TEXT INPUT METHODS
FOR INDIAN LANGUAGES

By

Sowmya V.B.

200607014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Science (by Research)
in
Computer Science & Engineering

Search and Information Extraction Lab

Language Technologies Research Center

International Institute of Information Technology

Hyderabad, India

September 2008

Dedicated to all those people, living and dead, who are directly or indirectly responsible to the wonderful life that I am living now.

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled " **Text input methods for Indian Languages** " by **Sowmya V.B. (200607014)** submitted in partial fulfillment for the award of the degree of Master of Science (by Research) in Computer Science & Engineering, has been carried out under my supervision and it is not submitted elsewhere for a degree.

Date

Advisor :
Dr. Vasudeva Varma
Associate Professor
IIIT, Hyderabad

Acknowledgements

I would like to first express my gratitude to my advisor Dr Vasudeva Varma, for being with me and believing in me throughout the duration of this thesis work. His regular suggestions have been greatly useful. I thank Mr Prasad Pingali for his motivation and guidance during the intial phases of my thesis. I thank Mr Bhupal Reddy for giving me the first lessons in my research. I entered IIIT as a novice to Computer Science in general and research in particular. I thank all those faculty members who have taken my first semester courses - Dr Vasudeva Varma, Dr Rajeev Sangal and Dr Dipti Misra Sarma for making me decide to take this direction, which has been a memorable journey throughout.

Thesis acknowledgements is a place to thank not only the academic help but also all the others who unknowingly helped me coup up with the pressure and overcome it. Without Pratibha's help, giving a shape to this thesis draft would have been impossible. Her mysterious confidence in me gave me a lot of moral support. I thank Rahul, Swathi, Santosh and Uma who have helped me in ways more than one, all through these days here. I thank my hostel mates Shaheda,Neeba and Prakani for bearing with my sickening humor all the while. I thank all those people who have constantly reminded me to work, showing their extra ordinary concern, by asking - "Hows your thesis going?" and "How many more days will you take?"

I thank all my labmates and my test users for their support in getting my work done. I thank Saras for his help in learning about statistical significance. Sandeva deserves a special mention for the chats and mails that went on between us on text input. I thank Babji especially, for every help.

I should have mentioned this first, but mentioning now nevertheless. I would

like to thank the users of DC++, all the FTP server owners, IIIT library and Ashakiran and the Internet for all the education and entertainment that I received from them. Life would have been different without them. I thank the Coffee Shop people for providing life saving Tea at many points of time in these 2.5 years. I thank the IIIT main road and HCU for all the ”lost in thought” walks which gave me some insights in to my work periodically. It will be ungrateful if I forget to thank all the admin, security and house keeping staff, who made life easy at IIIT, remaining in invisible mode forever.

Last, but not the least, I thank mom, Halley and Sriram - for being my invisible sources of moral and mental support.

Abstract

With the rapid spread of Indian language content on the web, user input oriented applications in Indian languages are also becoming popular. There is an increasing need for efficient and user friendly input interfaces to type in Indian languages. However, despite the obvious need and significance, text input has not been studied and researched so far, in the context of Indian languages. In this thesis, we study the problem of *text input for Indian languages*, by taking Telugu as the test case. Owing to the similarity among Indian languages, the results of this work can be applied with necessary changes, to other Indian languages as well. We have first classified data input systems in to four categories - keyboard layouts, input method editors, prediction systems and soft keyboards.

In the first category, we have designed some keyboard layouts for Telugu, based on a set of design principles formed after a study of the relevant literature. We have then evaluated them based on the set of evaluation measures and concluded that our designs work better than the existing solutions, thereby providing efficient alternatives to the present day Telugu keyboard. In the context of input method editors, we have studied the different mappings available to convert between Indian languages and Roman script, and compared them with our new mapping, in an attempt to find out a mapping closest to human intuition. We have concluded that though there is no significant difference in performance with different mappings, Rice Transliteration Scheme has a better ease of remembrance compared to other mappings. We have studied the problem of text prediction for Indian language input

and proceeded towards developing transliteration based prediction approach to text input in Telugu. We have proved that a simple approach like edit-distance based text prediction works very well as a text input method and offers a light weight and efficient system. Finally, we have worked on the design and evaluation of soft keyboards for Telugu. Our designs worked better than the existing soft keyboards for Telugu. Thus, we have successfully proposed efficient text input methods for Telugu which can be good alternatives to existing input methods.

Publications

- Sowmya V.B. and Vasudeva Varma, "Design and Evaluation of soft keyboards for Telugu", To appear in *Proceedings of ICON'08: 6th International Conference on Natural Language Processing*, Pune, December 2008.
- Sowmya V.B. and Vasudeva Varma, "Transliteration based text input methods for Telugu", To appear in *Proceedings of ICCPOL'09: 22and International Conference on the Computer Processing of Oriental Languages*, Hongkong, March 2009.
- Communicated: Sowmya V.B. and Vasudeva Varma, "Novice user experience on soft keyboards for Telugu", Australasian Conference on Computer and Human Interaction, 2008.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The widespread use of internet and its increased availability day by day made the number of Indian internet users grow at a quick pace. With this, the need for Indian language computing tools is also increasing, so that the needs of demands of the respective users can be met. The problems related to Indian language computing vary from input related problems to data processing problems, output display and storage related problems. Localization of computing devices is another issue involved in this context. Villages equipped with local language computers can bring about significant impact in taking computing to the rural majority in India. Working towards obtaining solutions for various persistent problems in Indian language computing is one of the main tasks involved in achieving this goal.

Text input is one of main problems of Indian language computing as data input is a major form of user interaction between the user and the computer. Though text input encompasses a wide range of input approaches, we restrict ourselves to keyboard and mouse based input in this work. In this thesis, we deal with the problem of text input for Indian languages and propose some efficient solutions, which perform better than the existing options. We have studied the problem of text input in English and other languages, before forming design rationale for Indian language text input methods.

The following sections of this chapter introduce issues in Indian language computing and explain the problems of text input in the case of Indian languages, in this context. Section 1 explains the issues related to Indian language computing in brief. Section 2 discusses the problem of text input in detail. Section 3 explains the problem of text input in the context of Indian languages. Section 4 explains the problem statement of this thesis. Section 5 mentions in brief, the contributions of this thesis. Section 6 concludes this chapter by explaining the organization of this thesis.

1.1 Issues in Indian language computing

Renu Gupta [19] gives an overview of a user perspective to the problem of technology development for Indian languages. All the text related issues in Indian language computing can be classified in to three categories. They are:

- Text input
- Text processing
- Text output and Display

This thesis concentrates on the first of these categories - text input, which is explained in detail in the next section. The other two categories are explained below.

1.1.1 Text processing

Text processing involves all forms of data manipulations as well as the storage related aspects of the data. All the applications like Part-Of-Speech tagging, performing morphological analysis of data, parsing, identifying different variations of the word considered, spell checking and correction and classifying the data will come under this category. Information retrieval and indexing also form a part of data processing related issues.

Tagging refers to the act of giving grammatical classes to the words in a natural language sentence. Chunking refers to dividing the sentence in to syntactically co-related parts of words. Identifying the Parts of Speech tags and Chunk Tags are the important tasks involved in this process. This can be done both using statistical and non-statistical methods. Tagging and chunking data is one of the widely studied areas in natural language processing. In the context of Indian languages too, there has been a lot of work in Part Of Speech (POS) tagging and chunking of text data. [1]

Morphology in linguistics refers to the study of the internal structure of words. Morphological analysis involves getting information like root, lexical category, gender, number, person, tense and other details for each word taken as its input. A morphological analyzer is the tool that performs this analysis on a text. Parsing is the process in which an input sentence is analyzed and a tree structure is assigned to it. Parsing involves two components - a parser and a grammar for the language. Identifying spelling variations of a word is another important text processing task. It helps in retrieving all the variations of a given word. It is majorly useful in tasks like identification of names, which can be spelt in multiple ways. Spell checker is a tool that corrects spelling errors in words. Spell checking The *did you mean?* utility on Google search [2] and auto correct utilities on word processor programs are some of the examples of spell checker programs. However, work in spell checking for Indian languages is under progress and there is no universally accepted spell checker for an Indian language so far.

Text classification and clustering refers to segregating the data in to different groups based on the nature of the text. Retrieval refers to the act of getting the relevant documents for a particular query and indexing refers to the process of parsing and storing data to facilitate

[1] Workshop on Shallow Parsing for South Asian Languages, IJCAI-2007, http://shiva.iiit.ac.in/SPSAL2007/

[2] www.google.com

fast and accurate information retrieval.

Akshar Bharathi et.al. [5] explains in further detail the issues in data processing in Indian languages in their book *Natural language processing: a Paninian perspective*.

1.1.2 Text output and display

Text output and display refers to how the retrieved stored text is displayed to the user. Issues of Fonts and rendering of fonts form a major part of the data display problems.

