a mixture of uppercases and lowercases symbols, results in an unreadable input sequence, for ex-

Table 4.1: Text Entry Example of *Sri Shell*

Output text	මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.
Input key sequence	*meya dxarannaata avahira baadxhaavalin txorava nidxahasee gaman kiriimata saha avasxyavana aadxhaara dxa aarakshaava dxa salasvaa dxena lesatx adxaala vagakiim dxarana siyalu dxenaa-genma sxrii la/nkaa prajaatxaantxrika samaajavaadxii janarajayee janaadxhipatxi illumkara dxa apeekshaakara dxa sitii.*

ample: "*kuruNA)gala*" of *Natural SinGlish*, and "*kOr#N$gl*" of *Kaputadotcom* (Section 2.3). One may argue that this is just an input system and there is no need for readability. However, if a sequence is readable it will be easier to memorize, and for an application like LaTeX where one has to type without any output feedback, it is an advantage if what is typed can be read. The Sinhala TeX Package [80] supports a transliteration scheme called *Samanala* [81] which uses uppercases and symbols, and hence is unreadable.

Even though *Natural SinGlish* is quite user-friendly, it relies too much on English spellings. So they tried to avoid key combinations which are very rare in English. Instead of being too dependent on English-like input sequences, we want to implement more systematic and efficient conversion system.

Here, our objective is to propose an efficient and user-friendly Sinhala input system based on principled phonetic notation, which uses only unshifted keys. Table 4.1 exemplifies the text input using *Sri Shell*. This key assignment what we propose here can also be used as a lossless transliteration scheme for Sinhala.

4.2 System Design

In this section we discuss the design principles of *Sri Shell*, and detail the phonetically-principled key assignment.

4.2.1 Principles of the Proposed System

The most prominent design principle is the phonetically-principled key assignment. It is based on the following three principles.

- It is based on the phonetic notation of characters:
  - All aspirated consonants can be produced by adding an “h” to unaspirated consonants.
  - Nasals can be produced by a voiceless vowel preceded by “/.”
  - Nasal+voiced can be produced by a voiced vowel preceded by “/.”

  (See Table 4.3)
- It is consistent:
  - All long-vowels can be produced by doubling the last character of a short-vowel. (See Tables 4.2)
  - If two Sinhala characters map to the same Roman character, then these Sinhala characters are differentiated by adding an “x” to the one with a lower occurrence probability.
    For example: retroflexes and dentals of Table 4.3.
- It is complete:
  Most of the existing Sinhala input systems have several missing characters. Such rare characters as ඍ, ඎ, ඏ, and ඐ are usually missing. *Sri Shell* supports all characters even though some cannot be displayed with most of the fonts.

4.2.2 Key Assignments

Sri Shell assigns a key combination to each Sinhala phoneme. The basis of this system is the phonetic notation of Sinhala characters. Based on the modern “Sammata Siṃhala Hōḍiya” (standard Sinhala alphabet: Table 2.4), Tables 4.2, 4.3 and 4.4 show the key assignment by *Sri Shell* for Sinhala vowels, occlusive consonants and the other consonants respectively with the phonetic notation using NLAC (*National Library at Calcutta Romanization*) [24], and using IPA (*International Phonetic Alphabet*) [25, 26]. The left most columns of these tables show *articulations* of the phonemes.

Table 4.2: Vowels, their Phonetic Notations (NLAC [IPA]) and their *Sri Shell* Key Assignments

	Sinhala Phoneme	අ	ඇ	ඉ	උ	ඍ	ඏ	එ	ඔ
Short	NLAC	a	æ	i	u	ṛ	ḷ	e	o
Vowels	IPA	[a]	[æ]	[i]	[u]	[r]	[l]	[e]	[o]
	Sri Shell	*a*	*ae*	*i*	*u*	*rx*	*lxx*	*e*	*o*
	Sinhala Phoneme	ආ	ඈ	ඊ	ඌ	ඎ	ඐ	ඒ	ඕ
Long	NLAC	ā	ǣ	ī	ū	ṝ	ḹ	ē	ō
Vowels	IPA	[aː]	[æː]	[iː]	[uː]	[rː]	[lː]	[eː]	[oː]
	Sri Shell	*aa*	*aee*	*ii*	*uu*	*rxx*	*lxxx*	*ee*	*oo*
	Sinhala Phoneme							ඓ	ඖ
Diphthongs	NLAC							ai	au
	IPA							[ai]	[au]
	Sri Shell							*ai*	*au*

Sinhala phoneme අ(=a) is normally pronounced as [a] as shown in Table 4.2. But when the phoneme අ(=a) is combined with a consonant, sometimes the pronunciation changes to [ə]. For example, ක(=ka) can either be pronounced as [ka] or [kə] depending its position in a word [82]. Similarly, Sinhala phoneme ආ(=ā) has two pronunciations, where කා(=kā) can be pronounced either as [kaː] or [ka], depending on the position. However, we do not assign two different key assignments for [ka] and [kə] etc., because the Sinhala characters are the same.

Sinhala has one conjunct consonant "ඥ" (=jñ [ɟɲ]), which represents ජ+ඤ as discussed in Section 2.1.1. For this special character, we have assigned a special key sequence: "*cx*."

4.3 Overall Architecture

Figure 4.1 illustrates the overall architecture of *Sri Shell* system. As soon as the user activates the *Sri Shell Controller*, it catches up all the keystrokes pressed by the user. Then the key sequence is converted to a Sinhala character sequence, and it is transmitted to the application program. Figure 4.1 shows a status where the user have already entered

Table 4.3: Occlusive Consonants, their Phonetic Notations (NLAC [IPA]) and their *Sri Shell* Key Assignments

		Unaspirated Voiceless	Aspirated Voiceless	Unaspirated Voiced	Aspirated Voiced	Nasal	Nasal+Voiced
Velars	Sinhala Phoneme	ක	ඛ	ග	ඝ	ඞ	ඟ
	NLAC	k	kh	g	gh	ṅ	ňg
	IPA	[k]	[k^h]	[g]	[g^h]	[ŋ]	[^{ŋ}g]
	Sri Shell	*k*	*kh*	*g*	*gh*	*/k*	*/g*
Palatals	Sinhala Phoneme	ච	ඡ	ජ	ඣ	ඤ	ඦ
	NLAC	c	ch	j	jh	ñ	ňj
	IPA	[c]	[c^h]	[ɟ]	[ɟ^h]	[ɲ]	[^{ɲ}ɟ]
	Sri Shell	*c*	*ch*	*j*	*jh*	*/c*	*/j*
Retroflexes	Sinhala Phoneme	ට	ඨ	ඩ	ඪ	ණ	ඬ
	NLAC	ṭ	ṭh	ḍ	ḍh	ṇ	ňḍ
	IPA	[ʈ]	[ʈ^h]	[ɖ]	[ɖ^h]	[ɳ]	[^{ɳ}ɖ]
	Sri Shell	*t*	*th*	*d*	*dh*	*nx*	*/d*
Dentals	Sinhala Phoneme	ත	ථ	ද	ධ	න	ඳ
	NLAC	t	th	d	dh	n	ňd
	IPA	[t]	[t^h]	[d]	[d^h]	[n]	[^{n}d]
	Sri Shell	*tx*	*txh*	*dx*	*dxh*	*n*	*/dx*
Labials	Sinhala Phoneme	ප	ඵ	බ	භ	ම	ඹ
	NLAC	p	ph	b	bh	m	m̌b
	IPA	[p]	[p^h]	[b]	[b^h]	[m]	[^{m}b]
	Sri Shell	*p*	*ph*	*b*	*bh*	*m*	*/b*

Table 4.4: Other Consonants, their Phonetic Notations (NLAC [IPA]) and their *Sri Shell* Key Assignments

Approximants	Sinhala Phoneme	ය්	ර්	ල්	ළ්	ව්
	NLAC	y	r	l	ḷ	v
	IPA	[j]	[r]	[l]	[ɭ]	[ʋ]
	Sri Shell	*y*	*r*	*l*	*lx*	*v*
Fricatives	Sinhala Phoneme	ශ්	ෂ්	ස්	හ්	
	NLAC	ś	ṣ	s	h	
	IPA	[ɕ]	[ʂ]	[s]	[h]	
	Sri Shell	*sx*	*sh*	*s*	*h*	
Anusvara and Visarga	Sinhala Phoneme	ං	ඃ			
	NLAC	ṃ	ḥ			
	IPA	[ŋ]	[h]			
	Sri Shell	*/n*	*hx*			

three key strokes: $\boxed{\text{A}}$, $\boxed{\text{A}}$, and $\boxed{\text{Y}}$, where the input sequence is "*aay.*" Then this input sequence was converted into "ආය්"(=āy)(=U+0D86 U+0DBA U+0DCA) by the *Sri Shell Converter*, and then it had been transmitted to the application program.

When the user presses the next key: $\boxed{\text{U}}$, the input sequence is updated to "*aayu.*" After converting this input sequence to Sinhala characters: "ආයු"(=āyu)(=U+0D86 U+0DBA U+0DD4), the controller identifies the differences between the two Sinhala character sequences: "U+0D86 U+0DBA U+0DCA" and "U+0D86 U+0DBA U+0DD4." Then the controller send two signals to the application program: (1) to delete the character "U+0DCA," and (2) to insert the character "U+0DD4."

4.4 Evaluation

Using the proposed methodologies in Section 3.3, we have evaluated our proposed input system: *Sri Shell.* The evaluation results obtained using the two measures: user-friendliness and efficiency, are given below.

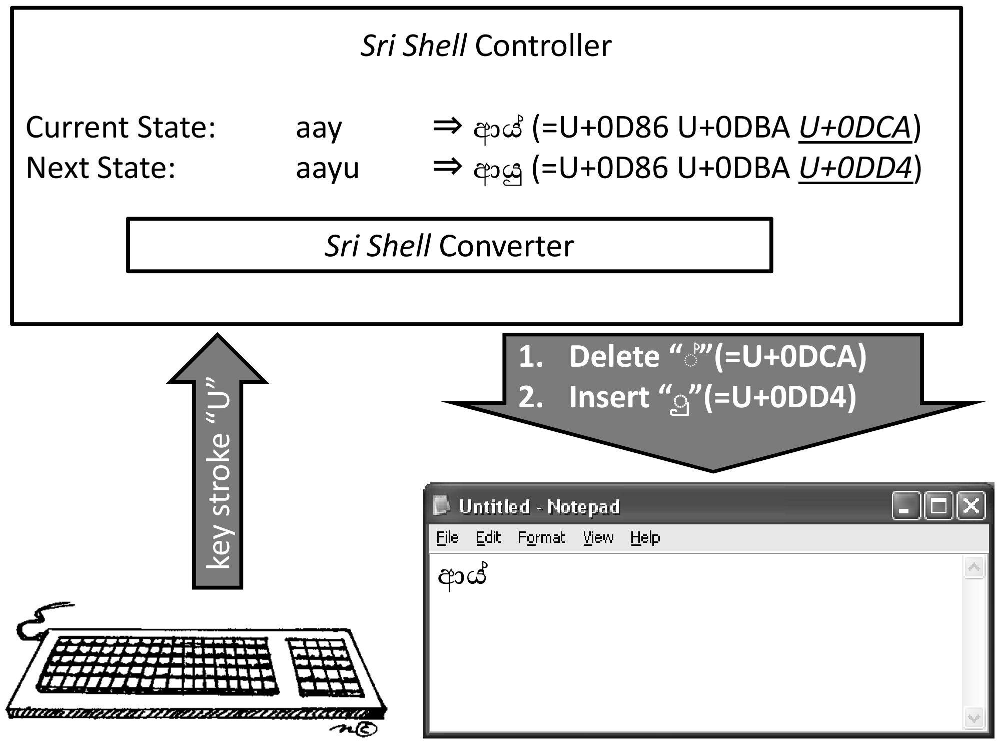

Sri Shell Controller
Current State: aay ⇒ ආය් (=U+0D86 U+0DBA *U+0DCA*)
Next State: aayu ⇒ ආයු (=U+0D86 U+0DBA *U+0DD4*)
Sri Shell Converter
key stroke "U"
1. Delete "්"(=U+0DCA)
2. Insert "ු"(=U+0DD4)
Untitled - Notepad
File Edit Format View Help
ආය්

Figure 4.1: System Architecture of *Sri Shell*

4.4.1 User-friendliness

Using Equation (3.15), we have calculated the average edit distance of *Sri Shell* and other existing Sinhala input systems. These value are calculated based on the occurrence probabilities derived from UCSC Sinhala Corpus BETA [79] provided by the University of Colombo. Table 4.5 shows the evaluation results. The results show a big difference between the user intuitive character sequence and the input sequence of *Wijesekara* and *Kaputadotcom* keyboard layouts. However, *Natural SinGlish* and *Sri Shell* were able to reduce this gap significantly. This was possible because the conversion systems have been designed to accept more user intuitive key sequences. The results also show that, *Natural SinGlish* is slightly more user-friendly than *Sri Shell.* This happened because the test subjects always tried to produce a transliterated Sinhala word that resembles an English word.

Even though *Natural SinGlish* and *Sri Shell* were able to improve the user-friendliness significantly, still they are not good enough for novice users because they force the users

Table 4.5: Average Edit Distances

input system	Average edit distance
Wijesekara	2.06
Kaputadotcom	1.43
Sri Shell	0.43
Natural SinGlish	0.33

Table 4.6: Average Typing Cost

Input	Keystroke typing speed [Keystrokes per minute]						
System	75	100	125	150	175	200	225
Kaputadotcom	1.70	2.01	2.20	2.33	2.42	2.48	2.54
Natural SinGlish	2.09	2.16	2.19	2.21	2.22	2.23	2.24
Wijesekara	1.63	1.69	1.72	1.74	1.76	1.77	1.78
Sri Shell	2.11	2.13	2.14	2.15	2.16	2.16	2.16

to memorize a set of key assignments for entering Sinhala characters. Therefore, an input system that accepts all user intuitive key sequences as input sequences, is greatly anticipated.