The issues with display of Indian languages on computers arises primarily because of the orthographic structure of Indian languages, which is reflected in the design of its fonts. Fonts for Indian languages can come in two ways - Unicode [3] and non-Unicode. While Unicode is the industry standard to express texts written in various writing systems, non-unicode fonts refer to that kind of fonts which are proprietary and are not searchable directly. Most of the fonts in general need to be installed on to the system seperately so that text written in this font can be displayed and they are not installed by default.

Font rendering issues related to the display of fonts properly on the system. One of the examples to this scenario relates to the failure of Firefox web browser till recently, in displaying Indic fonts properly. On a windows system, it required additional requirements compared to the default installation, to make the font rendering correct. However, this was rectified in Firefox 3 [4].

[3] http://www.unicode.org

[4] http://www.mozilla.com/en-US/firefox/

1.2 Text input

Text entry is a special form of human computer interaction([28]) and is an important computing task considering the fact that writing forms a major part of our computer based communication. The term *text input* encompasses all forms of input devices used to enter text data like- keyboards, stylii, mice and joysticks. Advanced techniques like eye tracking and hand writing modelling also come under text input. Sears and Jacko in their HCI Handbook [51] presents an introduction to various input methods and techniques for computing usage.

Hand writing recognition is one form of data input in which the computer receives and interprets handwritten input from pen drives, digital cameras and other devices. It generally involves optical character recognition for text processing. A hand writing recognition system involves a pen or stylus for the user to write, a touch sensitive surface and a software to interpret the stylus movements. Eye tracking is another form of data input which interprets the eye position and movement information to understand the user intentions. Dasher is another data input method which enables the user to input text without using keyboard by means of a pointing device. Voice based input is another form of text input. Shapewriter [33] is a text entry software in which the user draws on a graphical keyboard using a pen. Finger touching [5] is a method of mobile text input using a wrist band attached to the user's hand. Several such kind of methods exist for text input on various systems like desktops, laptops, PDAs, mobile phones and other appliances. Eatoni Ergonomics [14] explains in detail various alternative methods of text entry.

Keyboard is the most common method of text input. QWERTY is the most popular keyboard for text input on computers. There are several optimized layouts over QWERTY, which are explained in greater detail in Chapter 2.1. There are other kinds of keyboards with reduced number of keys, that use other features to keep the input rich enough. Such

[5] http://www.techdigest.tv/2007/02/turn_your_finge.html

keyboards are collectively called *Intelligent Keyboards*. Some of the examples of these Intelligent keyboards include *Half QWERTY keyboard, Single hand key card* and other chord based keyboards. Virtual keyboards are another kind of text input. They include various kinds of keyboards like mobile phone keypads, touch-screen keypads, projection keyboards and other stylus based input systems. Hexinput [6] is another onscreen keyboard based text entry which is both deterministic and predictable in nature.

A detailed survey of existing text input methods and the issues related to text input are explained by Poika Isokoski [7]. However, we consider only text input using keyboard and mouse throughout this work. Our target users are desktop/laptop computer users who are the prospective candidates for data entry in Indian languages. Though the work can be extended towards mobile phone input as well, we stick to text input for normal computing purposes throughout this thesis.

1.3 Text input research in Indian languages

Text entry with Indian languages becomes an interesting problem owing to the nature of the Indian language scripts compared to English. Apart from having a larger number of alphabets compared to English, the consonant-vowel combinations generate new symbols in Indian languages, which makes it impossible to use the English keyboard as it is, for data input in Indian languages.

One way to enable input using the same QWERTY keyboard is to create a mapping between English and the target Indian language and use this mapping while typing is done by the user. This kind of applications are usually referred to as *Input method editors*. Another way is to use the QWERTY keyboard directly as a keyboard layout by utilizing the shift

[6] http://www.strout.net/info/ideas/hexinput.html

[7] http://www.cs.uta.fi/ poika/g/node6.html

versions of keys to represent the additional alphabet in Indian languages.

Another possible method is to design specialized keyboards for Indian languages, which will have a number of keys and facilitate typing all possible combinations easily. However, in an era where switching to Roman script occasionally cannot be ruled out, usage of a seperate keyboard can be cumbersome. Hence, keyboards should be designed in such a way that they stick to the standard layout of a Roman keyboard.

Soft keyboard is another form of text input. It is an alternative form of text input in place of a physical keyboard. It can be used in scenarios where the input needs of the user are short and no prior learning is required on the part of the user. With alphabetical ordering, a soft keyboard can be easy for a novice user to scan visually. Other possible option is to use a auto-complete like prediction system, which understands the user intentions in the first few keystrokes and reduces further typing effort from the user's side.

Based on the above mentioned observations, we have classified the problem of text input for Indian languages in to four categories.

1. **Keyboard layouts:** A keyboard layout refers to the physical arrangement of keys on the surface of a keyboard. Depending on the way keys are arranged, there can be various combinations of keyboards. In this work, we classify all QWERTY like keyboards designed both in English as well as Non-English languages under this category. For example, Dvorak keyboard for English and Inscript for Indian languages are some of the examples of Keyboard layouts. A keyboard layout is the most common way to enter data in a language on to a computer.

2. **Input method editors:** These refer to all the systems which use a mapping between two scripts (typically Roman and a target language) to facilitate typing in one language, using the script of another. For eg: Input method editors like SCIM and

Baraha. All the online applications and other word processing tools, which make use of a mapping between English and Indian language to display data in Indian languages are also included under this category. Considering the case of Telugu, there are 56 letters in the alphabet, which need to be entered using 26 English letters. Hence, this kind of text input methods require multiple keystrokes to type a single letter of Indian language alphabet in most cases. Hence, usage of shift key becomes inevitable.

3. **Prediction systems:** These refer to the input systems which attempts to predict the user needs by means of various statistical and non-statistical techniques and generate suggestions for the user as he/she types in using a standard keyboard. The options like T9, Google Suggest etc. are some of the examples of text prediction systems. The aim of a text prediction is to reduce the typing load on the users part. Prediction becomes an important data input method especially in case of Indian languages owing to the possibility of using more than one keystroke per one character of Indian script. Depending on the implementation method, prediction systems are further classified in to different categories which are explained in 2.3.

4. **Soft keyboards:**Soft keyboards refer to those kind of input systems in which data is entered using an alternative input device like mouse or stylus. The term *soft keyboard* refers to any kind of keyboard that serves as an input method in place of an actual hardware keyboard. A soft keyboard can be understood as a software with an image map on the screen, which enables the user to enter data. It can mean anything from a tuoch screen keyboard to a projection keyboard. In this thesis, we consider only mouse-based input approaches. Our intended application area is for data input in Indian languages on computers. Throughout this thesis, we view soft keyboards as an alternative input method which requires little learning on the part of the users, especially friendly to novice users.

1.4 Problem statement

Our goal in this thesis is to study the problem of *text input* for Indian languages and develop efficient and user-friendly input methods. These methods can be applied in various applications involving a front-end with the user. In essence, this work provides solutions for using Indian languages as easily as English on computers and other modern electronic devices.Our thesis problem can be divided in to the following steps.

- Studying the problem of text input in English and Other languages
- Understanding the possibility of applying the existing techniques for Indian languages
- Designing input methods for Indian languages by formulating some design rationale
- Evaluating our designs along with some existing designs

Text input for Indian languages is not a well-studied research problem yet. Though there have been many research studies and related work for text input in case of English and other languages, attempts to use their approaches and principles to the context of Indian languages have been very little. Some tools and applications have been designed in the recent past for Indian languages, though there has been no structured study and evaluation for the same. This thesis, hence performs a study and evaluation of the existing methods in comparision with our designed methods.

1.5 Contributions

We began by classifying different approaches to data input based on the nature of their interaction with the user in Chapter 1.2 and formulated a framework to evaluate each of the approaches. We have performed all our experiments on Telugu, though the design principles are language independent and can be applied to other Indian languages as well.

We have successfully presented alternative solutions to data input, which work efficiently compared to the existing input tools and applications. We have taken Telugu as our test case in all our experiments, though the process can be repeated for other Indian languages as well, without major design changes.

The contributions of this thesis to the area of Indian language computing can be listed as follows:

- A study of different keyboard and mouse based input options for text input in English as well as other languages including Indian languages.
- Design and evaluation of new keyboard layouts for use on desktop/laptop computers. We have proved that they perform better than the existing standard keyboards for Indian languages.
- Development of a transliteration based text input method for Telugu, based on an edit distance based measure, which can provide an efficient alternative means of text input, without putting more load on the system resources.
- Design and evaluation of soft keyboards. We have also evaluated other existing designs and proved that one of our soft keyboard layouts performs the best among all the tested layouts.

1.6 Organization of Thesis

The main focus of this thesis is to solve the problems related to *text intput* in Indian languages. This chapter introduces the problem of text input apart from putting it in the context of other relevant problems in Indian language computing. We have classified approaches to text input in to four categories in this chapter, which will be dealt one by one in the coming chapters. In each of the classified category, we have worked on new designs and evaluated them along with a comparision with already existing designs. The classification

is presented in Section 2. The rest of the thesis is organized as follows:

Chapter 2 explains some of the relevant work in this area. The related work is also classified in to four categories based on our classification of text input approaches. Each section surveys the related work done in English and other non-Indian languages, apart from mentioning the work done in Indian languages, if any. We have also compared different works and tried to understand their merits and de-merits in the context of applying for Indian language text input. This section presents a detailed over view of different text input approaches in Indian as well as non-Indian languages.

Chapter 3 explains the design and evaluation of keyboard layouts for Telugu, which can be applied to other Indian languages. We have compared the designs with the existing standard keyboard layout for Telugu and have proven that our layouts performed better. The chapter begins with explaining the design rationale for our keyboards and then proceeding further to explain our designs in detail. The last part of this chapter deals with the evaluation of the designed keyboards, along with *inscript*, the standard Indian language keyboard on computers.

Chapter 4 introduces the problem of transliteration by comparing various existent mappings between Roman to Indian languages. Apart from the existing mappings, we have proposed a new mapping ”All-low“, which is an all lower case mapping, and evaluated it along with the other mappings. This work forms the background for the design of prediction systems, which is explained in the next chapter.

Chapter 5 explains the problem of text prediction in the context of Indian languages and explains the experiments conducted in *transliteration based text input for Indian languages*. We then explain the evaluation of our prediction method in this chapter. The experiments have been done taking Telugu as the test case. But, with necessary modifications, the

method can be applied to other Indian languages as well.

Chapter 6 explains the design and evaluation of soft keyboards for Telugu, which can be extended to other Indian languages. The term *soft keyboard* refers to any kind of keyboard that serves as an input method in place of an actual hardware keyboard. We explain our design rationale and compare our soft keyboards with the existing soft keyboards in this chapter.

Finally **Chapter 7** covers concludes this thesis by explaining the work done and explaining the contributions of this thesis. This chapter also provides an outline for the future work.

Chapter 2

Related Work

In this chapter, we survey the related work in the area of text input methods. We restrict ourselves to the input methods which cater to the needs of normal desktop computers, since we concentrate on Indian language input for general purpose use on computers, throughout this thesis. We have classified the input methods for computers into four types, as mentioned in chapter 1.2. The classification is listed here again, for convenience.

1. Keyboard layouts
2. Transliterators and Input Method Editors
3. Prediction systems
4. Soft keyboards

The relevant literature in each of these categories is presented in the following sections.

2.1 Keyboard Layouts

Keyboard layout refers to the design of the keyboard used as a means of data input. It refers to the position of keys on the keyboard and the pattern of their arrangement. Different

keyboard layouts have been designed for English language, though QWERTY is the one which gained popularity and eventual widespread use.

2.1.1 English Keyboard Layouts

QWERTY[1] is the most popular English keyboard layout, which is used all over the world today. However, a lot of work has been done in an attempt to design a keyboard which performs better than QWERTY. Altering the key arrangements and testing it with respect to typing speed and hand alternation, several alternative keyboard layouts have been proposed. DVORAK[2] is one such alternative keyboard, which is the best known among them. It is also known as American Simplified Keyboard (ASK) layout. It is claimed to have ergonomical benifits over QWERTY. Several other layouts like - Turkish F-Keyboard, Turkish-Q keyboard have been proposed for Roman alphabet. An AZERTY[3] layout is used in France, Belgium and other neighbouring countries, which differs from QWERTY in some minor ways. A QWERTZ[4] keyboard is used in Germany and other parts of Central Europe, which has many diacritical characters included in the keyboard instead of special characters like brackets. Different countries use a variation of QWERTY like AZERTY or QWERTZ based on their needs and requirements. Frogpad [5] is a one-handed typing enabled keyboard. There are specialised keyboards like Maltron [6], Plum [7] etc designed for commercial use.

Michael Capewell performed a study of alternative keyboard layouts in English ([7]).The Carlpax keyboard layout optimizer project worked towards scientifically studying the problem of design of keyboard layout apart from discussing on the evaluation methodology ([34]) to evaluate various keyboard layouts. Peter M.Klausler worked towards an evolu-

[1]Patent: 207,559 (US) issued August 27, 1878 to Christopher Latham Sholes

[2]http://www.mwbrooks.com/dvorak/dissent.html

[3]http://en.wikipedia.org/wiki/AZERTY

[4]http://en.wikipedia.org/wiki/QWERTZ

[5]www.frogpad.com/

[6]www.maltron.com/

[7]www.plum.bz/

tionary algorithm to discover better keyboard layouts ([41]). Capewell ([8]) also worked towards evolving and evaluating new keyboard layouts through KeyboardEvolve [8] project. Applications like KbdEdit[9] allow the users to create their own keyboard layouts. Microsoft's Keyboard Layout Creator allows the user to create customized keyboard layouts. The Kinesis ([56]) keyboard is another commercially available keyboard layout which is flexible and allows the users to modify keyboard at a hardware level. Arensito([20])is yet another alternative keyboard layout, with a letter arrangement based on statistics. Colemak ([11]) was proposed as an ergonomic alternative to QWERTY and DVORAK.

2.1.2 Non-Roman Keyboard Layouts

The Roman keyboard layout need not suffice to languages with non-Roman scripts owing to the inherent difference in their orthographies. Further, most of the non-roman languages have more number of letters in their alphabet than Roman languages, which makes the need for an alternate keyboard layout for their languages mandatory. While some of them uses the available QWERTY layout customized to their language needs, by arranging their alphabets over the QWERTY keys, some of them have new keyboard layouts with more keys. Languages like Hebrew, Arabic, Greek, Russian, Armenian, Ukranian, Bulgarian and Thai use the available QWERTY structure to suit to their needs.

East-Asian languages like Chinese, Japanese and Korean require Input Method Editors, which transliterate from Roman script to the required script, owing to a possibility of formation of thousands of characters in these languages. Since IMEs can be fit in to a QWERTY keyboard, most of the East-Asian countries have the same QWERTY keyboard. Different layouts exist for Chinese keyboards like - Zhuyin, Cangje and Wubi, which are summarized in [23]. Similarly, Japanese,Korean and Tibetan layouts also have different designs associated with them. They are briefly surveyed in Japanese language Input methods page of Wikipedia ([65]).

[8]http://keyboardevolve.sourceforge.net/

[9]http://www.kbdedit.com/

However, as mentioned above, all these keyboard layouts are essentially QWERTY based, following a definite mapping between QWERTY characters and the respective language characters. In essence, they are transliterator keyboards rather than actual keyboards. Hence, they are called Input Method Editors. [10]

2.1.3 Indian language Keyboard Layouts

Keyboard layouts for Indian languages began with the evolution of type-writers for Indian languages. Inscript ([1]) was the first keyboard layout which was designed for Indian language computing, in the lines of the type writer layout. It was designed keeping in view the two handed typing ability of a human user. The Inscript overlay can be used with any QWERTY keyboard. The Tamilnet99 report [24] listed and explained the design of existing Tamil keyboards. It also suggested some guidelines for designing a Tamil keyboard, which are not language specific and hence can be applied to any language in general. Keylekh [29] is another keyboard designed for text entry in Indic scripts. Keylekh does not follow a standard keyboard structure, but provides a new keyboard structure for data input in Indian languages. However, while Inscript is not very friendly with first time users, Keylekh does not follow a QWERTY structure. Hence, the need for alternative keyboard layouts for Indian languages arises out of the above mentioned short comings of the existing keyboards.

2.2 Transliterators

2.2.1 Transliteration

Transliteration is the process of mapping text written in one language in to another sometimes using a set of rules to map between the two language alphabets, sometimes without them. While the first case is deterministic, the second case is non-deterministic. In this section, we discuss only deterministic transliteration. Non-deterministic transliteration is

[10] http://www.microsoft.com/globaldev/handson/user/IME_Paper.mspx

referred to in this thesis as *Text Prediction*. Transliteration is useful when a user knows a language but does not know how to write its script. It is also useful in case of unavailability of a direct method to input data in a language. Transliteration can also be used for simple encryption. It is also useful in serving as a data input method for languages with a larger alphabet, using a constrained character set. Hence, Transliteration can be understood as a process of entering data in one language using the script of another language. In general, the mapping between source and target language alphabet in a transliteration scheme should be as close as possible to the pronunciation of the source alphabet in the target language. Transliterated text has found widespread use with the growth of internet usage, in the form of chats, mails, blogs and other forms of individual online writing. This kind of transliterated text is often referred by the words formed by a combination of English and the language in to which transliteration is performed, like - Arabish (Arabic + English), Hinglish (Hindi + English), Greeklish (Greek + English) etc.