4.4.2 Efficiency

The efficiency of the proposed system is evaluated using average typing cost defined in Equation (3.4). The evaluation results of *Sri Shell* and other existing input systems, which are calculated based on the occurrence probabilities of the UCSC Sinhala Corpus BETA [79], are shown in Figure 4.2 and Table 4.6. The X-axis shows the keystroke typing speed in keystrokes per minute, and the Y-axis shows the Sinhala typing speed in Sinhala characters per minute. These results proved that the proposed input system *Sri Shell* gives the second highest efficiency in most cases with a considerably high level of user-friendliness, where *Sri Shell* has the second lowest typing cost among the four input systems. We can also say that *Sri Shell* is the most efficient conversion based Sinhala input system. Note that, the

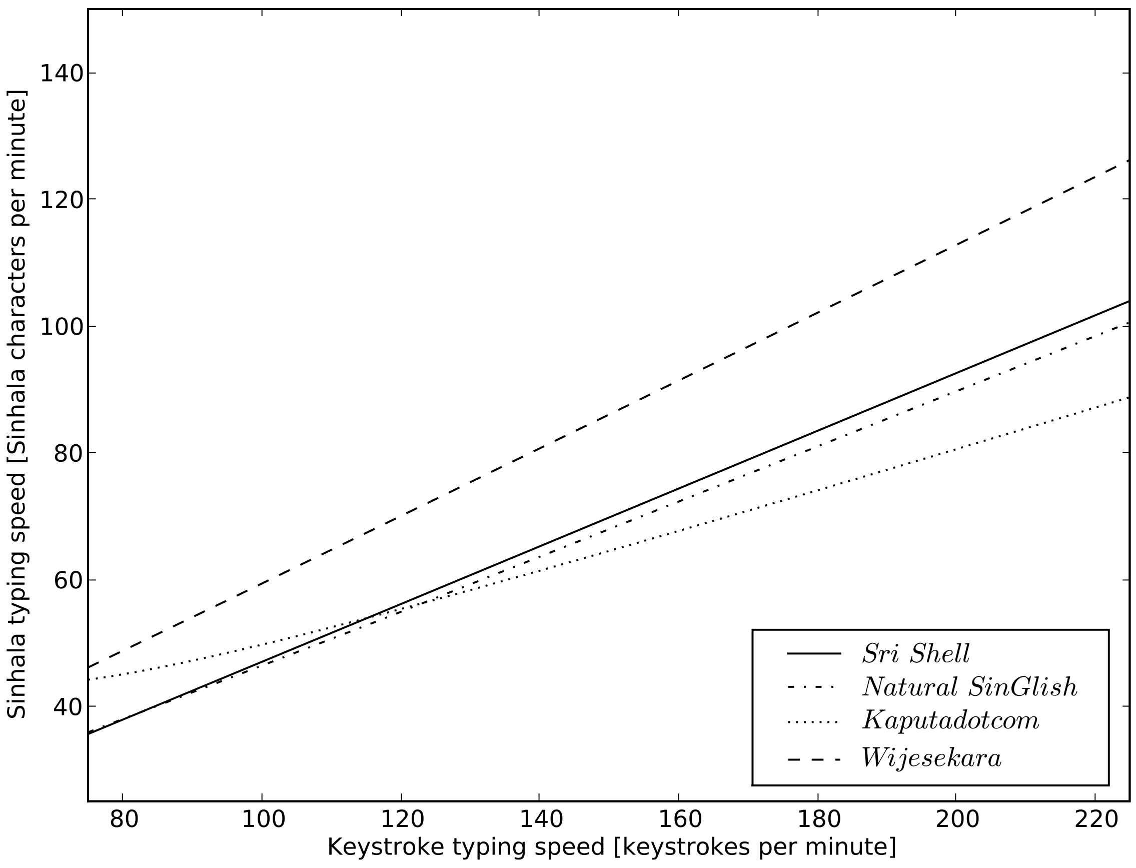


Figure 4.2: Efficiency of *Sri Shell*

most efficient two input systems use less number of shifted key strokes, where *Wijesekara* use them for less frequent character components and *Sri Shell* does not use any shifted key strokes. However, the non-conversion system *Wijesekara*, exhibits high efficiency, compared to *Sri Shell* conversion systems, even though the conversion systems are more user-friendly. According to the above results we can say that trade-off exists between efficiency and user-friendliness.

4.4.3 Overall Assessment

We have proposed a conversion-based phonetically associative direct input system, which is modestly user-friendly and efficient. As this system does not use any word lists, it can be implemented on a device which has a limited resources, such as a mobile phone. As this system is a direct input system, the users do not have to select candidates from a

Table 4.7: Example of *Sri Shell* LaTeX Converter

```
...
{\sinhala sxrii la/nkaa prajaatxaantxrika samaajavaadxii janarajaya}\\
{\NLAC sxrii la/nkaa prajaatxaantxrika samaajavaadxii janarajaya}\\
{\IPA sxrii la/nkaa prajaatxaantxrika samaajavaadxii janarajaya}\\
...
```

...

ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජය

śrī laṃkā prajātāntrika samājavādī janarajaya

ɕriː laŋkaː praɟaːtaːntrika samaːɟaʋaːdiː ɟanaraɟaja

...

menu. Additionally it does not use any shifted keystrokes. These two factors enable the users to improve their typing speeds more. Furthermore, *Sri Shell* is a complete input system; all Sinhala characters can be input correctly. Therefore, it can be used to input any Sinhala word in any technical field. *Sri Shell* can also be utilized with any *Pāli* or *Sanskrit* word.

Sinhala has many more phonemes compared to the number of Roman characters. As a result, there is no standard lossless transliteration scheme can be employed to transliterate Sinhala into Roman characters. On the other hand, English characters and English pronunciations are in many-to-many relationships. Therefore, Roman character transliteration of Sinhala is necessarily ambiguous. *Sri Shell* does not allow any of these ambiguities, as it is a direct input system. Thus, to realize a user-friendly input system which can handle the ambiguities, we need to realize a predictive input system.

Sri Shell's key assignment can also be used as an independent Sinhala transliterating scheme, which is highly readable. Table 4.7 shows such an example, where *Sri Shell* transliteration scheme is used to write Sinhala characters and Sinhala phonetics in LaTeX.

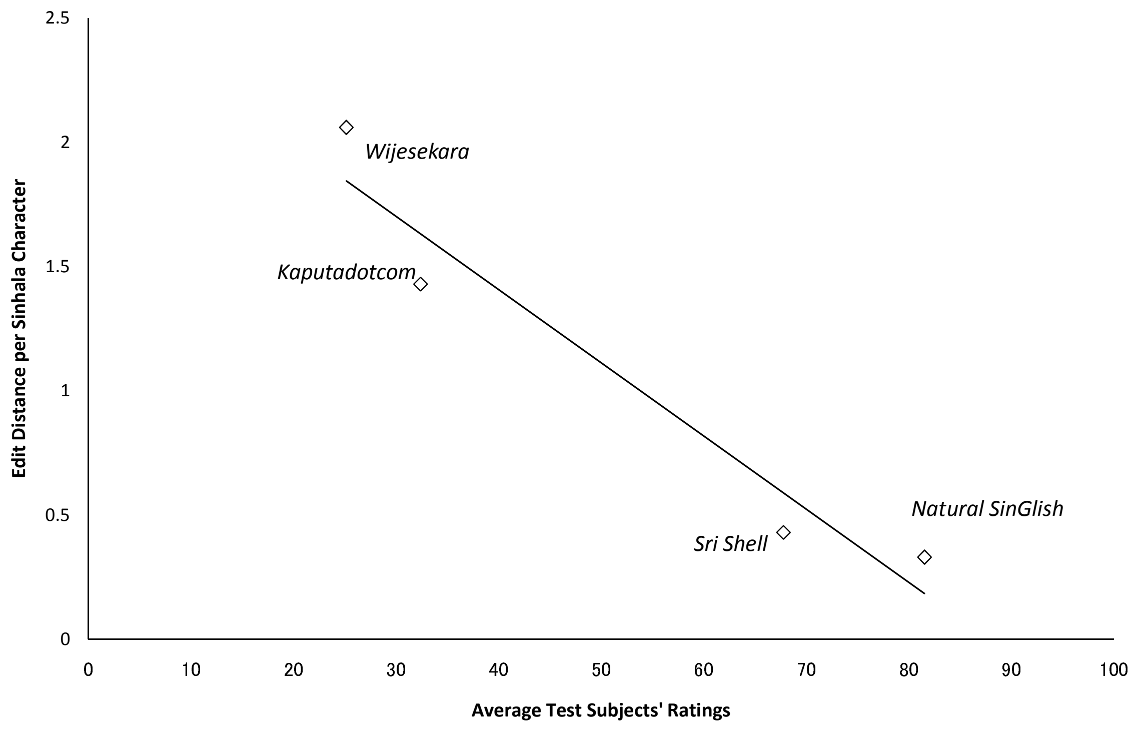


Figure 4.3: Average Edit Distance vs. Test Subjects' Ratings

4.4.4 Validity of the User-friendliness Measure

In Section 3.5, we discussed the validity of our user-friendliness measure. We have extended the results by incorporating the user-friendliness evaluation results of *Sri Shell.* Table 4.8 and Figure 4.3 summarize the results. Again, the user assessment and the edit distance value are highly correlated; the correlation coefficient is -96.7%.

4.5 Conclusion

In this chapter, we have proposed a phonetically-principled associative conversion-based direct input system called *Sri Shell.* The system is a light-weighted application independent module that can be realized without any language resources such as corpora or dictionaries. This *Sri Shell* system is moderately user-friendly while maintaining better level of efficiency comparing to other conversion-based direct input systems. This system was available freely online, where hundreds of Sinhala speakers downloaded and enjoyed

Table 4.8: Average Edit Distance vs. Test Subjects' Ratings

Input Systems	Average Edit Distance	Ratings by Test Subjects
Natural SinGlish	0.33	81.6
Kaputadotcom	1.43	32.4
Wijesekara	2.06	25.2
Sri Shell	0.43	67.8
Correlation coefficient	r=-96.7%	

it. Among them, some commented that the system was convenient to use, because it operated application-independent, where all other existing conversion based input systems at the time were applications by themselves. It also should be noted that *Sri Shell* is a complete input system that can be utilized in combination with the next proposed system *SriShell Primo* in Chapter 5.

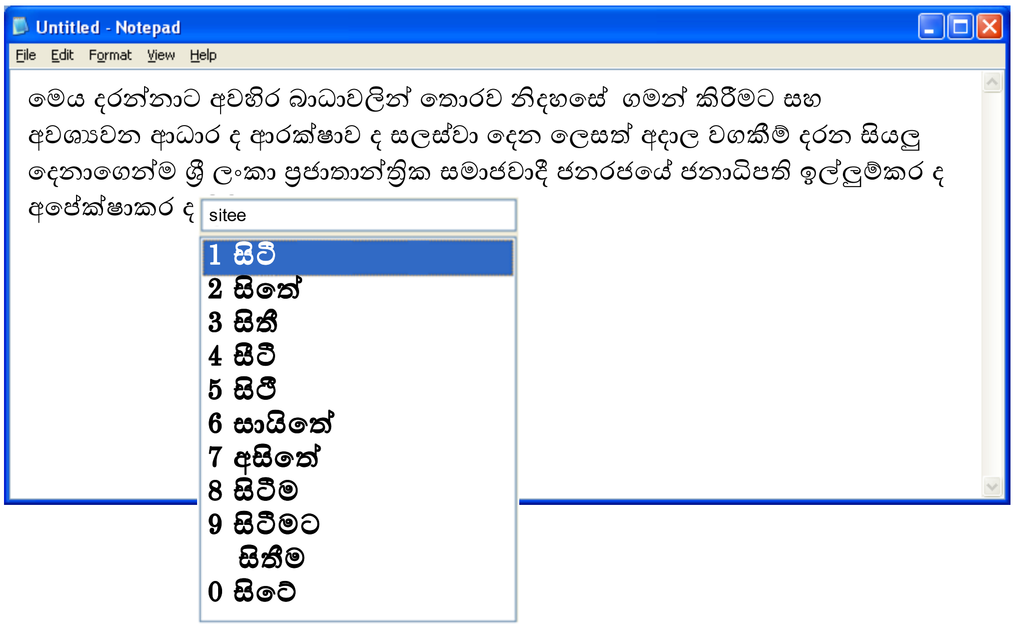
Chapter 5

SriShell Primo: Word-based Predictive Input System

The previous section argued that a Sinhala input system can be more user-friendly if it accepts a variety of user intuitive transliterations. To address this issue, we have incorporated a device called *input variation table* which lists possible user intuitive input variations. The introduction of this device however calls for the system to realize a mechanism to choose the best Sinhala character sequence toward the given user input sequence. We therefore propose a word-based predictive method to narrow down the ambiguities. Figure 5.1 shows a screenshot of text input using *SriShell Primo*, where the system lists predicted word candidates. The prediction is made based on the input variation table and the probabilistic mechanism to rank the candidates. This word-based method is also beneficial, as it can propose completion candidates during the input process. The objective of the proposed system is summarized in Section 5.1. System design is discussed in Section 5.2. Overall architecture, including the input variation table, language resources required, and the computational process, is detailed in Section 5.3. Section 5.4 discusses the implementation issues, and finally Section 5.5 concludes the evaluation results.

5.1 Objective

In Section 3.3 we have carried out an experiment to find out how the general Sinhala speakers transliterate Sinhala into Roman characters. There we have found out that the



මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ
අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු
දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද
අපේක්ෂාකර ද

sitee

1 සිටී
2 සිතේ
3 සිතී
4 සීටී
5 සිඨී
6 සායිතේ
7 අසිතේ
8 සිටීම
9 සිටීමට
සිතීම
0 සිටේ

Figure 5.1: Screenshot of *SriShell Primo*

Roman character transliteration of Sinhala is ambiguous. A few examples are shown in Figure 5.2.

There are two reasons for Roman character transliteration of Sinhala to be ambiguous.

1. Sinhala has many more phonemes, compared to the number of Roman characters. For example Sinhala has 18 vowel characters, where Roman characters have only 5 vowel characters.

2. English spelling itself is ambiguous in the sense that there are no direct correspondences between them and the pronunciations, especially in phoneme level.

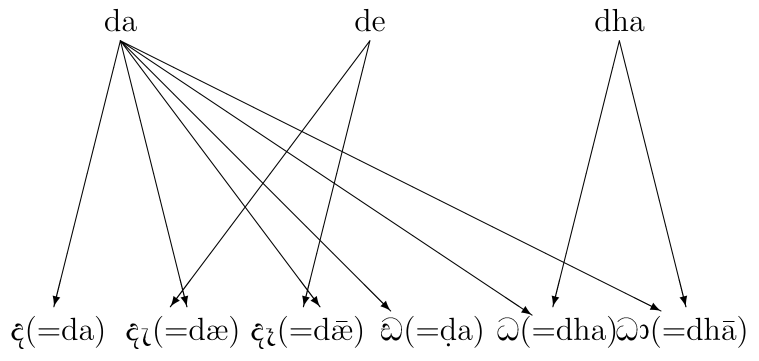


Figure 5.2: Some Many-to-many Relationships in Test Subjects' Proposals

In order to develop a fully “user-friendly” system, we have to accommodate the transliteration variations coming from the above discussed reasons. We have incorporated a device called *input variation table* to address the problem. The input variation table lists Sinhala phonemes and the corresponding input sequences. This table enables a user to input Sinhala text by using intuitive key sequences. This however introduces a problem of ambiguities; a key sequence is associated with a number of possible Sinhala character candidates, and the user has to choose among them. As the number of candidates can be very large, it is required that the system provides some mechanism to narrow down and rank the candidates. To address this technical issue, we apply a *word-based probabilistic language model* to filter out useless candidates and rank them appropriately. In general, being more user-friendly means being less efficient. We however try to achieve a high efficiency which is comparable to non-conversion direct input systems such as: *Wijesekara*, *Kaputadotcom* by introducing a *vowel omission* function.

5.2 System Design

We propose a Sinhala input system called *SriShell Primo*, which is a word-based predictive converter. A number of predictive input systems have been proposed so far, especially for handheld devices and mobile phones [83]. Among them, eZiText(R) [84] supports such Indic scripts as Hindi, Tamil, and Malayalam. A *SriShell Primo* user can input a Sinhala word by typing it as a sequence of Roman characters that they think is the most appropriate. Even though the Roman character sequence for a specific Sinhala word may differ from person to person, *SriShell Primo* is still capable of predicting the intended Sinhala word. Users can select the intended word from the candidate list.

Table 5.1 shows a text entry example of *SriShell Primo*. Here a novice user may use a key sequence that is intuitive for him/her. In this case for some input sequences, the user may not get his/her intended word as the topmost word in the menu. In this example user has selected the second menu item by pressing the numeric key: $\boxed{2}$, after “*darannata*” and “*siti*.” On the other hand a *SriShell Primo* expert is able to enter the same text with less number of keystrokes. Furthermore he/she also tries to use a key sequence with less ambiguity, which gives his/her intended word as the topmost candidate of the menu.