Transliteration is different from *translation* because, while translation involves conversion of the text from one language to another preserving the meaning, transliteration only maps between scripts of two languages. Transliteration is also different from *transcription* in the sense that while transcription is a mapping between sound to script, transliteration is a mapping between two scripts. In essence, transliteration can be understood as a form of data input. However, depending on various factors like mapping, source-target language pair etc, a word in source language can have more than one possible transliterations in the target language. This is more frequently seen in case of transliteration of proper nouns and other named entities. Transliteration can be both forward and backward. Forward transliteration refers to transliterating from Roman to Target language. Backward transliteration refers to performing the same from other language to Roman script. The word Transliteration is used to mean forward transliteration throughout this thesis.

2.2.2 Transliterators

A Transliterator is a tool that performs Transliteration. While the word "Transliterator" can be used to mean a method of data input as well as a "spelling transliterator", we use the word in the context of data input throughout this thesis. Transliterators as data input methods can be built in various ways, though most of them involve a mapping between the source and target language, which is used for transliteration purposes. In other words, the word "transliterator" is used throughout this section to mean an Input Method Editor (IME). *An input method is an operating system component or program that allows users to enter characters and symbols not found on their input device.*[11]. The tool which performs this on an operating system is called an Input Method Editor. In this section, we survey some of the transliterator tools designed for various non-English languages, including Indian languages.

2.2.3 Transliterators for Non-English Languages

There has been a lot of work in understanding the issues of Transliteration in to Asian languages like Chinese,Japanese and Korean as well as languages like Arabic, Russian, Hebrew, Bulgarian, Ukranian and Belarusian among others. There are a number of free online transliteration services, offering transliteration from Roman to hundreds of languages, following some pre-defined mapping. Keyman, by Tavultesoft[12] is said to support the transliteration to over 500 languages. Linguaseek[13] is a search engine interface powered by Google [14], which provides transliteration from Roman to more than 100 languages including several Indian languages. The International Components for Unicode(ICU) project supports almost all the scripts for transliteration. Various transliteration schemes exist for each of the languages separately, based on their scripts and their relation with Roman scripts. Most

[11] http://en.wikipedia.org/wiki/Input_method_editor

[12] http://www.tavultesoft.com/keyman/

[13] http://www.linguaseek.com/

[14] http://www.google.com

of them are mentioned in [66], sorted according to the language.

2.2.4 Indian Language Transliterators

Romanization of Indian languages -Issues

In case of Indian languages, the consonants have an inherent ”a” sound in the end. Hence, the consonants must inevitably be followed by an ”a” (eg: ka,ga,ja etc) to represent the pronunciation in Roman notation. Indian languages are characterized by more number of letters in the alphabet compared to English. Hence, mapping an Indian language in to Roman notation obviously makes it impossible to represent a letter in Indian language by an equivalent letter in English. So, a combination of English letters are used to represent a single letter from Indian language. Based on this idea, several transliteration schemes have been proposed for Indian language to Roman transliteration. Some of them are surveyed at the wiki page on Devanagari Transliteration [15]. They include International Alphabet of Sanskrit Transliteration (IAST) [16], ISO 15919 [17], Harvard-Kyoto [18],ITRANS [19], Velthius etc. Several other mapping schemes exist, which are briefly mentioned below and explained further in Chapter 4.

Transliterator tools for Indian languages -a survey

With the advent of Unicode and increase of Indian language websites, the need for data entry in Indian languages also arouse. Along with this and increasing interest in Indian language typing, many online transliterator tools came in to existence. While some of them followed standardized mapping scheme, most of them were random mappings, based on the intuition of the designer. The users had to familiarize themselves with the mapping fol-

[15] http://en.wikipedia.org/wiki/Devanagari_transliteration

[16] http://dictionary.sensagent.com/IAST/en-fr/ALEXDC/

[17] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=28333

[18] http://en.wikipedia.org/wiki/Harvard-Kyoto

[19] www.aczoom.com/itrans/

lowed by a particular transliteration tool, to use it comfortably. Some of the online transliterators for Indian languages include HiGopi[20], Gamabhana [21], TypePad [22], Lipikaar[23], the IndiChat plug-in of Yahoo Messenger! etc. Websites like Ibibo and Blogger.com also embedded these tools in to their blogging interface to facilitate users blogging in Indian languages. Offline tools like Baraha[24] and Acharya Editor ([60]) were also developed as text editors in Indian languages. Firefox plugins like Indic-IME and Padma also enable Indian language transliteration for online users. Microsoft has implemented an Input Method Editor(IME) for Indian languages in its Windows operating system. Other IMEs like SCIM also have Indian language support enabled, which can be used with Unix based operating systems too. Other than these, there are a lot of language specific transliteration tools designed only for one or two languages. Lekhini [25] is one such online tool for Telugu. There are numerous such tools for Telugu as well as other Indian languages, designed by enthusiasts and professionals alike. We describe some of the most common mappings followed in these tools in Chapter 4 and evaluate them.

2.3 Text Prediction systems

The term Text Prediction systems refer to the genre of systems, which offer suggestions to the user, based on what they type. They can be understood as a form of typing aid, which will reduce the burden to type on the part of the users. The "auto complete" and "word completion" features are a form of text prediction. Several text prediction utilities have been proposed for English for on-desktop as well as off-desktop computing so far. Prediction systems on mobile phones has been a common phenomenon for English text messaging and other applications in the recent past. Text prediction using statistical and

[20] http://www.higopi.com/ucedit/index.html

[21] http://var-x.com/gamabhana/

[22] http://www.vishalon.net/IndicResources/IndicIME/tabid/244/Default.aspx

[23] www.lipikaar.com

[24] www.baraha.com

[25] http://lekhini.org

machine learning methods has been a topic of recent interest in the case of other languages. In the case of Indian languages, text prediction is considered to be one of the alternatives to text input using a normal QWERTY keyboard. This is because of the presence of more alphabets in Indian languages compared to English. In this section, we survey some of the related literature in this area.

2.3.1 Text prediction systems for English

Predictive text input is commonly used in Mobile phones and auto complete facilities. Some of the popular text prediction systems for English are T9[26], iTap[27], eZiText[28] etc. among several others. T9 combines a group of letters on the phone key with a dictionary of words. It then looks up the dictionary for all the corresponding key press sequences of the user and orders them by frequency. iTap uses a different kind of interface compared to T9. It provides suggestions for word completions after only one key press in all cases while T9 completes custom words. Other words that users have entered can be retrieved after some key presses in T9. While iTap is context dependent, T9 is not. iTap and T9 are compared in performance by David Mackay [29]. eZiText has phrase prediction facility. Thus, it speeds up text entry by predicting multiple words that appear together normally. It can also learn the user's vocabulary.

Vitoria and Abascal ([16]) presents a survey of contemporary text prediction systems and methods followed. Harbusch [21] worked towards developing an adaptive communication aid for ambiguous keyboards. However, this approach of text prediction may not work for non-English languages, especially Indian languages owing to the increased alphabet count and increased number of possible vowel-consonant combinations.

[26] http://www.nuance.com/t9/

[27] http://www.inference.phy.cam.ac.uk/mackay/itprnn/itap/

[28] http://www.zicorp.com/TextEntry.htm

[29] http://www.inference.phy.cam.ac.uk/mackay/itprnn/itap/

2.3.2 Text prediction in non-Indian Languages

eZiText also provides text prediction for 60 different languages, which include complex languages like Chinese and Japanese. Goonetilleke et.al. [18] built , SriShell a predictive text input system for Sinhalese language, based on the presence of a big word list. Chen et.al [9] developed a statistical approach to enable Chinese Pinyin input. Ehara et.al. [10] developed a predictive text entry system for Thai language, which switches between Thai and English using automatic language detection. Thus, Prediction systems have been designed for non-Indian languages using both Learning as well as non-Learning methods. Usage of statistical methods as well as Gazetteer based methods for Named Entity Transliteration has been done by various research groups, which is surveyed in Chapter 5, since we have studied them in the context of a transliteration based text input system.

2.3.3 Indian language Text prediction systems

Text prediction in Indian languages has been a topic of recent interest. Tachyon Technologies [30] uses a learning based prediction engine in its services offered in Indian languages. Google's Indic Transliteration[31] is another prediction system for Indian languages. Google Suggest[32] extended its support to Indian languages now. Ganesh et.al. [46] worked towards developing a HMM and CRF based prediction system for Hindi. All the above mentioned systems with the exception of Google Suggest have incorporated some sort of learning in to their prediction approach. Google suggest made use of a huge wordlist and offered suggestions to the user on each keypress. In our work, we have exploited the availability of a big wordlist and the concept of edit distance to get predictions for the user input. We have also proved the efficiency of our approach in predicting the user input. Our system will be explained in greater detail in Chapter 5.