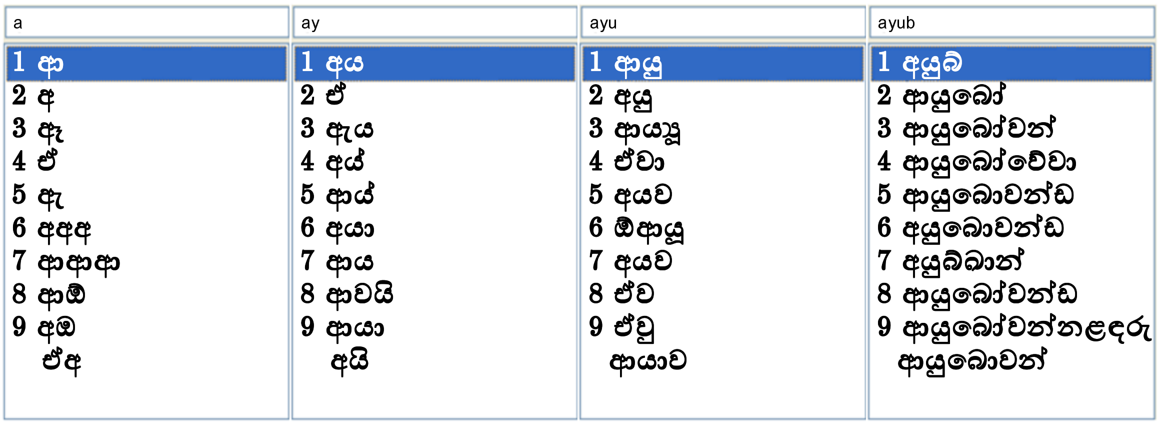
Note that, in this example, the *typing_cost* of the novice user is 1.70 keystrokes per Sinhala character, and the *SriShell Primo* expert reduces it up to 1.15 keystrokes per

Table 5.1: Text Entry Example of *SriShell Primo*

Output text	මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.
Input key sequence *(novice user)*	*meya darannata***2** *awahira badawalin torawa nidahase gaman kirimata saha awashyawana adarada arakshawada salaswa dena lesath adala wagakeem darana siyalu denagenma sri lanka prajatantrika samajawadi janarajaye janadhipathi ellumkarada apekshakarada siti***2***.*
Input key sequence *(SriShell Primo expert)*	*mey drnnaat avhir bdvln trv nidhse gmn kirmt sh avsyvn adr d arksv d slsv dena lest adl vgkim drn sylu dngnm sri lnk prjtntrik smjvd jnrjye jndpt illmkr d apkskr d sitee.*

Sinhala character. Here we assume, pressing key: $\boxed{2}$ to select the menu has the same weight as a normal alphabetic keystroke, and repeating the same key has the weight of two normal alphabetic keystrokes.

Figure 5.3 demonstrates how the menu changes dynamically as the keys are entered, taking ආයුබෝවන්(=āyubōvan: Welcome) as an example. When the user types "*a*," *SriShell Primo* gives a list of candidates in the menu that starts with ආ, අ, ඇ, ඒ, ඈ, etc., as shown in Figure 5.3(a). When the user types to "*ayub*," the intended word ආයුබෝවන් appears for the first time in the menu as the third candidate (Figure 5.3(d)). The user can select the word by arrow keys or by pressing numeric key: $\boxed{3}$. Otherwise he/she can continue typing. When the user types to "*ayubovan*," ආයුබෝවන් (intended word) surfaces as the first menu choice (Figure 5.3(h)). The user can select the menu item by pressing a punctuation key such as a space, comma, period, etc.



a
1 ආ
2 අ
3 ඈ
4 ඒ
5 ඇ
6 අඅඅ
7 ආආආ
8 ආඕ
9 අඔ
ඒඅ

(a)

ay
1 අය
2 ඒ
3 ඇය
4 අය්
5 ආය්
6 අයා
7 ආය
8 ආවයි
9 ආයා
අයි

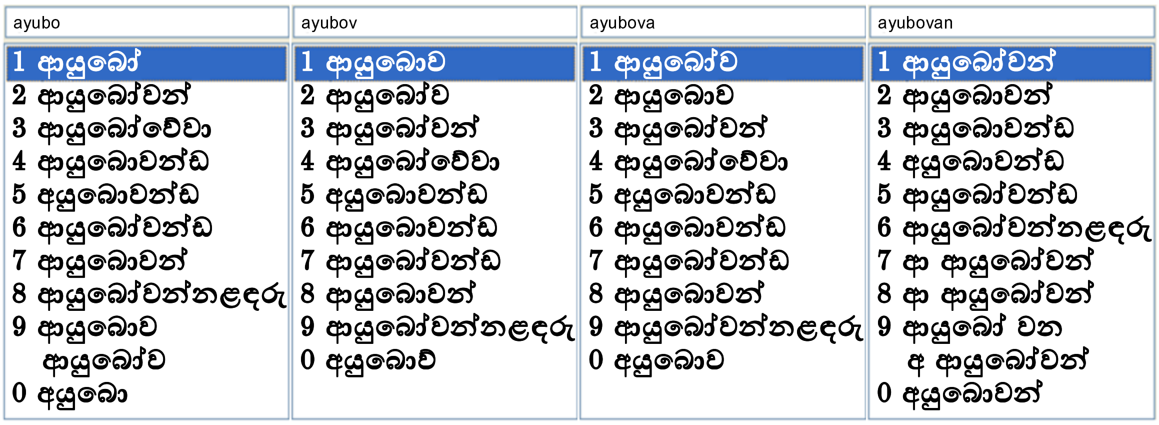
(b)

ayu
1 ආයු
2 අයු
3 ආයූ
4 ඒවා
5 අයව
6 ඕආයූ
7 අයව
8 ඒව
9 ඒවු
ආයාව

(c)

ayub
1 අයුබ්
2 ආයුබෝ
3 ආයුබෝවන්
4 ආයුබෝවේවා
5 ආයුබොවන්ඩ
6 අයුබොවන්ඩ
7 අයුබ්බොන්
8 ආයුබෝවන්ඩ
9 ආයුබෝවන්නළදරු
ආයුබොවන්

(d)



ayubo
1 ආයුබෝ
2 ආයුබෝවන්
3 ආයුබෝවේවා
4 ආයුබොවන්ඩ
5 අයුබොවන්ඩ
6 ආයුබෝවන්ඩ
7 ආයුබොවන්
8 ආයුබෝවන්නළදරු
9 ආයුබොව
ආයුබෝව
0 අයුබො

(e)

ayubov
1 ආයුබොව
2 ආයුබෝව
3 ආයුබෝවන්
4 ආයුබෝවේවා
5 අයුබොවන්ඩ
6 ආයුබොවන්ඩ
7 ආයුබෝවන්ඩ
8 ආයුබොවන්
9 ආයුබෝවන්නළදරු
0 අයුබොවි

(f)

ayubova
1 ආයුබෝව
2 ආයුබොව
3 ආයුබෝවන්
4 ආයුබෝවේවා
5 අයුබොවන්ඩ
6 ආයුබොවන්ඩ
7 ආයුබෝවන්ඩ
8 ආයුබොවන්
9 ආයුබෝවන්නළදරු
0 අයුබොව

(g)

ayubovan
1 ආයුබෝවන්
2 ආයුබොවන්
3 ආයුබොවන්ඩ
4 අයුබොවන්ඩ
5 ආයුබෝවන්ඩ
6 ආයුබෝවන්නළදරු
7 ආ ආයුබෝවන්
8 ආ ආයුබෝවන්
9 ආයුබෝ වන
අ ආයුබෝවන්
0 අයුබොවන්

(h)

Figure 5.3: Text Entering Example ආයුබෝවන්(=āyubōvan: Welcome)

Design Principles

The design principles to realize an efficient and user-friendly text input system can be described as follows. In addition to these two dimensions, completeness should be enforced; any intended text has to be entered anyway.

1. **Highly user-friendly**

   - **High coverage of possible input sequences**

     The Roman character sequence used to represent a Sinhala word depends on the user. For example, all the following sequences represent the same Sinhala word:

$$\left.\begin{array}{l} desei, dase, dese, daasee, \\ desee, dasee, daesei, dasay, \\ deesee, desee, dhasay, dhese \end{array}\right\} \to \begin{array}{l} \text{දෑසේ} \\ \text{(=dǣsē:} \\ \text{in the eyes)} \end{array}$$

On the other hand an input sequence can also be ambiguous:

$$bata \to \left\{\begin{array}{ll} \text{භට} & \text{(=bhata: soldier),} \\ \text{බැට} & \text{(=bæta: hurt),} \\ \text{බට} & \text{(=bata: bamboo or pipe),} \\ \text{බාටා} & \text{(=bātā: a trade name)} \end{array}\right.$$

SriShell Primo can convert all of these possible sequences into the word intended by the user.

- **Self-adaptation**

  The system continues updating the frequencies of each conversion and records them in the "Input Variation Table" described in 5.3.1.

2. **Substantially efficient**

- **Vowel omissions**

  According to UCSC Sinhala Corpus BETA [79], 23.36% of Sinhala phonemes are *'a' vowels*, as shown in Table 5.2. This means that the most frequent pattern of a Sinhala character is Consonant syllabic (C) of Equation (2.3). Non-conversion direct input systems are highly efficient for this, because users can strike a single key to input a Consonant syllabic (C) character. Here we provide a vowel omitting feature to improve typing efficiency, based on Huffman coding concept: shorter key sequences for more frequent characters [85]. For example, අනුරාධපුරය (anurādhapuraya: *Anuradhapura* City) can be input either as *anuradhapuraya* or *anrdpry.*

- **Abbreviated key sequences**

  Some abbreviated key sequences are introduced to improve typing efficiency and to reduce ambiguities. For example, most of the time, phoneme "ත්"(=t; occurrence probability=3.91%) is transliterated as "*th*" or "*t*." If the user

Table 5.2: Sinhala Phonemes and their Occurrence Probabilities

phoneme		frequency	phoneme		probability	phoneme		probability	phoneme		probability
අ	(=a)	23.36	ඇ	(=æ)	2.01	ඳ්	(=ňd)	0.29	ඤ්	(=ñ)	0.02
ඉ	(=i)	6.51	ඒ	(=ē)	2.00	ච	(=c)	0.23	ෆ්	(=f)	0.02
න්	(=n)	5.86	ට	(=ṭ)	1.78	භ්	(=bh)	0.23	ඵ්	(=ph)	0.01
ව	(=v)	4.88	ග්	(=g)	1.63	ථ්	(=th)	0.17	ඪ්	(=ḍh)	0.00
ය්	(=y)	4.38	ඊ	(=ī)	1.21	ඈ	(=ǣ)	0.15	ඞ	(=ṅ)	0.00
ආ	(=ā)	4.27	බ	(=b)	1.06	ඟ්	(=ňg)	0.11	ඍ	(=ṛ)	0.00
ක්	(=k)	4.17	ඔ	(=o)	1.05	*separater*		0.09	ඣ	(=jh)	0.00
ම	(=m)	3.97	ණ්	(=ṇ)	0.77	ඹ	(=m̆b)	0.06	ඃ	(=ḥ)	0.00
ත්	(=t)	3.91	ළ්	(=ḷ)	0.62	ඖ	(=au)	0.04	ඎ	(=ṝ)	0.00
ර	(=r)	3.69	ශ්	(=ś)	0.45	ඡ්	(=ch)	0.04	ඦ්	(=ňj)	0.00
උ	(=u)	3.54	ඕ	(=ō)	0.42	*connector*		0.03	ඏ	(=ḷ)	0.00
ස්	(=s)	3.08	ජ්	(=j)	0.42	ඨ්	(=ṭh)	0.03	’	(=’)	0.00
එ	(=e)	2.61	ඩ	(=ḍ)	0.41	ඬ	(=ňḍ)	0.03	ඐ	(=ḹ)	0.00
ද්	(=d)	2.51	ධ	(=dh)	0.40	ඥ්	(=jñ)	0.03			
ප්	(=p)	2.28	ඌ	(=ū)	0.39	ඛ	(=kh)	0.03			
හ්	(=h)	2.10	ෂ්	(=ṣ)	0.30	ඓ	(=ai)	0.02			
ල්	(=l)	2.02	ං	(=ṃ)	0.30	ඝ්	(=gh)	0.02			

wants to improve his/her input efficiency, he/she will have to choose “*t*,” as it requires only one keystroke to input the phoneme. However, key sequence “*t*” is highly ambiguous, as it is used for other phonemes including “ට”(=ṭ) which also has quite high occurrence probability. In such cases, we introduce abbreviated key sequences:

- $x \rightarrow$ ත්(=t), ථ්(=th)
- $q \rightarrow$ ද්(=d), ධ(=dh)
- $z \rightarrow$ ඇ(=æ), ඈ(=ǣ)

- **Auto completion**

SriShell Primo not only gives Sinhala words that can be completely represented by an input Roman character sequence but it also dynamically adds automatically completed Sinhala words to the menu, as depicted in Figure 5.3.

- **Word combinations**

  A Sinhala word is usually separated by spaces. Our preliminary experiments however revealed that sometimes users omit the spaces especially for frequently co-occurred word pairs. This is because Sinhala has *word boundary problem* as explained in Section 2.1.2. *SriShell Primo* thus allows up to one space omission, and gives word pairs in the menu, if the number of word candidates from the above methods are less than ten.

3. **Complete**

   SriShell Primo also allows *Sri Shell* input sequences. Using *Sri Shell* as a back-off input system, users can input any new Sinhala word that is not included in the word list.

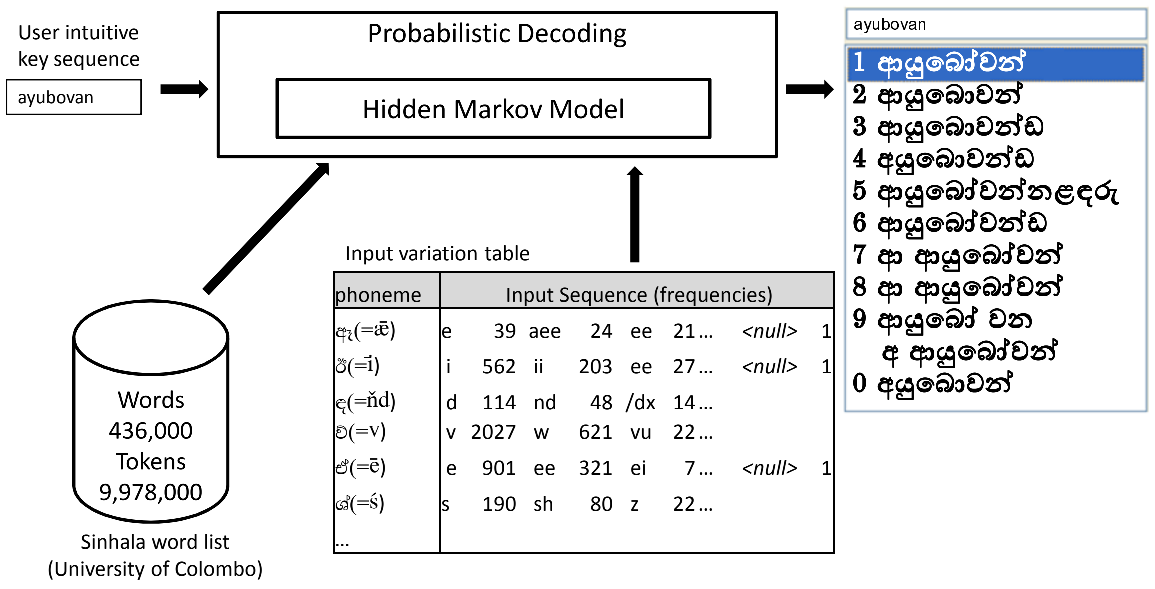
5.3 Overall Architecture

Figure 5.4 illustrates the overall architecture of *SriShell Primo*. A user intuitive key sequence is converted to a Sinhala word through a probabilistic decoding process that employs an input variation table and a Sinhala word list.

5.3.1 Input Variation Table

In Chapter 4 we have carried out an experiment to examine how the highly frequent 275 Sinhala characters are transliterated in Roman characters by 30 Sinhala speakers. We further divided the Roman character sequence for each Sinhala character into phonemes. Based on the experiments, we constructed a table that shows how each Sinhala phoneme can be transliterated into Roman characters by various users, as shown in Table 5.3. The system increases an entry's frequency by 1 each time it is used.

The probability of each conversion is calculated using Equation (5.1):



phoneme	Input Sequence (frequencies)							
ඇ(=ǣ)	e	39	aee	24	ee	21…	<null>	1
ඊ(=ī)	i	562	ii	203	ee	27…	<null>	1
ඳ(=ňd)	d	114	nd	48	/dx	14…		
ව(=v)	v	2027	w	621	vu	22…		
ඒ(=ē)	e	901	ee	321	ei	7…	<null>	1
ශ්(=ś)	s	190	sh	80	z	22…		
…								

Figure 5.4: System Architecture

$$P(c \leftarrow k_i) = \frac{f(c \leftarrow k_i)}{\sum_{i=1}^{n} f(c \leftarrow k_i)}, \tag{5.1}$$

where

c	=	a Sinhala phoneme,
$k_i (i = 1..n)$	=	key sequences that can be converted to phoneme 'c', and
$f(c \leftarrow k_i)$	=	frequency of conversion '$c \leftarrow k_i$'.

Notice that we included special rules for supporting vowel omission. For the frequently occurring phoneme අ(=a), we assigned an artificial frequency f_a for vowel omission (<*null*>). The value for f_a is set to 10% of the frequency of the actually entered අ(=a) phoneme. With respect to Table 5.3, f_a is calculated by $\frac{f_a}{16425+551+f_a} = 0.10$. On the other hand, we assigned f_x for the other vowel phonemes, where f_x is uniformly set to 1. This account reflects an engineering viewpoint. To maintain proper menu ordering, we set the values to achieve the following relation (Equation (5.2)):

$$P(v \leftarrow k) \gg P(v_a \leftarrow <\!null\!>) \gg P(v_{a'} \leftarrow <\!null\!>), \tag{5.2}$$

where

Table 5.3: Input Variation Table

phoneme	Input Sequence (frequencies)											
අ(=a)	a	(16425)	e	(551)							<*null*>	(f_a)
ඈ(=ǣ)	e	(39)	aee	(24)	ee	(21)	ae	(5)	aa	(3)	<*null*>	(f_x)
ඊ(=ī)	i	(562)	ii	(203)	ee	(27)	y	(1)	ie	(1)	<*null*>	(f_x)
ඳ(=ňd)	d	(114)	nd	(48)	/dx	(14)	ndx	(4)	/d	(2)		
ව(=v)	v	(2027)	w	(621)	vu	(22)	wu	(22)	u	(11)		
ඒ(=ē)	e	(901)	ee	(321)	ei	(7)	ay	(2)	a	(1)	<*null*>	(f_x)
ශ(=ś)	s	(190)	sh	(80)	z	(22)	sx	(10)				
ඹ(=b̆)	b	(16)	mb	(7)	/b	(2)						
ඬ(=ňḍ)	/d	(5)	nd	(1)	d	(1)						
ඟ(=ňg)	ng	(21)	/g	(3)	g	(17)						
ඇ(=æ)	e	(526)	ae	(83)	a	(41)					<*null*>	(f_x)
...												

v	=	a vowel phoneme,
v_a	=	vowel phoneme අ(=a),
$v_{a'}$	=	a vowel phoneme other than phoneme අ(=a),
k	=	key sequence in which $k \neq$<*null*>, and
<*null*>	=	null key sequence.

5.3.2 Sinhala Word List

We used a word list provided by the University of Colombo, School of Computing [79]. This word list contains about 436,000 words and their occurrence frequencies, extracted from a 9,978,000 token corpus.

To improve the searching speed, the words are stored in a TRIE [86]–[88] structure: an ordered tree [89] data structure that is used to store an associative array, where each branch represents a consonant part, a vowel part, or a consonant sign part of a Sinhala character. Therefore any single Sinhala character can be retrieved in up to three hops. Figure 5.5 shows a part of our TRIE data structure. To reduce the amount of memory,

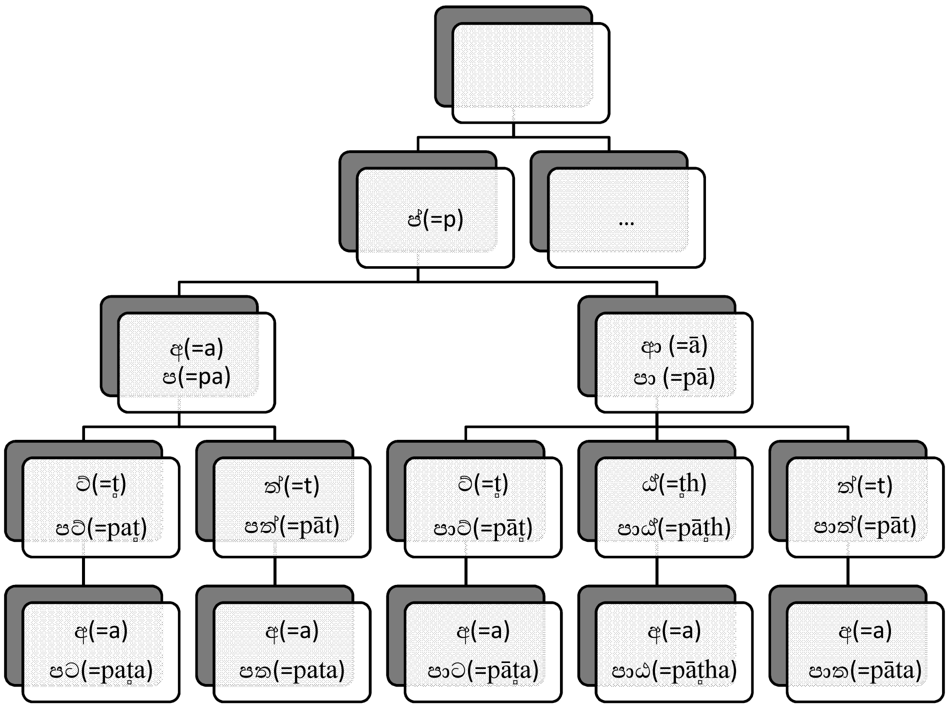


Figure 5.5: TRIE Data Structure

the required part of the data structure is copied onto the memory when the user starts to type.

5.3.3 Probabilistic Conversion Process

In Japanese text input, the process can be divided into two steps: *romaji-nyuryoku*, and the succeeding *kana-kanji conversion* as shown in Figure 5.6. The final Kanji characters depend only on the *Hiragana* representation, but not on the original Roman character representation. For example:

$$P(\text{感知} \leftarrow \text{かんち} \leftarrow \text{kanti}) = P(\text{感知} \leftarrow \text{かんち} \leftarrow \text{kannchi}). \qquad (5.3)$$

Therefore Japanese input system should utilize rich contextual information to choose among possible Kanji candidates as explained in Section 2.2.

In the case of Sinhala, the situation is not the same. Even though various input sequences produce the same Sinhala word as shown in Figure 5.7, still the probability of the conversion should be different by reflecting user preferences based on their intuitions.

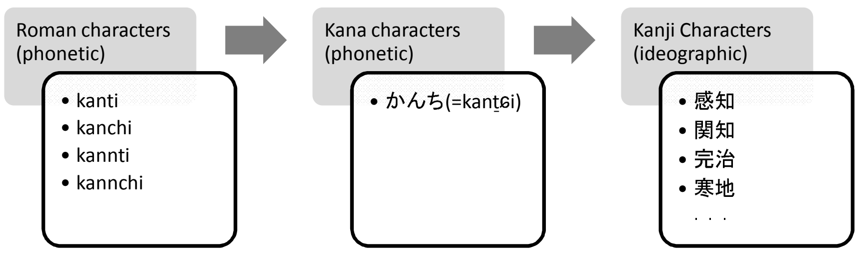

Roman characters (phonetic)
kanti
kanchi
kannti
kannchi
Kana characters (phonetic)
かんち(=kant̠ɕi)
Kanji Characters (ideographic)
感知
関知
完治
寒地
. . .

Figure 5.6: Japanese Roman-kana-kanji Predictive Input System

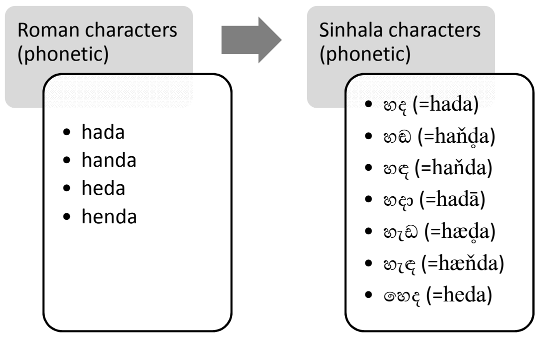


Figure 5.7: Roman-Sinhala Predictive Input System

For example:

$$P(\text{හඬ}(=\text{haňd̥a}) \leftarrow \text{handa}) \quad \neq \quad P(\text{හඬ}(=\text{haňd̥a}) \leftarrow \text{hada}). \tag{5.4}$$

This probabilistic information is used in Hidden Markov Model based predicting procedure.

Hidden Markov Model

We modeled the word generation process with a hidden Markov model (HMM) [90]–[92] whose states correspond to Sinhala phonemes and whose observations are associated with the corresponding input keystrokes. Given this setting, the goal was to estimate the Sinhala word $\hat{w}$ by maximizing $P(c_1^m|k_1^m)$, where $c_1^m = w$ denotes a Sinhala word and k_1^m denotes its input sequence. Here, $c_i(1 \leq i \leq m)$ is a Sinhala phoneme and $k_i(1 \leq i \leq m)$ is an input sequence for each phoneme given in the input variation table. By applying a hidden Markov model, the maximization of $P(c_1^m|k_1^m)$ can be formulated, as shown in

Equation (5.5):

$$
\begin{aligned}
\hat{w} &= \arg\max_{w} P(c_1^m|k_1^m) \\
&= \arg\max_{w} P(k_1^m|c_1^m)P(c_1^m) \\
&\approx \arg\max_{w} \Big(\prod_{i=1}^{m} P(k_i|c_i)\Big)P(c_1^m) \\
&= \arg\max_{w} \Big(\prod_{i=1}^{m} P(c_i \leftarrow k_i)\Big)\left(\frac{P(c_1^m)}{\prod_{i=1}^{m} P(c_i)}\right) \\
&= \arg\max_{w} \Big(\prod_{i=1}^{m} P(c_i \leftarrow k_i)\Big)\left(\frac{P(w)}{\prod_{i=1}^{m} P(c_i)}\right). \qquad (5.5)
\end{aligned}
$$

Here, $P(c_i \leftarrow k_i)$ corresponds to the probability of a specific conversion. Thus the first term in Equation (5.5) can be calculated using the input variation table. $P(c_i)$ is the probability of each Sinhala phoneme, and $P(c_1^m) = P(w)$ is the probability of a specific Sinhala word. Therefore the second term is calculated offline from the word list.

Procedure

Whenever a user strikes a key, *SriShell Primo* creates a list of probable Sinhala character candidates. Then the created candidate list is sorted in descending estimated probabilities as explained in Section 5.4. For example, in Figure 5.3(a), candidates from 1 to 5 are created through this process.

Then *SriShell Primo* searches the Sinhala character sequence list to determine whether any sequence matches the beginning of a Sinhala word. These predicted words are then added to the end of the candidate list. The candidates from 6 on in Figure 5.3(a) are thus added.

If *SriShell Primo* was unable to find any candidates up to this point, it searches for word pairs that match the input character sequence, assuming that the user omitted a space.

Finally the character sequence derived from *Sri Shell* is added at the end of the candidate list to allow the typing of a word that is not included in the word list. The candidate number 0 in Figure 5.3(e) is added at this point. The candidate list is displayed as a menu from which users can select an intended word by mouse, up/down arrow keys, or numeric keys.

This process is repeated for each user keystroke. The selected item can be entered into the document by striking space or punctuation keys.

5.4 Implementation Issues

We have incorporated several technical elements to realize the word-based predictive input system *SriShell Primo*. However these elements introduced technical issues that had to be considered when we were to implement the system.

Handling of <*null*> key sequence: With *SriShell Primo* which is equipped with the input variation table, a key sequence is highly ambiguous; the system has to generate all possible Sinhala character strings for the given input sequence. This forces the system to estimate probabilities of the possible candidate strings, requiring us to implement an efficient computational mechanism. Furthermore, the <*null*> key sequence for vowels introduced to improve the efficiency may significantly slow down the searching process, because an infinite number of Sinhala character strings can be associated with a given key sequence.

We solved this problem by focusing on the fact that a very small number of the possible strings are actually Sinhala words. More specifically, we travelled through the TRIE data structure while generating the possible Sinhala character strings which are on the word list represented in the TRIE. By applying this technique, all the existing Sinhala words that can be represented by a key sequence can be efficiently retrieved.

Word list search: The simplest way to maintain the TRIE data structure is to keep the entire data structure on memory, definitely speeding up the conversion process. However, this strategy introduces another problem; the initialization of the system would be very slow, if the system has to load the whole TRIE data structure into the memory. This problem was solved by focusing on the fact that the whole TRIE data structure was not necessarily required on memory during one typing session. Therefore, we implemented our system to keep the data structure on the disk and load the necessary parts in response to the requirements. Thereby we were able to remedy the trade-off problem between the initialization time and data retrieval efficiency.

Pruning candidates: In order to implement the auto completion function, it is required to search the TRIE structure down to the leaves. This process takes longer time, because the system has to go through hundreds of words and select the most probable ones among them. We have reduced the processing time by only considering the most probable 10 words, for each the probability had been computed beforehand and stored in the data structure.

5.5 Evaluation

This section describes the evaluation of the proposed input system. We evaluated the proposed method in terms of user-friendliness and efficiency through an experiment with subjects.

First we gave the test subjects 10 to 30 minutes to practice with *SriShell Primo* until they felt comfortable with it. Then each was given a Sinhala text to input taken from Sinhala newspapers: "*Divaina*," "*The Silumina*," "*Lakbima*," and "*Lankadeepa*." The text lengths ranged from 812 to 1418 characters. We informed them to input a Sinhala word by whatever Roman character sequence they considered best to represent the Sinhala word. We also informed them that they can increase their Sinhala typing speed by omitting vowels. *SriShell Primo* maintains a log that records the typed keys, the selected menu items, and time lapses between them. This experiment was carried out on a group of 10 subjects (5 females and 5 males, age 18-45 years). Our test subjects were native Sinhala speakers who use computers in their daily lives in English and some in Japanese. However, most had no experience typing in Sinhala with any Sinhala input system.

5.5.1 User-friendliness

As discussed above, user-friendliness is quantified by how a required input key sequence resembles the user intuitive character sequence. Therefore we evaluated an input system user-friendliness by calculating the edit distance between the user intuitive input sequence and the input sequence acceptable to the system.

In *SriShell Primo*, if a user fails to input a Sinhala word using his/her initial key sequence due to incompleteness of the input variation table or typing errors, he/she revises it to be accepted by the system. To calculate the average edit distance as per Equation (3.15), we used the initial key sequence of the user as the *user_intuitive_character_sequence* and the key sequence accepted by *SriShell Primo* as the *input_sequence*. The calculated edit distance safely assessed the system's user-friendliness, because typing errors might have increased the edit distance as well.

In our experiment the average edit distance per Sinhala character was 0.07, which is far better than the 0.33 of *Natural SinGlish* shown in Table 4.6. This means that the users were able to correctly type 93% of the Sinhala characters in the text with their initial input sequence, given the current input variation table. Note that by personalizing the input variation table, higher efficiency and user-friendliness can be achieved. For example an expert can have less number of variations in his/her input variation table, and more abbreviated key sequences, in order to reduce ambiguity. On the other hand a computer used in a public place such as a library etc., can provide more flexible input system by adopting more variations into the input variation table.