[30] http://quillpad.in

[31] http://www.google.com/transliterate/indic

[32] http://www.google.com/support/bin/answer.py?answer=106230&hl=en

2.4 Soft Keyboards

Soft keyboards refer to the software that allows the user to input characters without making use of hard keyboards. Soft keyboards use pointing devices like mouse to achieve this purpose. They are also referred to as *on screen keyboards* or *virtual keyboards*. Kolsch et.al.([32]) define soft keyboard as a touch-typing device that does not have a physical manifestation of the sensing areas. That is, ”button” in the keyboard is not in fact a button but is programmed to act as one. While the term ”virtual keyboard” or ”soft keyboard” or “on screen keyboard” can be used to mean anything from optical projection keyboards to stylus based typing, in this work, we use the term in the context of an on screen keyboard for desktops, which can be possibly extended to mobile computing.

2.4.1 Soft keyboards for English

Design of soft keyboards for English has been a widely researched area. Mackenzie and Zhang ([26]) argued that the design of a soft keyboard need not stick to QWERTY or any such general keyboard layout since touch typing skill need not transfer to touch tapping skill of a user. The two handed eyes-free touch typing is different from one handed tapping on a soft keyboard. Hence, comparison with QWERTY performance, of soft keyboards is not fully justified. Zhai and Smith ([70]) worked towards developing a soft keyboard which eases novice users' visual search difficulty without losing movement efficiency for the expert users. This new layout with alphabetical ordering enhanced the novice user performance by 9%. In another work, [49] explained the need to design an optimized soft keyboard and compared it with that of a hard-keyboard. They called it ”Metropolis keyboard”, since it was based on the metropolis random walk algorithm. They also evaluated the performances of four other soft keyboard layouts. Smith and Zhai([55]) have argued that, while optimized keyboards are well suited for expert users, they are difficult to use for novice users. They showed that it is possible to design soft keyboards with a general tendency of an alphabetical ordering of the keys, at minimal movement efficiency cost.

2.4.2 Soft keyboards in other languages

Several online tools exist demonstrating the implementation of soft keyboards for various world languages. LITETYPE [33] is an online tool which provides soft keyboards in 36 world languages. Bengali is the only Indian language included in the list. *ok-board* is another online tool [34] which supports 36 languages, most of them same as those supported by LITETYPE. Other tools like *Touch-it*[35], *Keyman Virtual keyboard software*[36] and *IMTranslator*[37] are available for online use.

2.4.3 Indian Language Soft keyboards

Development of virtual keyboards for Indian languages has been a topic of recent interest. There have been some attempts at designing new virtual keyboards, though there was no structured study and evaluation of the same, yet. Initial attempts have been made to design soft keyboards for Indian languages. Google Inc [38] designed a soft keyboard for use in its Google Indic Transliteration (shown in Chapter 6.4), to edit the predictions that the system gives to its users. Guruji [39] is an Indian language search Engine. It has its own soft keyboard (shown in Chapter 6.5), allowing the users to enter query in the chosen language by using a mouse. INSCRIPT ([1]) (shown in Chapter 6.6) is the keyboard standardized by Department of Electronics, Government of India, in 1986, proposed by Report of the Committee for Standardization of Keyboard Layout for Indian Script Based Computers. The Inscript overlay can be used on any standard QWERTY keyboard. There are soft keyboard implementations too. Google Inc has a soft keyboard gadget, which has this layout. This keyboard was primarily designed for an efficient two-handed text

[33] http://www.litetype.com/

[34] http://ok-board.com/

[35] http://www.brothersoft.com/touch-it-virtual-keyboard-108081.html

[36] http://www.tavultesoft.com/

[37] http://imtranslator.com/

[38] http://www.google.com

[39] http://www.guruji.com/te/

entry. However, the idea behind designing a soft keyboard is opposite to that of a normal keyboard, as mentioned in [70]. Further, the presence of shift key in itself indicates that this keyboard is not well suited to be a soft keyboard. Nevertheless, we are considering this keyboard as a part of our evaluation since it is also used as a soft keyboard implementation. The telugulipi.net[40] provides another soft keyboard which gives all possible variants of the letter, when mouse rolls over the letter. Google proposed another version of soft keyboard (Google's onscreen keyboard gadget), which is a predictive layout, static in nature. This layout is easy to remember since the alphabets are arranged in their natural order.

The above keyboard designs have been in use on the internet. But, there has been no structured study on the design rationale for an Indian language soft keyboard. The aim of the above designs appears to be providing a facility of soft keyboard first and improving on it next, which means there is no structured study or detailed analysis of the problem. Lack of evaluation also makes an authentic comparison among them impossible. Our work aims at forming design rationale for design of soft keyboards based on previous work in English and applying them to design and evaluate soft keyboards for Telugu. However, because of the similarity among Indian languages, this work can be extended to other Indian languages as well.

2.5 Summary

In this chapter, we have surveyed the related work on text input methods. We have classified text input methods in to four categories and classified the literature accordingly. The four categories are:

1. Keyboard layouts
2. Transliterators and Input Method Editors
3. Text prediction systems

[40] http://www.telugulipi.net

4. Soft keyboards

This was further classified in to English, Non-English and Indian languages. Relevant related work has been studied and the issues involved in their design and implementation have been mentioned. The issues dealt with in this thesis have been cursorily mentioned wherever necessary, which will be discussed in detail in the coming chapters.

Chapter 3

Keyboard Layouts

A Keyboard Layout refers to the arrangement of keys on the physical surface of the keyboard. Arrangement implies the placement of physical location of keys on a keyboard. Depending on this arrangement, various keyboards can be evolved. QWERTY is the most popular keyboard layout used worldwide for English typing. Other Roman script based languages used QWERTY variants as their national keyboards. A brief survey of keyboards in English and other languages is presented in Chapter 2.1. Though there are some pre-existing designs for keyboard layouts in Indian languages, the need for new design arose from the fact that while Inscript [1], which became de facto standard for Indian language input on Windows systems is not easy to learn by novice users, layouts like Keylekh [29] and its variations required a seperate keyboard which was different from the standard QWERTY keyboard hardware. In this chapter, we explain our design rationale, describe the design of our keyboard layouts for Indian languages and evaluate them.

3.1 Design Principles

Based on a study of the existing work in English language [34, 7] and relevant studies for Indian languages [29, 24], we have formed a set of design rationale for the design of our keyboards. They are explained below.

- **Alternation between English and Indian language:** Though the purpose of the keyboard is to enable Indian language typing in a convenient manner, occasional shift to English might be inevitable. Some of the cases include typing email addresses or web urls etc. Hence, the keyboard layout should allow the user to switch to English whenever it is possible. Hence, we have tried to stick to the normal keyboard hardware, i.e. the QWERTY keyboard. In this process, we are also eliminating the possibility of purchasing a new keyboard for Indian languages, since the same hardware suffices the need of data input in both the languages.

- **Making use of two hands:** An ideal keyboard layout should make efficient use of the two-handed typing ability of the humans. Hence, the difference in hand usage must be kept as minimum as possible.

- **Easy learning:** The keyboard should be easier to learn and remember.

- **Lesser used alphabets in the corners:** The lesser used alphabets should be placed in corners of the keyboard, since they are least likely to be used frequently.

- **Friendly to first time users:** The problem with existing Indian language keyboards was that they had a greater visual scan time for non-trained users. However, as Norman described in [12], an ideal keyboard should be so ready to use that, you can just walk in, "*search for the letter you want and push the key*".

- **Keeping punctuation markers and numbers intact:** This is preferred since punctuation and numbers can be used independent of the language anywhere.

3.2 Design

Based on the above mentioned design principles, we have come up with three keyboard layouts. There is no specific reason behind the number *3* and a number of layouts can be formed by varying the combinations. Our designs are explained below. The accented

Figure 3.1 Keyboard-1

versions of alphabets are placed in the shift positions of the normal alphabets wherever it is possible and the least possible number of non-alphabet keys on QWERTY are taken to accommodate all the alphabets of Telugu.

3.2.1 Keyboard-1

In this keyboard, all the phonetic variants and vowels are arranged to the left side of the keyboard as much as possible. Since every alternate key is likely to be a phonetic variant key, hand alternation is utilized efficiently in this case. This is easier to remember since some kind of order exists in the key arrangement, as it can be understood from the figure above. Two of the least used alphabets are placed on the top left corner, on the ” ” key with shift and without shift respectively. It is expected to reduce the visual scan time for novice users, owing to the inherent order in the layout. Most of the normal pronunciation symbols and all the number keys are kept intact.