5.5.2 Efficiency

We redefine the *typing_cost* given in Equation (3.4) by adding a menu selecting time factor as shown in Equation (5.6). Both keystroke and Sinhala typing speeds are calculated using Equations (3.7) and (3.8):

$$\begin{aligned} typing\_cost &= w_m + \sum_{i=1}^{N} P(c_i) \times (|K_{C_i}| + w_s \times S(K_{C_i}) + w_r \times R(K_{C_i})), && (5.6) \\ w_m &= \frac{t_{sel}}{t_{xy}} \times \frac{1}{ACPW}, && (5.7) \end{aligned}$$

where

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

$ACPW$ = average number of Sinhala character per Sinhala word.

Note that, as explained in "Text entry example of *SriShell Primo*" in Section 5.2, this additional menu selecting time factor can be reduced to nearly zero using less ambiguous key sequences.

The results are summarized in Figure 5.8. The X-axis shows keystroke typing speed in keystrokes per minute, and the Y-axis shows the Sinhala typing speed in Sinhala characters

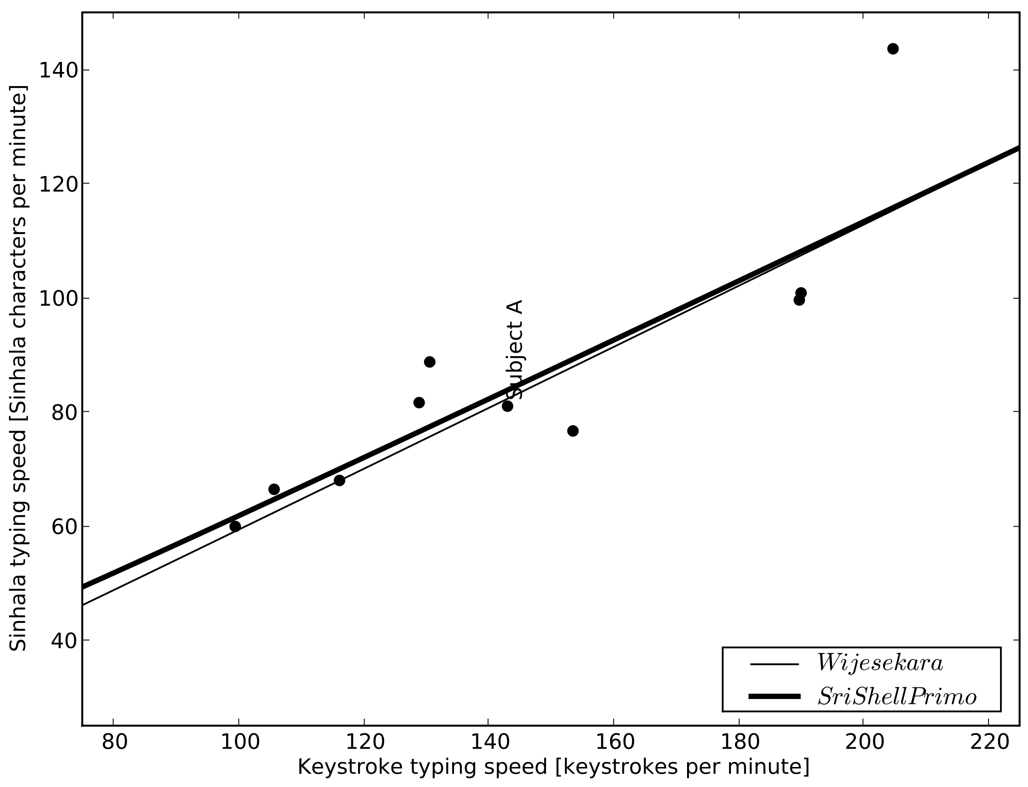


Figure 5.8: Average Typing Cost

per minute. For comparison purposes we plotted the result for the *Wijesekara* keyboard layout, which was the most efficient existing Sinhala input system. A "●" shows subject performances. For example, Subject A's keystroke typing speed is 143.1 keystrokes per minute and his/her Sinhala typing speed is 81.0 Sinhala characters per minute. This graph shows that *SriShell Primo* is comparable with *Wijesekara*, because 5 subjects out of 10 subjects could type Sinhala text more efficiently than *Wijesekara*. Since *Wijesekara* is the most efficient existing input system, as shown in Figure 3.5, the efficiency of *SriShell Primo* is not worse than the other existing input systems discussed in Section 2.3.

This efficiency was achieved by our two proposed techniques. First, the hidden Markov model improved the menu ordering where w_{sel} went down. In Figure 5.9, a "▲" shows how the performances are degraded if the system only uses the occurrence frequencies of the words to determine the menu order, without considering the input variation weights. By comparing the "●"s and "▲"s in Figure 5.9, it is clear how the performances have been

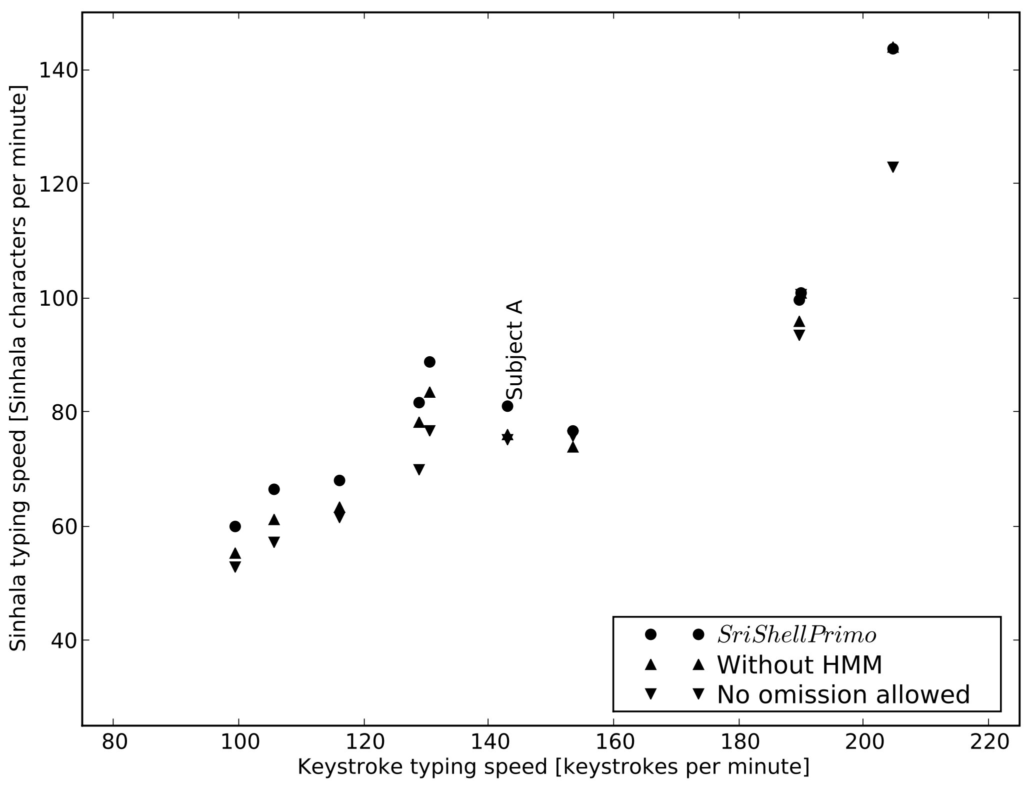


Figure 5.9: Sinhala Typing Speed vs. Keystroke Typing Speed

improved. For example, Subject A's Sinhala typing speed is 81.0 Sinhala characters per minute and decreases to 75.9 Sinhala characters per minute if the hidden Markov model is not used; it decreases to 75.0 Sinhala characters per minute if the vowel omission facility is unavailable.

Second, *SriShell Primo* supports the omission of vowels with which the number of required keystrokes itself has been reduced. In Figure 5.9, the "▼"s show how the performances are degraded if the users do not omit any vowels. These values are calculated on the following basis. If users do not omit any vowel, that implies that the users will have to type at least one extra keystroke instead of omitting a vowel. By comparing the "●"s and "▼"s, the vowel omission feature has clearly contributed to the efficiency.

5.5.3 Overall Assessment

SriShell Primo is suitable both for novices and more advanced users. This system's higher user-friendliness supports novice users, because it accepts almost all user intuitive input sequences. This system is also highly efficient, as indicated by the result that the test subjects reduced the typing cost (keystrokes per Sinhala character) to 1.56. These performances were achieved because the system supports a vowel omission feature with which users can improve their Sinhala typing speed just by omitting vowels, unlike any other existing Sinhala input system. Thanks to the implementation details described in Section 5.4, the system could generate a menu list fast enough to be utilized by an expert; the actual average elapsed time was 34.7 milliseconds.

However, excessive omission of vowels leads to high ambiguity. For example, "*pt*" may mean පත්(=pat), අපට(=apaṭa), පට(=paṭa), පාඨ(=pāṭha), etc. For this reason a user may not get the intended word as the top menu candidate. To avoid this problem users must judiciously choose where to omit vowels. Sometimes this thinking process may hamper the keystroke typing speed. However we believe that if users continue to use the system, they will learn where vowels can be safely omitted and will recover their keystroke typing speeds. Otherwise, frequencies for vowel omissions in the input variation table can be adjusted to users.

5.6 Conclusion

In this chapter, we have proposed a word-based predictive Sinhala input system called *SriShell Primo*. The most prominent feature of this system is its high user-friendliness. A key to the user-friendliness is a pre-compiled *input variation table* that lists weighted correspondences between conceivable Roman character sequences and the associated Sinhala phonemes. This table is constructed to accept and adapt to the key sequences for a wide range of users. The introduction of this device however calls for the system to realize a mechanism to choose the best Sinhala character sequence toward the given user input sequence. We therefore have proposed a word-based predictive system to narrow down the ambiguities. This word-based system is also beneficial, as it can propose completion candidates during the input process. *SriShell Primo* has maximum user-friendliness while exhibiting a level of efficiency that is comparable to the most efficient direct input system. Our test subjects highly appreciated the user-intuitiveness, and commented that the system is very easy to use. They anticipated that this system can be popular among Sinhala computer users.

We have tested our system on a personal computer: Genuine Intel CPU 2.0GHz processor, 2.0GB of RAM, and Microsoft Windows XP operating system. The system well responds in real-time to the user's key strokes. As this level of PC specification is not very demanded these days, our system can be fully utilized by general Sinhala users; this will provide them opportunities to generate and disseminate various contents in Sinhala. The system is written in Microsoft Visual C#, and implements a fast search algorithm utilizing the TRIE data structure.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis we have proposed a highly user-friendly yet efficient Sinhala text input system, targeting general Sinhala computer users, who have average-level operational knowledge of computers, and are familiar with Roman character keyboards. We have approached to this goal by implementing two systems: *Sri Shell*, a phonetically-principled system, and *SriShell Primo*, a word-based predictive system. To be user-friendly, *Sri Shell* is based on a phonetically-principled key assignments, while *SriShell Primo* is equipped with a mechanism that accepts user-intuitive key sequences.

We have also established adequate measures for evaluating the user-friendliness and efficiency of Sinhala input systems, because we think the user-friendliness is quite important, given the targeted users. To this end, we have proposed an efficiency measure that quantifies the average typing cost per Sinhala character. We have also proposed a user-friendliness measure that evaluates the intuitiveness of required/acceptable key sequences. These measures are proven useful in evaluating existing Sinhala input systems as well as the proposed two systems.

Each chapter of the thesis is summarized as follows:

In Chapter 1 we gave a brief introduction on Sinhala language and summarized use of computers in Sinhala. Based on these arguments, our research motivation is stated.

Chapter 2 provided necessary background information to understand the presented research: linguistic nature of Sinhala language and classification of text input systems. Based on these materials we reviewed representative Sinhala input systems. In the final section of this chapter, desiderata for realizing an effective Sinhala input system are presented.

Chapter 3 proposed a new methodology to evaluate Sinhala input systems. First we have discussed the general measures used to evaluate input systems. Text input systems should be evaluated not only by the efficiency but the user-friendliness, especially when the users are not professionals. The efficiency is quantified by the average typing cost per Sinhala character, while the user-friendliness is assessed by the average edit distance between a user-intuitive character sequence and the input sequences of an input system. We reported the evaluation results of existing Sinhala systems by employing these measures. We finally proved that the proposed user-friendly measure is valid to evaluate the user-friendliness through questionnaire based experiment.

One of the strategies to ensure the user-friendliness is to develop a key assignment which is intuitive or principle-based. In Chapter 4, we proposed a phonetically-principled associative conversion-based direct input system called *Sri Shell*. The system is a light-weighted application independent module that can be realized without any language resources such as corpora or dictionaries. This chapter concluded that *Sri Shell* is moderately user-friendly while maintaining better level of efficiency comparing to other conversion-based direct input systems. It also should be noted that *Sri Shell* is a complete input system that can be utilized in combination with the next proposed system *SriShell Primo*.

In Chapter 5, we proposed a word-based predictive Sinhala input system called *SriShell Primo*. The most prominent feature of this system is its high user-friendliness. A key to the user-friendliness is a pre-compiled *input variation table* that lists weighted correspondences between conceivable Roman character sequences and the associated Sinhala phonemes. This table is constructed to accept and adapt to the key sequences for a wide range of users. The introduction of this device however calls for the system to realize a mechanism to choose the best Sinhala character sequence toward the given user input sequence. We therefore proposed a word-based predictive system to narrow down the ambiguities. This word-based system is also beneficial, as it can propose completion candidates during the input process. This chapter concluded that *SriShell Primo* has maximum user-friendliness while exhibiting a level of efficiency that is comparable to the most efficient direct input system.

Chapter 6, summarizes the results, and proposes research issues for improving the proposed systems, as well as more general research agenda for computing in Sinhala.

6.2 Future Work

Our future work can be divided into two topics: further improvements to Sinhala input systems and other improvements in the field of Sinhala computing.

Further Improvement of Sinhala Input Systems

We hope to improve our text input system in three aspects: (1) Improve the coverage of our predictive input system; *SriShell Primo*, (2) Improve the typing efficiency, (3) Improve the quality of input text, by introducing misspelling prevention function.

1. **Improvements to the coverage** *SriShell Primo* uses a word list of 436,000 words. However, we need a list of words with better coverage to assure better applicability. This task is achievable by developing a systematic and automatic way to generate morpho-syntactically related derivational word forms as explained in Section 2.1.2. For example, in our word list all declensions: ගස(=gasa: a tree), ගස්(=gas: trees), ගසට(=gasaṭa: to tree), ගස්වලට(=gasavalaṭa: to trees), etc. are included as separate entries. We expect to mechanically produce these derivational word forms by applying techniques such as "Prediction by Partial Matching" [93, 94].

2. **Improvements to the efficiency** *SriShell Primo* gives the user intended word as the first choice of the menu in a high probability. However we can further improve the efficiency, by improving prediction accuracy. Here, contextual linguistic models such as word bi-grams [95], can be used to improve the prediction accuracy.

   On the other hand, in the present system, users have to look at the screen time to time to check whether the selected word is correct or not. We can improve this checking efficiency by dictating the input word back to the user; using Sinhala text to speech techniques [6], where the users do not have to look at the screen. T. Magnuson et al. [96] argue that by dictating the input words back to the user, the typing speed of a predictive input system can be improved to a level, which is comparable with the speed of a direct input system.

3. **Misspelling prevention** Sinhala language has some character pairs where both characters are pronounced exactly the same, as shown in Table 6.1. For this reason many Sinhala speakers frequently make spelling mistakes [97], even though Sinhala uses

Table 6.1: Ambiguously Pronounced Sinhala Characters

Character 1		Character 2		Modern Pronunciation
Character	Original Pronunciation	Character	Original Pronunciation	
න	[na]	ණ	[ɳa]	[na]
ල	[la]	ළ	[ɭa]	[la]
ශ	[ɕa]	ෂ	[ʂa]	[ɕa/ʂa]
ක්‍රු	[kru]	කෘ	[kr]	[kru]
ඤ	[ɲa]	ඥ	[ɟɲa]	[ɲa]

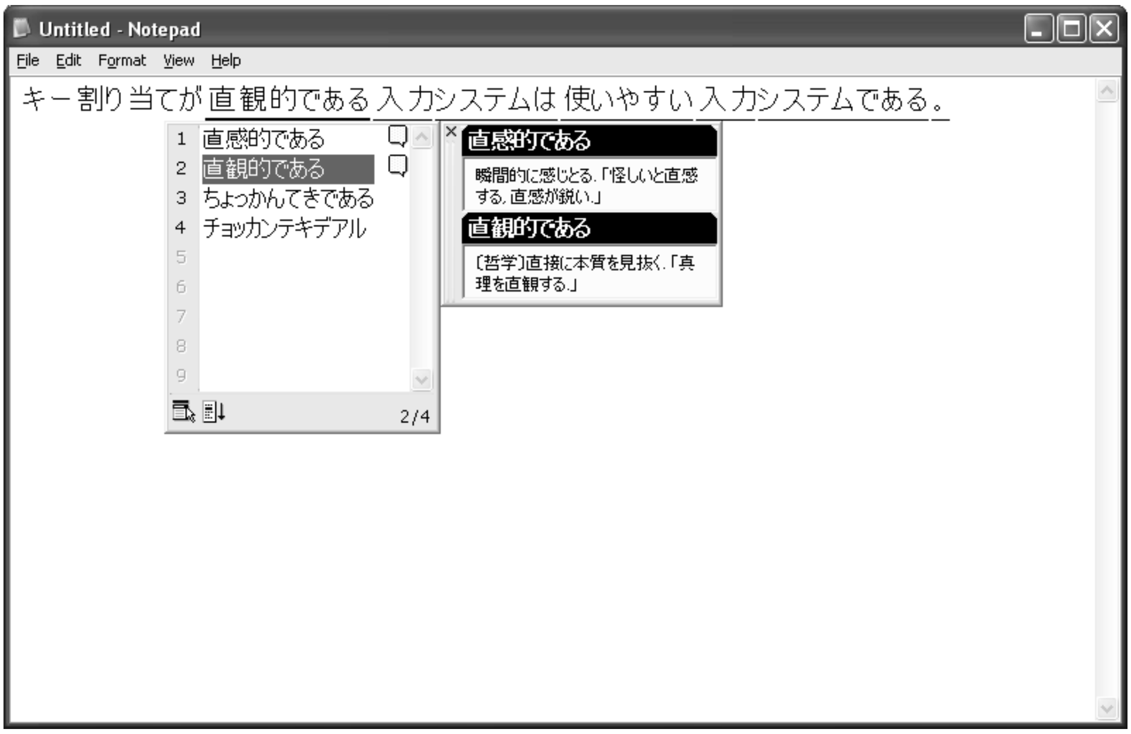
phonograms. Since *SriShell Primo* gives the correct spellings from the word list, we believe that this problem is fixed to a considerable extent. However, there are homonym pairs which are not possible to disambiguate in word level.

We may be able to consult some techniques utilized in Japanese text input. Actually, Japanese is a language which has a large number of homonyms. Japanese input systems assist the user to select the proper word not only by predicting the appropriate word based on the context, but also by giving an explanation about the word, as shown in Figure 6.1. We hope to adopt these kinds of techniques to support the user to decide the appropriate word.

In addition to these issues directly associated with the input system, we should also improve the proposed evaluation measures. As discussed in Chapter 3, our measures successfully assess all the dimensions of usability, but *satisfaction*. Therefore we may need to establish a comprehensive measure to access the satisfaction, which would be highly subjective.

Computing in Sinhala

Given a user-friendly and efficient input system, Sinhala computer users will be able to produce not only their own Sinhala text, but also translations of foreign text. By arranging these data as monolingual and bilingual corpora, they can be used in future researches such as: machine translation systems, error correction tools for OCRs, etc. to further improve the accuracy and hence the applicability.



キー割り当てが 直観的である 入力システムは 使いやすい 入力システムである。

1 直感的である
2 直観的である
3 ちょっかんてきである
4 チョッカンテキデアル
5
6
7
8
9
2/4

直感的である
瞬間的に感じとる.「怪しいと直感する, 直感が鋭い.」

直観的である
〔哲学〕直接に本質を見抜く.「真理を直観する.」

Figure 6.1: Input System Level support for Inputting Japanese Homonyms

As Sinhala has a very limited number of speakers, most often Sinhala people have to depend on data written in foreign languages, in order to acquire information from the outside world. Therefore, a machine translation system which translates from foreign languages to Sinhala is greatly anticipated. Though English is the most widely used language all over the world, it is not an easy task to implement an English-to-Sinhala machine translation system, because the grammars are highly different.

In this regard, Japanese-to-Sinhala translation systems are highly expected, because huge amount of electronic data is available in Japanese. Also translation system of this kind may be feasible, as Japanese grammar exhibits many similarities with Sinhala grammar.

Therefore, our natural next step toward further expansion of Sinhala computing is to implement a Japanese-to-Sinhala translation support system which can benefit Japanese-to-Sinhala translators. Such a translational aid will also play a role in constructing bilingual corpora, which, in turn, can be utilized to improve the translation support system.

Acknowledgement

Upon the completion of this thesis, I would like to take this opportunity to express my sincerest gratitude to those who have done their best to offer me assistance.

First and foremost, I am deeply indebted to my supervisor, Prof. Dr. Fumio Kishino of the Graduate School of Information Science and Technology at Osaka University, who not only accepted me as his Ph.D. Student but also gave me insightful guidance and great encouragement from my undergraduate program.

I am heartily grateful to Prof. Dr. Yoshihiko Hayashi of the Graduate School of Language and Culture for supervising my doctoral thesis. His continuous encouragement, careful reading and invaluable comments have helped to accomplish this thesis. This would never have been completed without his unfailingly wise advice and support.

I would also like to acknowledge the committee members of my thesis, Prof. Dr. Toru Fujiwara, Prof. Dr. Shojiro Nishio, and Prof. Dr. Norihisa Komoda of the Graduate School of Information Science and Technology at Osaka University. Their insightful and constructive comments considerably improved the quality of the thesis.

I am very grateful for the help and support from Associate Prof. Dr. Yuichi Itoh, and Associate Prof. Dr. Yoshifumi Kitamura of Human Interface Engineering Laboratory.

Many thanks to Prof. Dr. Kogure and Associate Prof. Dr. Haruo Noma of Advanced Telecommunications Research Institute International, Kyoto, for their repeated instances of kindness and assistance during my pursuance of the Master Program and throughout the Ph.D. Program.

To Prof. Dr. A. R. Weerasinghe, Mr. Viraj Welgama, and Mr. Asanka Wasala of the Language Technology Research Laboratory of University of Colombo School of Computing, I would extend my heartfelt gratitude for their assistance in furnishing me a Sinhala corpus and suggestions to improve this thesis.

I am grateful to Prof. Dr. Norio Furushiro, Associate Prof. Dr. Tomoko Arikawa, Ms. Miho Yamagishi, Ms. Keiko Fukumoto, Ms. Kahori Tomino, Ms. Hitomi Ami, Ms. Kyoko Bandoh, Ms. Misao Saito, and other teachers and staff members of Osaka University International Student Center for their assistance.

Furthermore, I would like to thank Mr. and Ms. Junnosuke Hattori, Ms. Miyuki Kijima, Mr. Hiroshi Ida, Ms. Natsuki Kijima, Mr. and Ms. Torayuki Ishimitsu, Ms. Yoko Nagahama, Mr. and Ms. Yosuke Tsujii, Ms. Kimiko Miura, Mr. and Ms. Tamio Hara, and all other Japanese volunteers for the overall support for my studies in Japan.

I owe thanks to my good friends Dr. Ms. Wong Pui Wah, Ms. Ng Wee Hia, Ms. Mogana Sunthari, and Mr. Navinda Wickramasinghe as well as all my well wishers who had always been a great source of support, and to my course mates, Dr. Ryoichi Watanabe, Mr. Bora Savas, Mr. Masataka Niwa, Mr. Satoshi Sakurai, Dr. Kazuki Takashima, Mr. Tokuo Yamaguchi, and Mr. Hiromi Onoe who had shared their experience and knowledge with me.

My special appreciation goes to Monbukagakusho (Japanese Government) for their funding during my undergraduate and post-graduate studies, which has enabled me to achieve my dream to obtain a doctoral degree in Japan.

I am highly indebted to my parents, Mr. M. D. Nimal Gunasiri Goonetilleke and Ms. K. Seetha Nandani Goonetilleke and my brother, Mr. M. D. Krishan Darshana Goonetilleke for their invaluable encouragement and help with the surveys and experiments conducted in Sri Lanka. I am grateful to my wife, Ms. Uthpala Goonetilleke for her understanding, support and tolerance which made it possible for me to successfully complete this thesis. I would like to thank her parents, Mr. and Ms. H. K. De Silva, and her sister, Ms. Wathsala De Silva, who have been encouraging me always.

Finally, I am heartily thankful to all my friends who had extended their continuous encouragement and inspirations thus far.

References

[1] U. S. Department Of State: “Background Note: Sri Lanka,” `http://www.state.gov/r/pa/ei/bgn/5249.htm`, 2008.

[2] Internet World Stats, `http://www.internetworldstats.com/stats3.htm`, 2008.

[3] List of Wikipedias, `http://meta.wikimedia.org/wiki/List_of_Wikipedias`, 2008.

[4] H. L. Premaratne, E. Järpe, and J. Bigun: “Lexicon and hidden Markov model-based optimisation of the recognised Sinhala script,” *Pattern Recognition Letters*, Vol. 27, No. 6, pp. 696–705, 2006.

[5] H. L. Premaratne: “Recognition of Printed Sinhala Characters Using Linear Symmetry,” in *ACCV2002: The 5th Asian Conference on Computer Vision*, pp. 23–25, 2002.

[6] R. Weerasinghe, A. Wasala, V. Welgama, and K. Gamage: “Festival-si: A Sinhala Text-to-Speech System,” in *Proceedings of Text, Speech and Dialogue, 10th International Conference*, pp. 472–479, TSD 2007, 2007.

[7] S. Thelijjagoda: “A study on machine translation system from Japanese to Sinhala on MT engine Jaw,” Gifu University, 2006. [Ph. D. thesis]

[8] R. Weerasinghe: “A Statistical Machine Translation Approach to Sinhala-Tamil Language Translation,” in *SCALLA*, 2004.

[9] G. V. Dias: “Challenges of enabling IT in the Sinhala Language,” in *Proceedings of 27th Internationalization and Unicode Conference*, pp. 1–7, 2005.

[10] A. Dvorak and D. L. Daeley: "Typewrite keyboard." US patent 2,040,248 U.S. Patent Office, 1936.

[11] L. J. West: "The standard and dvorak keyboards revisited: Direct measures of speed," *Santa Fe Institute Working Paper*, 1998.

[12] "ISO 3602: 1989 Romanization of Japanese (kana script)," `http://www.age.ne.jp/x/nrs/iso3602/iso3602_unicode.html`, 1989.

[13] S. Goonetilleke, Y. Hayashi, Y. Itoh, and F. Kishino: "An efficient and user-friendly Sinhala input method based on phonetic transcription," *IPSJ SIG Technical Reports*, Vol. 2006, No. 124, p101–106, 2006.

[14] S. Goonetilleke, Y. Hayashi, Y. Itoh, and F. Kishino: "An efficient and user-friendly Sinhala input method based on phonetic transcription," *Journal of Natural Language Processing*, Vol. 14, No. 5, pp. 147–166, 2007.

[15] S. Goonetilleke, Y. Hayashi, Y. Itoh, and F. Kishino: "*SriShell Primo*: A predictive Sinhala text input system," in *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pp. 43–50, 2008.

[16] S. Goonetilleke, Y. Hayashi, Y. Itoh, and F. Kishino: "*SriShell Primo*: A user-friendly yet efficient Sinhala text input system," *Journal of Natural Language Processing*, (accepted for publication).

[17] R. Salomon: "Brahmi and Kharoshthi," in Daniels and Bright (Eds.), *The World's Writing Systems*, 1996.

[18] R. F. Hosking and G. M. Meredith-Owens: "A Handbook of Asian Scripts," British Museam, 1966.

[19] K. F. Holle: "Table of old and new Indic alphabets: Contribution to the paleography of the Dutch Indies," *Written language and literacy*, Vol. 2, No. 2, pp. 167–245, 1999.

[20] D. A. Indrasēna: *Siṃhala Akshara Mālāva*, Sridevi Printers (pvt) Ltd, 2001. [In Sinhala]

[21] Glossary of Unicode Terms, `http://unicode.org/glossary/`, 2008.

[22] Y. Mikami: *A History of Character Codes in Asia (in Japanese)*, Kyoritsu Publishing Co., 2002.

[23] M. Davis: "Text Boundaries, Unicode Standard Annex #29," `http://unicode.org/reports/tr29/`, 2006.

[24] National Library at Kolkata Romanization, `http://en.wikipedia.org/wiki/National_Library_at_Kolkata_romanization`.

[25] "Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet," International Phonetic Association, Cambridge University Press, 1999.

[26] P. Ladefoged: "The revised International Phonetic Alphabet," *Language*, Vol. 66, No. 3, pp. 550–552, 1990.

[27] U. S. Sannasgala: "Pada sādhanaya," in U. S. Sannasgala and A. Perera (Eds.), *Vyākaraṇa Vimansāva*, pp. 494–513, Rasangani Pradeepika Kodikara, 1999. [In Sinhala]

[28] A. Joshi, A. Ganu, A. Chand, V. Parmar, and G. Mathur: "Keylekh: a keyboard for text entry in Indic scripts," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, pp. 928–942, 2004.

[29] M. Silfverberg: " Historical Overview of Consumer Text Entry Technologies, "in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 3–25, Morgan Kaufmann, 2007.

[30] Indian text converter "Baraha," `http://www.baraha.com/`.

[31] R. Gupta and V. Sornlertlamvanich: " Text entry in south and southeast asian scripts," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 227–249, Morgan Kaufmann, 2007.

[32] W. Hadamitzky and M. Spahn: "Japanese Kanji & Kana Revised Edition: A Guide to the Japanese Writing System," Tuttle Language Library, 1997.

[33] R. B. Kaplan and R. B. Baldauf Jr.: *Language Planning and Policy in Asia: Japan, Nepal and Taiwan and Chinese Characters*, Multilingual Matters, 2008.

[34] "KIS Input Sytem," ki-systems Co., LTD.
`http://ki-systems.com/index.htm`.

[35] "NE–KANTEX Input System," KANTEC Co., LTD.
`http://www.kantec.ne.jp/w_kcf.html`.

[36] H. Yamada: "A historical study of typewriters and typing methods: from the position of planning Japanese parallels," *Journal of Information Processing*, Vol. 2, No. 4, pp. 175–202, 1980.

[37] H. Yamada: "Certain problems associated with the design os input keyboards for Japanese writing," in W. E. Cooper (Ed.), *Cognitive aspects of skilled type writing*, pp. 305–407, 1983.

[38] A. Kurihara and H. Kurosaki: "About a transformation method of kana into kanji text," *Technical Report of the University of Kyusyu*, Vol. 39, pp. 659–664, 1967. [In Japanese]

[39] K. Mori and T. Yagihashi: *The birth of Japanese word processors.* Maruzen publishing. 1989. [In Japanese]

[40] K. Tanaka-Ishii, M. Zhou, and J. D. Kim: "Text entry in east asian langugaes," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 203–225, Morgan Kaufmann, 2007.