3.2.2 Keyboard-2

This is a slight variation over Keyboard-1. The left and right balance between vowels/phonetic variants and consonants is preserved in this keyboard too, though the order

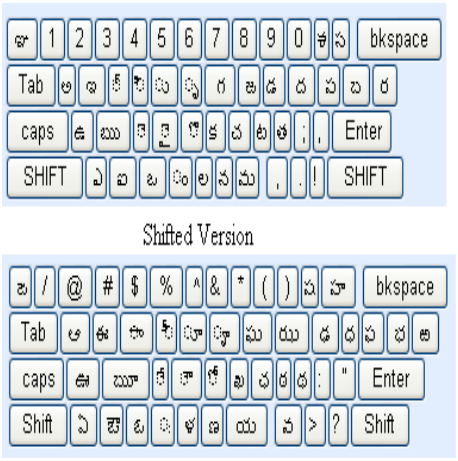

Figure 3.2 Keyboard-2

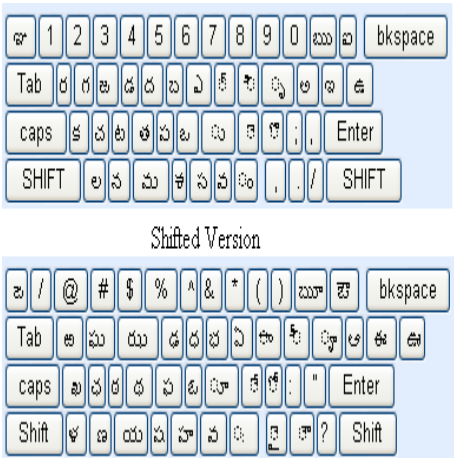

Figure 3.3 Keyboard-3

of occurrence of the alphabets has been changed to some extent. This is more easier to remember since this is more closer to the natural order of the alphabet than Keyboard-1. Like in Keyboard-1, most of the normal punctuation symbols and all the number keys are accessible easily on this keyboard.

3.2.3 Keyboard-3

This design is different from the previous two in the sense that the places of vowels/phonetic variants and consonants are exchanged. While consonants occupy the left side of this keyboard, vowels and phonetic variants occupy the right side. This is closer to the natural order of writing Telugu, which is left to right. The ease of remembrance is kept as close to user cognition as possible in the design, as in the previous two layouts.

3.3 Design from Statistical facts

The above mentioned designs were proposed with the idea of achieving a balance between the ease of learning and remembrance on the users part along with balancing between both the hands while typing. Though all the above mentioned layouts performed better than the previously existent layout, We wanted to obtain a superior layout compared to our designs by using language statistics. By using the crawl index of a search engine, we have obtained an estimate of the character counts of all the consonants, vowels and phonetic variants in Telugu language. Though there might be more data available online as well as offline, we consider this data sample as a considerable representation of the characteristics of the language. Studies have been conducted on the best and worst positions on a standard keyboard before ([42]). We have used their conclusions along with the data available with us, to design an optimized keyboard layout based on the characteristics of a good layout, mentioned by [34]. We have also calculated digraph frequencies, excluding punctuation markers, numbers and space characters. This data is also used in designing the keyboard, to verify the factors like - hand rolling, finger and row movements.

3.3.1 Data Analysis

The crawl data index comprised of a huge index of 1,600,000 words which are listed along with their word counts. From this data, we have extracted the individual character counts.

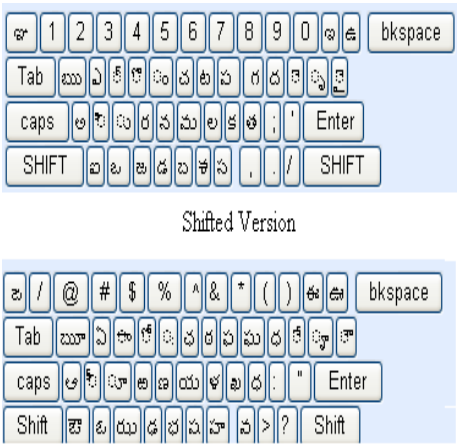

Figure 3.4 Keyboard-4

Since all the alphabets in our keyboard layouts exist in pairs (one for normal key press, one for shift key press), we have combined the character counts of the pairs in to one. Thus, we got the most frequently occurring character list from the available data. We have arranged these letters in the most accessible positions, which are obtained from the results mentioned in [42]. Similarly, we positioned the least occurring alphabets in the most difficultly accessible positions.

3.3.2 The Design

We have designed a keyboard based on the analysis of data, the results of which are explained in the previous section. This keyboard is shown in Figure 3.4. We have tried to place the most frequently and least frequently occuring alphabets in the appropriate positions on the keyboard, without compromising on the ease of remembrance of the layout.

3.4 Evaluation

We have evaluated the keyboard layouts the we have designed, along with the Inscript keyboard based on the typing effort models created by [34] and the evaluations performed by [41] and [63] for English keyboard layouts. The factors that we have considered for our evaluation are:

- Limited use of weak fingers like pinky and ring finger
- Limited use of bottom row and maximum use of home row
- Limited same finger typing
- Balanced hand use
- Hand alternation
- Finger alternation on the same hand
- Avoiding usage of Pinky on non-home rows.

The above mentioned factors add to the score of the layout since they are the requirements of an optimal layout. Considering these factors, we have evaluated the keyboard layouts, by giving them positive or negitive points based on their adherence to the above mentioned factors. The overall score of the layout is the summation of all these scores. The evaluation was performed on a 2500 word test data, collected from Telugu wikipedia pages. The statistics of this evaluation are tabulated below:

3.5 Results

The finger statistics can be visualized in the graph shown in Figure 3.5. Ideally, the graph should be two inverted “U”s, with 0s at fingers 5 and 6. Fingers 5 and 6 represent the thumbs of left and right hands respectively, which are not used except for pressing the space

Keyboard Layout	Left Hand usage	Right Hand usage	Hand balance
Layout 1	8241	9761	1520
Layout 2	8922	9032	110
Layout 3	8218	9218	1000
Layout 4	9910	7662	2248
Inscript	9634	8057	1577

Table 3.1 Hand Usage statistics for different keyboards

Keyboard Layout	Row 0	Row 1	Row 2	Row 3
Layout 1	1093	9841	3225	3843
Layout 2	1095	9170	3165	4524
Layout 3	257	6832	4933	5414
Layout 4	455	6976	8121	2020
Inscript	343	3783	8250	5315

Table 3.2 Row Usage statistics for different keyboards

key. Since we are considering only alphabet, punctuation markers and number strokes, counts for the two thumbs will be zero. In an ideal scenario, the index and middle fingers of both the hands must be the most loaded fingers. Though none of the keyboards achieved the perfect inverted-U's structure, Layout-5 (Inscript) appears to be a winner with this evaluation, since it achieves a near inverted-U shape with both hands. However, Layout-2 appears to perform the best with left hand, while Layout-3 performs the best with right hand.

The row statistics can be visualized in the graph shown in Figure 3.6. According to the available literature, home row (Row-2) should be the place where most used alphabets are kept. Considering that factor, only Layout-4 and Inscript have the maximum number of strokes on the home row. However, Layout-4 shows a better performance in a balanced

Keyboard Layout	1	2	3	4	5	6	7	8	9	10
Layout 1	384	164	3392	4301	0	0	4298	1211	770	3482
Layout 2	384	164	3392	4982	0	0	3354	1148	1089	3441
Layout 3	2804	1629	1379	2406	0	0	2906	3672	2109	531
Layout 4	228	1606	4264	3812	0	0	3239	1586	1140	1697
Inscript	402	1293	3585	4354	0	0	3524	1496	1513	1524

Table 3.3 Finger Usage statistics for different keyboards

Keyboard Layout	Overall score
Layout 1	30560
Layout 2	29324
Layout 3	30612
Layout 4	30360
Inscript	28078

Table 3.4 Overall evaluation scores for different keyboards

distribution of weight on Rows 2 and 3, keeping least weight on Row 4, which is highly desirable for a good keyboard, as mentioned in Section 3.4.

The overall score of the keyboards, based on the factors mentioned in Section 3.4 are as mentioned in Table 3.4

As it can be seen from the table, Layout-3 scored the best amongst all, followed by Layout-1 and Layout-4. The final evaluation considered factors like finger alternation and finger rolling too while scoring a layout, as mentioned in Section 3.4. Despite the statistical ground, Layout-4 did not score better than two of the other intuition based designs. Inscript had the least score of all the layouts. These experiments prove that Our layouts provide better alternatives to existing inscript keyboard layout.