[41] "Google Indic Transliteration,"
`http://www.google.co.in/transliterate/indic/`.

[42] K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi: "Japanese input system with digits: Can Japanese be input only with consonants," Human Language Technology Conference, pp. 211–218, 2001.

[43] Y. Ichimura, Y. Saito, K. Kimura, and H. Hirakawa: "Kana-kanji conversion system with input support based on prediction," in *Proceedings of the 18th conference on Computational linguistics*, pp. 341–347, 2000.

[44] K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi: "Entering text with a four-button device," in *Proceedings of the 19th international conference on Computational linguistics*, pp. 1–7, 2002.

[45] M. Nagata: "A study on Japanese text processing by statical model," University of Kyoto, 1998. [Ph.D. thesis; in Japanese]

[46] Z. Chen and K. F. Lee: "A new statistical approach to Chinese Pinyin input," in *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 241–247, 2000.

[47] Z. Chen, M. J. Li, F. Zhang, and R. Yang: "Chinese Pinyin Input on Mobile Phone," in *The 2nd International Symposium on Chinese Spoken Language Processing*, pp. 13–15, 2000.

[48] J. F. Gao and K. F Lee: "Distribution-based pruning of backoff language models," in *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 579–588, 2000.

[49] N. Elumeze and K. Nishimoto: "Intelligent Predictive Text Input System using Japanese Language," *Final Report for CSCI 5832: Natural Language Processing*, 2006.

[50] *Kaputadotcom*, "Sinhala/Tamil Fonts & Software Download Zone," `http://www.info.lk/slword/news.htm`, 2008.

[51] A. D. R. Sasanka: *Natural SinGlish*, `http://www.geocities.com/naturalsinglish/`, 2004.

[52] J. Nielsen: *Usability Engineering (The Morgan Kaufmann Series in Interactive Technologies)*, Morgan Kaufmann, 1994.

[53] T. Masui: "An efficient text input method for pen-based computers," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 328 – 335, 1998.

[54] J. O. Wobbrock: "Measures of Text Entry Performance," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 47–74, Morgan Kaufmann, 2007.

[55] I. S. MacKenzie: "Evaluation of text entry techniques," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 75–101, Morgan Kaufmann, 2007.

[56] I. S. MacKenzie and R. W. Soukoreff: "Text entry for mobile computing: models and methods, theory and practice," *Human-Computer Interaction*, Vol. 17, No 2 & 3, pp. 147–198, 2002.

[57] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay: "Dasher - A data entry interface using continuous gestures and language models," in *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pp. 129–137, 2000.

[58] G. Alsio and M. Goldstein: "Productivity Prediction by Extrapolation: Using Workload Memory as a Predictor of Target Performance," *Behaviour and Information Technology*, Vol. 19, No 2, pp. 87–96, 2000.

[59] I. S. MacKenzie, B. Nonnecke, S. Riddersma, C. McQueen, and M. Meltz: "Alphanumeric entry on pen-based computers," *International Journal of Human-Computer Studies*, Vol. 41, No. 5, pp. 775–792, 1994.

[60] H. Rau and S. S. Skiena: "Dialing for documents: an experiment in information theory," in *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pp. 147–155, 1994.

[61] D. Venolia and F. Neiberg: "T-Cube: a fast, self-disclosing pen-based alphabet," in *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 265–270, 1994.

[62] P. Isokoski and M. Käki: "Comparison of two touchpad-based methods for numeric entry," in *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 25–32, 2002.

[63] T. Evreinova, G. Evreinov, and R. Raisamo: "Four-key Text Entry for Physically Challenged People," in *Adjunct Proceedings of the 8th ERCIM Workshop "User Interfaces for All,"* 2004.

[64] M. Ingmarsson, D. Dinka, and S. Zhai: "TNT: a numeric keypad based text input method," in *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 639–646, 2004.

[65] E. Matias, I. S. MacKenzie, and W. Buxton: "One-handed touch-typing on a QWERTY keyboard." *Human-Computer Interaction*, Vol. 11, No. 1, pp. 1–27. 1996.

[66] I. S. MacKenzie and S. X. Zhang: "The design and evaluation of a high-performance soft keyboard," in *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 25–31, 1999.

[67] R. W. Soukoreff and I. S. MacKenzie: "Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic," in *CHI '01 extended abstracts on Human factors in computing systems*, pp. 319–320, 2001.

[68] R. W. Soukoreff and I. S. MacKenzie: "Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric," in *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 113–120, 2003.

[69] J. O. Wobbrock and B. A. Myers: "Analyzing the input stream for character-level errors in unconstrained text entry evaluations," *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 13, No.4, pp. 458–489, 2006.

[70] J. O. Wobbrock, B. A. Myers, and B. Rothrock: "Few-key text entry revisited: mnemonic gestures on four keys," in *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 489–492, 2006.

[71] J. R. Lewis: "Input rates and user preference for three small input methods: Standard keyboard, predictive keyboard, and handwriting," in *Proceedings of the Human Factors and Ergonomics Society*, pp. 425–429, 1999.

[72] D. Hughes, O. Buyukkokten, and J. Warren: "Empirical bi-action tables: a tool for the evaluation and optimization of text input systems, application I: stylus keyboards," *ACM Transactions on Computer-Human Interaction (TOCHI)*, Vol. 17, pp. 131 – 169, 2002.

[73] I. S. MacKenzie: "A Note on Calculating Text Entry Speed," unpublished work, available online at `http://www.yorku.ca/mack/RN-TextEntrySpeed.html`, 2002.

[74] I. S. MacKenzie: "KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques," in *Mobile HCI '02: Proceedings of the 4^{th} International Symposium on Mobile Human-Computer Interaction*, pp. 195–210, 2002.

[75] A. Bjorck: *Numerical Methods for Least Squares Problems*, SIAM, 1996.

[76] C. R. Rao, H. Toutenburg, A. Fieger, C. Heumann, T. Nittner, and S. Scheid: *Linear Models: Least Squares and Alternatives*, Springer, 1999.

[77] Divaina newspaper, "Divaina Online Edition," `http://www.divaina.com/`.

[78] R. A. Wagner and M. J. Fischer: "The String-to-String Correction Problem," *Journal of the ACM (JACM)*, Vol. 21, No. 1, pp. 168–173, 1974.

[79] UCSC Sinhala Corpus BETA. (Retrieved March 17th, 2008), University of Colombo School of Computing, Language Technology Research Laboratory, 2008.

[80] Y. Haralambous: "A Sinhalese TEX System," available at `http://userweb.pdn.ac.lk/~nimalr/sinhala/lreport.pdf`, 1994.

[81] P. Dharmasena: "'samanala' Transliteration Scheme Version 2," available at `http://www.ocs.mq.edu.au/~vsaparam/samanala.pdf`, 1996.

[82] A. Wasala, R. Weerasinghe, and K. Gamage: "Sinhala grapheme-to-phoneme conversion and rules for schwa epenthesis," in *Proceedings of the COLING/ACL on Main conference poster sessions*, pp. 890–897, 2006.

[83] I. S. MacKenzie and K. Tanaka-Ishii: "Text entry using a small number of buttons," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 105–121, Morgan Kaufmann, 2007.

[84] Zi corporastion: eZiText™, `http://www.zicorp.com/eZiText.htm`.

[85] D. A. Huffman: "A Method for the Construction of Minimum-Redundancy Codes," in *Proceedings of the IRE*, pp. 1098–1102, 1952.

[86] R. de la Briandais: "File Searching Using Variable Length Keys," in *Proceedings of the Western Joint Computer Conference*, pp. 295–298, 1959.

[87] E. Fredkin: "Trie memory," *Communications of the ACM*, Vol. 3, No. 9, pp. 490–499, 1960.

[88] D. Knuth: "Section 6.3: Digital Searching." in *The Art of Computer Programming, Volume 3: Sorting and Searching*, Third Edition. Addison-Wesley, pp. 492–512, 1997.

[89] D. Knuth: "Section 2.3: Trees," in *The Art of Computer Programming: Fundamental Algorithms*, Third Edition. Addison-Wesley, pp. 308–423, 1997.

[90] L. R. Rabiner: "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257 – 286, 1989.

[91] Y. Ehara and K. Tanaka-Ishii: "Multilingual text entry using automatic language detection." in *IJCNLP 2008: Third International Joint Conference on Natural Language Processing*, pp. 441–448, 2008.

[92] K. Tanaka-Ishii: "Language Model for Text Entry," in I. S. MacKenzie and K. Tanaka-Ishii (Eds.), *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 27–45, Morgan Kaufmann, 2007.

[93] J. G. Cleary, W. J. Teahan, and I. H. Witten: "Unbounded length contexts for PPM," *Proceedings of Data Compression Conference (DCC '95)*, pp. 52–61, 1995.

[94] J. G. Cleary and I. H. Witten: "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, Vol. 32, pp. 396–402, 1984.

[95] J. Hasselgren, E. Montnemery, P. Nugues, and M. Svensson: "HMS: A Predictive Text Entry Method Using Bigrams," in *Proceedings of the Workshop on Language Modeling for Text Entry Methods, 10th Conference of the European Chapter of the Association of Computational Linguistics*, pp. 43–49, 2003.

[96] T. Magnuson and S. Hunnicutt: "Measuring the effectiveness of word prediction: The advantage of long-term use," TMH-QPSR, Vol. 43, No. 1, pp 57–67, 2002.

[97] U. S. Sannasgala: "Akṣara vinyāsaya," in U. S. Sannasgala and A. Perera (Eds.), *Vyākaraṇa Vimansāva*, pp. 427–448, Rasangani Pradeepika Kodikara, 1999. [In Sinhala]

Appendix A

Most Frequent Sinhala Characters and their Occurrence Probabilities

	%		%		%		%		%		%
ය	4.52	ති	1.23	ල්	0.60	ආ	0.45	උ	0.36	ගැ	0.29
ව	4.19	ග	1.11	මි	0.59	තා	0.44	වු	0.36	දා	0.28
න්	4.06	සි	1.01	යා	0.58	නු	0.44	නො	0.35	නැ	0.27
ම	3.49	නි	0.90	රි	0.58	ඔ	0.44	හු	0.35	පැ	0.27
න	3.35	ස්	0.89	වෙ	0.58	වැ	0.43	ද	0.35	ෂ	0.26
ක	2.98	එ	0.89	ළ	0.57	ලි	0.42	ශ	0.35	කො	0.26
ර	2.73	බ	0.88	කා	0.56	කු	0.41	පා	0.35	තේ	0.26
ට	2.54	ම්	0.87	පි	0.56	දෙ	0.41	ගෙ	0.34	දැ	0.25
ස	2.11	කි	0.86	යෙ	0.54	ජ	0.41	ධ	0.34	ද්	0.24
ත	1.95	තු	0.77	ං	0.54	හැ	0.41	යු	0.32	ගී	0.24
ත්	1.91	යි	0.75	නා	0.53	සු	0.40	ට්	0.32	ළි	0.23
අ	1.76	ශ්‍ර	0.73	හා	0.52	මේ	0.40	ඒ	0.32	තෙ	0.22
ක්	1.72	හි	0.72	දි	0.51	මු	0.40	ලා	0.32	රී	0.22
ප	1.65	ඇ	0.72	යේ	0.50	සා	0.39	ඩ	0.31	හා	0.20
ල	1.62	වා	0.71	දු	0.49	වේ	0.38	වූ	0.30	හෙ	0.20
ද	1.61	ර්	0.68	ගේ	0.49	වී	0.37	සේ	0.30	ශ්‍රී	0.19
වි	1.34	මා	0.64	මෙ	0.47	ප්‍ර	0.37	පු	0.29	දේ	0.19
හ	1.27	රු	0.64	ඉ	0.46	දී	0.36	රා	0.29	කෙ	0.19

	%		%		%		%		%		%
ති	0.18	ජා	0.10	මො	0.06	යැ	0.04	ඨ	0.03	දැ	0.02
පෙ	0.18	සෙ	0.10	කු	0.06	ශ්‍ය	0.04	ජ්‍ය	0.03	කූ	0.02
ලෙ	0.18	ගු	0.10	ෂා	0.06	ඔ	0.04	තෝ	0.02	බ	0.02
ණු	0.16	ථ	0.10	ළේ	0.06	ශේ	0.04	ණෙ	0.02	ඥ	0.01
බු	0.16	සී	0.10	හේ	0.06	බෝ	0.04	ණී	0.02	භී	0.01
මැ	0.16	ලු	0.10	බො	0.06	කෝ	0.04	ළෙ	0.02	හී	0.01
ථා	0.15	ච	0.10	රේ	0.06	ණේ	0.04	ටී	0.02	පැ	0.01
බැ	0.15	ට	0.10	රෝ	0.06	ඇ	0.04	ජ	0.02	දී	0.01
ධා	0.15	දු	0.10	ත්‍ය	0.06	කේ	0.04	ඥා	0.02	ෂ්‍ය	0.01
සැ	0.15	ඩු	0.10	රූ	0.06	ජී	0.04	භි	0.02	බැ	0.01
බි	0.14	රෙ	0.09	හි	0.06	ත්‍රි	0.04	ජා	0.02	වෝ	0.01
එ	0.14	ශ්	0.09	කෘ	0.06	ග්‍ර	0.04	නෝ	0.02	වෛ	0.01
තැ	0.14	තු	0.09	ඹ	0.06	පී	0.04	ඝ	0.02	ධී	0.01
වු	0.14	බෙ	0.09	පේ	0.05	පූ	0.04	ඛ	0.02	දූ	0.01
ඩි	0.14	වි	0.09	ලෝ	0.05	ටා	0.03	ත්‍රී	0.02	පෝ	0.01
පො	0.14	ලේ	0.09	ළා	0.05	ටි	0.03	ධ්‍යා	0.02	ක්‍රී	0.01
කී	0.14	තී	0.08	නැ	0.05	ප්‍රා	0.03	කෑ	0.02	ළො	0.01
හෝ	0.14	හ	0.08	තො	0.05	ෂි	0.03	ද්‍ය	0.02	ජ්	0.01
ව	0.13	ෂ්	0.08	ධ්‍ය	0.05	ශ්‍රී	0.03	වෘ	0.02	තෘ	0.01
බා	0.13	ශා	0.07	යො	0.05	ප්‍රු	0.03	මෝ	0.02	ප්‍රි	0.01
තේ	0.13	ගා	0.07	ද්‍යා	0.05	බූ	0.03	ශ්‍ර	0.02	ක්‍ෂ	0.01
ලැ	0.12	වා	0.07	සො	0.05	ලූ	0.03	ෂු	0.02	වී	0.01
ධි	0.12	ජී	0.07	ශි	0.04	ශී	0.03	සැ	0.02	හ්	0.01
යෝ	0.12	ණා	0.07	දි	0.04	දො	0.03	ටෙ	0.02	ත්‍යා	0.01
ඩා	0.12	ඩෙ	0.07	ව්‍ය	0.04	ව්‍යා	0.03	ඵ	0.02	තූ	0.01
කැ	0.12	ඍ	0.07	බී	0.04	රො	0.03	ටෙ	0.02	ප්‍රේ	0.01
භ	0.11	ලී	0.07	ශී	0.04	බ	0.03	ග්	0.02	භු	0.01
හො	0.11	ප්	0.07	මූ	0.04	න්‍ය	0.03	සෝ	0.02	ග්‍ය	0.01
ගො	0.11	ච	0.07	ඊ	0.04	ඹු	0.03	ගෝ	0.02	දෝ	0.01
ණ්	0.11	ක්‍රි	0.07	සූ	0.04	ඩ	0.03	බෞ	0.02	ග්‍රා	0.01
තෙ	0.11	මී	0.07	ලො	0.04	ඳු	0.03	හු	0.02		

Appendix B

Questionnaire

ප්‍ර ශ් නා ව ලි ය Questionnaire

ඔබගේ නම: Name:		වයස අවු.: Age:		ස්ත්‍රී/පුරුෂ භාවය: Sex:	

1. ඔබ යතුරු ලියනයක් හෝ පරිගණකයක් භාවිතා කොට ඉංග්‍රීසි අකුරු ලියා ඇත්තේ ද?
   Have you ever typed English text using a computer or a typewriter?

2. ඔබ යතුරු ලියනයක් හෝ පරිගණකයක් භාවිතා කොට සිංහල අකුරු ලියා ඇත්තේ ද?
   Have you ever typed Sinhala text using a computer or a typewriter?

3. පහත පෙනෙන ඡේදය ඉංග්‍රීසි **අකුරෙන්** ලියන්න.
   Transliterate the following sentence into Roman characters.

   මෙය දරන්නාට අවහිර බාධාවලින් තොරව නිදහසේ ගමන් කිරීමට සහ අවශ්‍යවන ආධාර ද ආරක්ෂාව ද සලස්වා දෙන ලෙසත් අදාල වගකීම් දරන සියලු දෙනාගෙන්ම ශ්‍රී ලංකා ප්‍රජාතාන්ත්‍රික සමාජවාදී ජනරජයේ ජනාධිපති ඉල්ලුම්කර ද අපේක්ෂාකර ද සිටී.[1]

4. පහත දැක්වෙන්නේ පරිගණකය භාවිතා කොට සිංහල අක්ෂර ලිවීම සඳහා යෝජිත ක්‍රම සතරකි. එම ක්‍රම අතුරින් වඩා පහසු ලිවීමේ ක්‍රමය තේරීම සඳහා, එක් එක් ක්‍රමයට ලකුණු සියයෙන් කොපමණ ලකුණු ප්‍රමාණයක් ඔබ ලබා දෙන්නේ දැයි සටහන් කරන්න. ක්‍රම දෙකක් සඳහා සමාන ලකුණු ලබා නොදිය යුතුය. ලකුණු ලබා දීමට ප්‍රථමයෙන් ක්‍රම සතරම හොඳාකාරව අධ්‍යයනය කරන්න.

   Four Sinhala input method proposed to be used in computers. An example of each

[1]This sample text has been extracted from the passport of The Democratic Socialist Republic of Sri Lanka.

input system is given below. After studying them well, please rate each input system from a viewpoint of "easiest-to-input," on a scale of 1 to 100.

A)		B)		C)		D)	

A)

මෙ	ය	ද	ර	න්	නා	ට	අ	ව	හි	ර		
me	ya	dxa	ra	n	naa	ta	a	$\overset{va}{wa}$	hi	ra		
බා	ධා	ව	ලි	න්	තො	ර	ව	නි	ද	හ	සේ	
baa	dxhaa	$\overset{va}{wa}$	li	n	txo	ra	va	ni	dxa	ha	see	
ග	ම	න්	කි	රී	ම	ට	ස	හ				
ga	ma	n	ki	rii	ma	ta	sa	ha				
අ	ව	ශ්‍ය	ව	න	ආ	ධා	ර	ද				
a	va	$\overset{sxya}{zya}$	$\overset{va}{wa}$	na	aa	dxhaa	ra	dxa				
ආ	ර	ක්	ෂා	ව	ද	ස	ල	ස්	වා	දෙ	න	
aa	ra	k	shaa	$\overset{va}{wa}$	dxa	sa	la	s	$\overset{vaa}{waa}$	dxe	na	
ලෙ	ස	ත්	අ	දා	ල	ව	ග	කී	ම්	ද	ර	න
le	sa	tx	a	dxaa	la	$\overset{va}{wa}$	ga	kii	m	dxa	ra	na
සි	ය	ලු	දෙ	නා	ගෙ	න්	ම	ශ්‍රී	ල	ං	කා	
si	ya	lu	dxe	naa	ge	n	ma	$\overset{sxrii}{zrii}$	la	/n	kaa	
ප්‍ර	ජා	තා	න්	ත්‍රි	ක	ස	මා	ජ	වා	දී		
pra	jaa	txaa	n	txri	ka	sa	maa	ja	$\overset{vaa}{waa}$	dxii		
ජ	න	ර	ජ	යේ	ජ	නා	ධි	ප	ති			
ja	na	ra	ja	yee	ja	naa	dxhi	pa	txi			
ඉ	ල්	ලු	ම්	ක	ර	ද	අ	පේ	ක්	ෂා	ක	ර
i	l	lu	m	ka	ra	dxa	a	pee	k	shaa	ka	ra
ද	සි	ටී	.									
dxa	si	tii	.									

B)

මෙ	ය	ද	ර	න්	නා	ට	අ	ව	හි	ර
me	ya	dha	ra	n	naa na)	ta	a	va wa	hi	ra

බා	ධා	ව	ලි	න්	තො	ර	ව	නි	ද	හ	සේ
baa ba)	Dhaa Dha)	va wa	li	n	tho	ra	va wa	ni	dha	ha	sea se) sei

ග	ම	න්	කි	රී	ම	ට	ස	හ
ga	ma	n	ki	rii ri) rie ree	ma	ta	sa	ha

අ	ව	ශ්‍ය	ව	න	ආ	ධා	ර	ද
a	va wa	shYa	va wa	na	aa a)	Dhaa Dha)	ra	dha

ආ	ර	ක්	ෂා	ව	ද	ස	ල	ස්	වා	දෙ	න
aa a)	ra	k	Shaa Sha)	va wa	dha	sa	la	s	vaa va) waa wa)	dhe	na

ලෙ	ස	ත්	අ	දා	ල	ව	ග	කී	ම්	ද	ර	න
le	sa	th	a	dhaa dha)	la	va wa	ga	kii ki) kie kee	m	dha	ra	na

සි	ය	ලු	දෙ	නා	ගෙ	න්	ම	ශ්‍රී	ල	ං	කා
si	ya	lu	dhe	naa na)	ge	n	ma	shrii shri) shrie shree	la	\n	kaa ka)

ප්‍ර	ජා	තා	න්	ත්‍රි	ක	ස	මා	ජ	වා	දී
pra	jaa ja)	thaa tha)	n	thri	ka	sa	maa ma)	ja	vaa va) waa wa)	dhii dhi) dhie dhee

ජ	න	ර	ජ	යේ	ජ	නා	ධි	ප	ති
ja	na	ra	ja	yea ye) yei	ja	naa na)	Dhi	pa	thi

ඉ	ල්	ලු	ම්	ක	ර	ද	අ	පේ	ක්	ෂා	ක	ර
i	l	lu	m	ka	ra	dha	a	pea pe) pei	k	Shaa Sha)	ka	ra

ද	සි	ටී	.
		tii	
		ti)	
		tie	
dha	si	tee	.

C)

ෙ	ම	ය	ද	ර	න්	න	ා	ට	අ	ව	හි	ර			
@	m	y	q	r	n˜	n	`	t	a	v	hQ	r			
බ	ා	ධ	ා	ව	ලි	න්	ෙ	ත	ා	ර	ව				
b	`	{	`	v	lQ	n˜	@	w	`	r	v				
නි	ද	හ	ෙ	ස්	ග	ම	න්	කි	රී	ම	ට	ස	හ		
nQ	q	h	@	s˜	g	m	n˜	kQ	rW	m	t	s	h		
අ	ව	ශ්‍ය	ව	න	අ	ා	ධ	ා	ර	ද					
a	v	X&	v	n	a	`	{	`	r	q					
අ	ා	ර	ක්	ෂ	ා	ව	ද	ස	ල	ස්	ව	ා	ෙ	ද	න
a	`	r	k˜	;	`	v	q	s	l	s˜	v	`	@	q	n
ෙ	ල	ස	ත්	අ	ද	ා	ල	ව	ග	කී	ම්	ද	ර	න	
@	l	s	w˜	a	q	`	l	v	g	kW	m|	q	r	n	
සි	ය	ලූ	ෙ	ද	න	ා	ෙ	ග	න්	ම	ශ්‍රී				
sQ	y	lE	@	q	n	`	@	g	n˜	m	XYW				
ල	ං	ක	ා	ප්‍ර	ජ	ා	ත	ා	න්	ත්‍රි	ක				
l	A	k	`	pY	j	`	w	`	n˜	wYQ	k				
ස	ම	ා	ජ	ව	ා	දී	ජ	න	ර	ජ	ෙ	යේ			
s	m	`	j	v	`	qW	j	n	r	j	@	y˜			
ජ	න	ා	ධි	ප	ති	ඉ	ල්	ලූ	ම්	ක	ර	ද			
j	n	`	{Q	p	wQ	i	l˜	lE	m|	k	r	q			
අ	ෙ	ප්	ක්	ෂ	ා	ක	ර	ද	සි	ටී	.				
a	@	p˜	k˜	;	`	k	r	q	sQ	tW	.				

D)

ෙ ම ය ද ර න් න ා ට අ ව හි ර	f u h o r ka k d g w j ys r
බ ා ධ ා ව ලි න් ෙ ත ා ර ව	n d O d j ,s ka f ; d r j
නි ද හ ෙ ස් ග ම න් කි රී ම ට ස හ	ks o y f ia . u ka ls rS u g i y
අ ව ශ්‍ය ව න අ ා ධ ා ර ද	w j YH j k w d O d r o
අ ා ර ක් ෂ ා ව ද ස ල ස් ව ා ෙ ද න	w d r la I d j o i , ia j d f o k
ෙ ල ස ත් අ ද ා ල ව ග කී ම් ද ර න	f , i ;a w o d , j . lS ua o r k
සි ය ලු ෙ ද න ා ෙ ග න් ම ශ්‍රී	is h ,q f o k d f . ka u Y`S
ල ං ක ා ප්‍ර ජ ා ත ා න් ත්‍රි ක	, x l d m` c d ; d ka ;`s l
ස ම ා ජ ව ා දී ජ න ර ජ ෙ ය්	i u d c j d oS c k r c f ha
ජ න ා ධි ප ති ඉ ල් ලු ම් ක ර ද	c k d Os m ;s b ,a ,q ua l r o
අ ෙ ප් ක් ෂ ා ක ර ද සි ටී .	w f ma la I d l r o is gS ’

Appendix C

Input Variation Table

phoneme	Input sequences and frequencies											
අ(=a)	a	15609	e	522							*<null>*	1768†
ඉ(=i)	i	4416	e	13	y	11					*<null>*	1*
න්(=n)	n	3976	nn	64								
ව(=v)	v	2356	w	1258	vu	25	wu	24	u	14		
ය්(=y)	y	2808	iy	18								
ආ(=ā)	a	1712	aa	1354	ar	4					*<null>*	323†
ක්(=k)	k	3287	c	13	kk	11	ck	2				
ම(=m)	m	2961	n	1*								
ත්(=t)	th	2040	x	500	tx	318	tt	1*	t	306†		
ර්(=r)	r	3054	ru	132								
උ(=u)	u	2401	oo	24							*<null>*	1*
ස්(=s)	s	2268	z	1*	c	1*						
එ(=e)	e	1796									*<null>*	1*
ද්(=d)	d	1526	q	186	dx	36	dh	11	dd	1		
ප්(=p)	p	1690	pp	3								
හ්(=h)	h	1453										
ල්(=l)	l	1518	ll	25								
ඇ(=æ)	e	791	z	218	a	126	ae	91			*<null>*	1*
ඒ(=ē)	e	1264	ee	339	ei	8	ay	3	a	1*	*<null>*	1*

†Constant conversion probabilities

*Constant conversion frequencies

ට(=ṭ)	t	1414										
ග්(=g)	g	1250	gg	1								
ඊ(=ī)	i	762	ii	243	ee	62	y	2	ie	2	e	1^*
											<null>	1^*
බ(=b)	b	903	bb	2								
ඔ(=o)	o	590									*<null>*	1^*
ණ්(=ṇ)	n	516	nx	16								
ළ්(=ḷ)	l	350	lx	6								
ශ්(=ś)	s	243	sh	129	z	22	sx	11				
ඕ(=ō)	o	332	oo	67	oe	1					*<null>*	1^*
ජ්(=j)	j	400	jj	1^*								
ඩ(=ḍ)	d	340	dd	2								
ධ(=dh)	d	434	dh	22	q	19	dxh	10	dd	2	qh	2
ඌ(=ū)	u	132	uu	42	oo	2					*<null>*	1^*
ෂ්(=ṣ)	s	180	sh	106								
ං(=ṃ)	n	231	/n	12	ng	9	nn	2				
ඳ(=n̆d)	d	154	nd	56	/dx	16	q	16	ndx	4	/d	2
ච(=c)	c	93	ch	56								
භ්(=bh)	b	123	bh	93	bb	1						
ථ්(=th)	th	62	txh	41	x	7	xh	2	t	$9^\dagger$		
ඈ(=ǣ)	e	45	aee	25	ee	23	zz	14	ae	8	a	5
	z	5	aa	4							*<null>*	1^*
ඟ්(=n̆g)	ng	23	g	23	/g	3						
separater	/-	2									*<null>*	78
ඹ(=m̆b)	b	27	mb	8	/b	2						
ඖ(=au)	au	23	o	8	oo	3	ou	1	ooo	1		
ඡ්(=ch)	ch	21	c	10	j	1						
connector	/+	2									*<null>*	7
ඨ්(=ṭh)	t	15	th	4								

ඬ(=ňḍ)	/d	5	d	3	nd	1		
ඥ(=jñ)	n	21	gn	9	cx	2	j/c	1
ඛ(=kh)	k	19	kh	3	c	1		
ඓ(=ai)	ai	13	i	10				
ඝ(=gh)	g	7	gh	1				
ඤ(=ñ)	/c	2	n	2				
ෆ(=f)	f	17	ph	1				
ඵ(=ph)	p	30	ph	6				
ඪ(=ḍh)	dh	3						
ඞ(=ṅ)	n	2	/k	1				
ඍ(=ṛ)	r	4	iru	3	ri	2	rx	1
ඣ(=jh)	j	2	jh	1				
ඃ(=ḥ)	h	2	hx	1				
ඎ(=ṝ)	iru	1	rxx	1	iruu	1	r	1
ඦ(=ňj)	/j	1	j	1	nj	1		
ඏ(=ḷ)	lxx	1	ilu	1				
’	’	1						
ඐ(=ḹ)	ilu	1	lxxx	1	iluu	1		
ක්‍යූ(=kyū)	q	1*						
උව(=uva)	ua	1						
එස්(=es)	s	1*						
ඩබ්(=ḍab)	w	1*						
කේ(=kē)	k	1*						
බී(=bī)	b	1*						
එන්(=en)	n	1*						
එල්(=el)	l	1*						
එම්(=em)	m	1*						
ආර්(=ār)	r	1*						
එච්(=ec)	h	1*						

ඉය්(=iy)	i	7				
වයි(=vayi)	y	1^*				
ස්‍ර(=sr)	rx	1	iru	1	ri	1
ජී(=jī)	g	1^*				
එක්ස්(=eks)	x	1^*				
එෆ්(=ef)	f	1^*				
වු(=vu)	u	15	v	4		
යි(=yi)	i	428	y	1^*		
යු(=yu)	u	36				
වී(=vī)	v	1^*				
ටී(=ṭī)	t	1^*				
වූ(=vū)	u	11	uu	2		
සෙඩ්(=seḍ)	z	1^*				
ජේ(=jē)	j	1^*				
ඩබ්ලිව්(=ḍabliv)	w	1^*				
යූ(=yū)	u	1	uu	1		
ඩී(=ḍī)	d	1^*				
යී(=yī)	i	3	y	$5^\dagger$		
පී(=pī)	p	1^*				
අඉ(=ai)	i	1^*				
සී(=sī)	c	1^*				