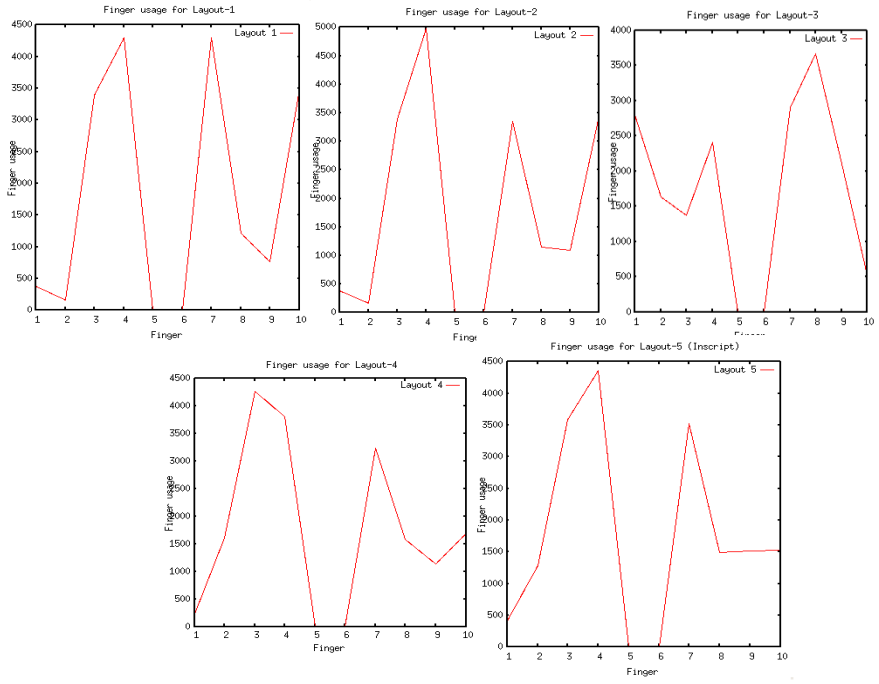

Figure 3.5 Finger usage for various keyboards

(Finger arrangement: 1 represents the little finger of left hand and 10, the little finger of right hand)

3.6 Summary

In this chapter, we have worked towards designing new keyboard layouts for Telugu. We have modelled our designs based on a set of design principles, which are formed based on the premise that the keyboards designed should be efficient as well as easy to learn. They should also be friendly for the first time users. We have designed three intuition based layouts and one layout based on statistics. The statistics obtained were obtained from the crawl data of a search engine and hence can be expected to represent the language in general. A new keyboard was designed by placing most commonly occurring alphabets on the most accessible positions and least occurring ones on the not so used positions. After the designs were formed, we have evaluated all the four of them along with *inscript*, which is the existing keyboard layout for Indian languages. The evaluations were performed based

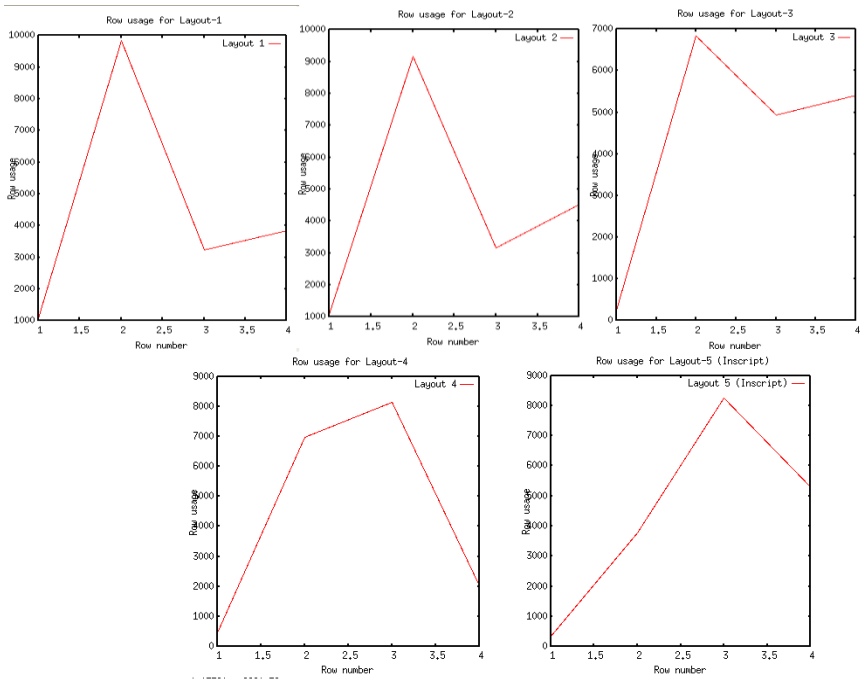

Figure 3.6 Row usage for various keyboards

(Row 1: Numbers; Row 2: QWERTY row; Row 3: Home row; Row 4: ZXCV row)

on previous evaluations performed for English by [34] and [63].The results are explained in Section 3.5. On analysis of results, it is clear that though none of the proposed keyboards perform perfectly according to all the evaluation measures, all of them displayed a better performance than Inscript, thereby providing alternatives to the current method. Hence, these experiments provided better keyboard layouts within the limitations of QWERTY structure, which are better than the existing options.

Chapter 4

Transliteration

Transliteration in this context refers to the process of entering text in Telugu using English script, following a mapping between Telugu and English alphabet. In this chapter, we explain different mapping schemes available for Indian languages and the evaluations performed on them, apart from comparing them with a new mapping that we have formed.

4.1 Motivation

Different mappings exist for Indian language transliteration though there is no standardized mapping as such. The way to form a mapping is dependent entirely on the intuition of the designer. Hence, despite the fact that various tools exist for Indian language typing, each one of them need some amount of adaptability from the user's side. A user accustomed to one tool might not be very comfortable with a new tool, owing to this difference in mapping between alphabets of the two languages. Amongst the several mappings available for Indian languages, some of the popular mappings are shown in Figures 4.1, 4.2 and 4.3.

అ-a	ఆ-A/aa	ఇ-i	ఈ-ee/I	ఉ-u	ఊ-U/oo	ఋ-R	ౠ-Ru
ఎ-e	ఏ-E	ఐ-ai	ఒ-o	ఓ-O	ఔ-au	అం - aM	అ: – a@h
క-ka	ఖ-kh	గ-ga	ఘ-gh	ఙ ~ma	చ - cha	ఛ- Cha	జ-ja
ఝ-jha	ఞ- ~na	ట-Ta	ఠ- Tha	డ- Da	ఢ- Dha	ణ- Na	త-ta
థ- tha	ద-da	ధ- dha	న-na	ప-pa	ఫ- pha	బ-ba	భ- bha
మ- ma	య- ya	ర-ra	ల-la	వ-va	శ-Sa	ష- Sha	స-sa
హ- ha	ళ-La	క్ష – ksha	ఱ = ~ra				

Figure 4.1 RTS - Rice Transliteration Scheme (Telugu Mapping)

4.2 Saral : Yet another mapping

There are enough mapping schemes between Roman and Indian languages, which in itself questions the need for a new mapping. However, attempts to eliminate the usage of SHIFT key altogether and an attempt to achieve better uniformity in a mapping scheme made us work towards a new mapping without using a SHIFT key. The new mapping, called *Saral* is shown in Figure 4.4 As it can be understood from Figure 4.4, *Saral* mapping uses ”.” to indicate an elongated vowel sound as well as a stressed sound. The usage of ”.” is done as an alternative to the usage of “shift” key. However, this will not make much of a difference in terms of average number of key strokes, since instead of pressing two keys at a time (shift + a given key), it now allows the user to press two keys back to back, not reducing the number of key strokes in any way. However, since this sticks to the rule mentioned above regarding the usage of “.”, it is expected to be easier to remember compared to other mappings.

అ-a	ఆ-A	ఇ-i	ఈ-I	ఉ-u	ఊ-U	ఋ-RRi	ౠ-RRI
ఎ-e	ఏ-E	ఐ-ai	ఒ-o	ఓ-O	ఔ-au	అం - aM	అ: – aH
క-ka	ఖ-kha	గ-ga	ఘ-gha	ఙ ~Na	చ - cha	ఛ-Cha	జ-ja
ఝ-jha	ఞ-~na	ట-Ta	ఠ-Tha	డ-Da	ఢ-Dha	ణ-Na	త-ta
థ-tha	ద-da	ధ-dha	న-na	ప-pa	ఫ-pha	బ-ba	భ-bha
మ-ma	య-ya	ర-ra	ల-la	వ-va	శ-sha	ష-Sha	స-sa
హ-ha	ళ-La	క్ష – kSh	ఱ = shr				

Figure 4.2 ITRANS mapping (Telugu)

4.3 Evaluation & Results

Evaluation of mapping schemes has not been performed so far in the available literature. Hence, there are no established evaluation metrics either. So, we have used three simple heuristics to compare different mappings. They are:

1. Average number of keystrokes taken to type a word
2. Ability to alternate between both the hands while typing
3. Ease of remembrance of a mapping scheme.

4.3.1 Average number of keystrokes

All the transliteration schemes under evaluation (RTS, Wx, ITrans, Saral) were taken and average number of keystrokes taken to type in a test data was calculated. The evaluation was done for Telugu language and test data was collected from a Telugu news daily corpus. The testing was done with around 100 words. A text is taken and typed using each of the

అ-a	ఆ-A	ఇ-i	ఈ-I	ఉ-u	ఊ-U	ఋ-q	ౠ-Q
ఎ-eV	ఏ-e	ఐ-E	ఒ-oV	ఓ-o	ఔ-O	అం - aM	అ – aH
క-ka	ఖ-Ka	గ-ga	ఘ-Ga	ఙ fa	చ - ca	ఛ-Ca	జ-ja
ఝ-Ja	ఞ-Fa	ట-ta	ఠ-Ta	డ-da	ఢ-Da	ణ-Na	త-wa
థ-Wa	ద-xa	ధ-Xa	న-na	ప-pa	ఫ-Pa	బ-ba	భ-Ba
మ-ma	య-ya	ర-ra	ల-la	వ-va	శ-Sa	ష-Ra	స-sa
హ-ha	ళ-lYa	క్ష – kRa	ఱ = rYa				

Figure 4.3 Wx Mapping (Telugu)

notations. The average number of keystrokes taken to type using that notation is logged. This experiment does not involve any user participation as the process is automated by writing reverse transliterators which convert the given Unicode Telugu text in to Roman script, following the given mapping and using this converted text calculate the number of keystrokes involved in typing it. The scheme with least average keystrokes per word is the best mapping according to this measure.

The results are as follows : (in Keystrokes per word)

- RTS: 13.32
- Wx: 13.23
- ITRANS: 13.35
- Saral: 13.34

అ-a	ఆ-aa	ఇ-i	ఈ-ee	ఉ-u	ఊ-oo	ఋ-r.	ౠ-r.u
ఎ-e	ఏ-ae	ఐ-ai	ఒ-o	ఓ-o.	ఔ-au	అం - am.	అ: — ah.
క-ka	ఖ-kha	గ-ga	ఘ-gha	ఙ — g.a	చ - cha	ఛ-ch.a	జ-ja
ఝ-jha	ఞ-j.a	ట-t.a	ఠ-th.a	డ-da	ఢ-dh.a	ణ-n.a	త-ta
థ-tha	ద-da	ధ-dha	న-na	ప-pa	ఫ-pha	బ-ba	భ-bha
మ-ma	య-ya	ర-ra	ల-la	వ-va	శ-sha	ష-sh.a	స-sa
హ-ha	ళ-l.a	క్ష — ksh	ఱ = rr.				

Figure 4.4 Saral Mapping (Telugu)

4.3.2 Hand alternation

The keys in the keyboard are classified in left and right handed keys and the ability of a scheme to alternate between both the hands is calculated. This gives the ability of the encoding to switch between two sides efficiently. Our intuition is that if a user is able to freely alternate between two hands while typing, the encoding scheme uses the two-handed typing ability of the user efficiently. This experiment also does not involve user participation as the process is automated in the way same as above. The text obtained in Roman form for each encoding is used to calculate the ability to alternate between both the hands. This is done by first classifying the keyboard keys in to right-handed and left-handed respectively. Whichever scheme achieves the least score in this evaluation is the one with best hand alternation.

The results of this evaluation are as follows:

- RTS: 19.79
- Wx:19.62
- ITRANS: 20.88
- Saral: 19.55

Mapping Scheme	Average Edit Distance per word
RTS	0.37
Wx	0.92
ITRANS	0.54
Saral	0.51

Table 4.1 Ease of remembrance experiment

4.3.3 Ease of Remembrance

This experiment was performed based on a similar experiment by Goonetilleke et.al. [17], which was performed to estimate the ease of remembrance of a particular romanized mapping compared to other for Sinhalese language. We have asked a group of 9 users to romanize the alphabets of Telugu language. Users were given a 2-column table. First column contains the Telugu alphabet . Second column is vacant and the user was asked to fill it the spelling of the Telugu alphabet according to their intuition. It is followed by the following procedure:

1. For a given encoding scheme, calculate the average edit distance between the user's Roman transliteration and the encoding scheme's roman transliteration.

2. The scheme with the lowest average edit distance is the easiest to remember.

We performed the experiment with three popular mapping schemes for Indian languages (RTS, Wx and ITRANS). The results are tabulated below:

Conclusions

1. The ranking was: RTS,Saral,ITrans and Wx

2. RTS had a better closeness to user intuition, despite the fact that it uses capital letters,which involves pressing a shift button frequently.

The conclusions of this experiment were later used in the design of prediction systems for Indian languages using edit distance as a characteristic feature. This is explained in detail in Chapter 5.1.

4.4 Summary

In this chapter, we have explained the presence of different transliteration schemes for Indian languages and lack of standardization among them. We have also attempted to propose a new transliteration scheme by trying to eliminate the usage of shift-key in typing Indian languages using English. We have also attempted to evaluate these transliteration schemes by using simple heuristics. We have performed three evaluation tests. The first test aimed at estimating the average number of keystrokes per word taken by a given transliteration scheme. It was shown from the experiment that all the 4 Transliteration schemes tested performed more or less in a similar manner in this test. The second test aimed at estimating the ability of the transliteration scheme to alternate between two hands while typing. It was proven that all the Transliteration schemes under test performed more or less in a similar manner in this test too. The third test aimed at estimating the ease of remembrance of a transliteration scheme. Rice Transliteration Scheme clearly outperformed other transliteration schemes in this test, which was performed using 9 users. Though no public evaluation was performed on the various transliteration schemes in vogue for Indian languages, these tests have proven that there is not much of a difference among them in terms of performance. It thus establishes that a rapport between user and any of these mapping schemes is totally dependent on the user's familiarity with it.

Chapter 5

Text Prediction

Text prediction refers to the act of estimating the user's needs and reducing his/her typing effort by offering suggestions based on what he has typed. We have classified prediction systems in to four categories:

- Wordlist based static systems
- Statistical prediction
- Fuzzy string matching based prediction systems
- Typing aid based prediction

We have studied the problem and designed prediction systems based on the first and third approaches. The first approach is more similar to the likes of Google Suggest, which was implemented on Webkhoj ([44]), an Indian language search engine. The fuzzy string matching based approach was designed keeping in view the Indian internet users. There hasn't been a fuzzy string matching based approach for text input in Telugu so far.

Most of the Indian language users on the internet are those who are familiar with typing using an English keyboard. Hence, instead of introducing them to a new keyboard designed for Indian languages, it is easier to let them type their language words using Roman script.

In this chapter, we deal with text input as a transliteration problem. We attempted to solve this problem by trying out different ways to exploit a large word list which is available to us from the crawl data of an Indian language search engine. Amongst the different methods tried out, an edit distance based approach worked most efficiently for the problem. We have performed these experiments on Telugu. But, the idea can be generalized to other Indian languages as well. This approach is not restricted to Indian languages, though. It can be used with other non-Roman script based languages, where a large word list exists.

Further, the technique will also help transliterating text offline. There is a huge amount of data in the form of Romanized Telugu or any other language, used in various sites on the internet. Lyrics sites and discussion boards are the best examples for this kind of a scenario. Our technique can be applied in converting this text to Unicode. We describe our design and subsequent experiments conducted to verify the efficiency of the system in this chapter.

5.1 Experiments in English to Telugu transliteration

We have considered text input as a transliteration problem in this work. Hence, we have worked on evolving approaches which consider text input from this point of view. Since we had access to a 1,600,000 strong word list for Telugu, collected from a crawl data index of an Indian language search engine, we have made attempts to exploit it. Further, lack of a parallel corpus for this kind of data came in the way of using any Machine Learning based techniques. Hence, we have come up with several approaches to text input, which involved the usage of this word list. The word list in Telugu was first stored in Roman script, by following a pre-defined mapping. This mapping was formed by performing an *Ease of remembrance experiment* on the existing mapping schemes for Indian languages, which is explained in Chapter 4.3.3.

Since RTS had a better ease of remembrance as proved by this experiment, we have used a near RTS notation to convert our word list to Roman notation. After converting the Telugu word list in to Roman script, we have utilized it in different approaches for forming a data input method. Though we have finally concluded that Levenshtein distance based approach works the best of all, we are presenting the different approaches we have tried in the process of forming an efficient text input method for Telugu. The experimented approaches are explained below.

5.1.1 Generating Combinations

In this method, the input word is split in to individual phonemes. For example: bharat is split as bha-ra-t. An input mapping table is created, which has two columns. The first column contains the phoneme and the second column contains possible alternatives to that phoneme. For example, the phoneme bha can have its variations as ba,baa,bha,bhaa బ,బా,భ,భా . The second column is filled using the data obtained by performing the "ease of remembrance" experiment, explained in detail in Chapter **??**. This approach is similar to that performed by [18], who also tried a word list based approach for building a Sinhalese language prediction system. Finally, using this mapping table, all possible combinations of words that can be formed from the input word are generated. Valid words from the generated combinations are returned as predictions. However, as the length of the word increases lakhs of combinations need to be generated, which slows down the whole process. Further, the method has to search through the word list to give only valid words as output, which again involves usage of a large word list. This makes search extremely slow owing to the size of the word list. Hence, an improvement over this idea was implemented, which involves converting the existing word list in to a mapping table.

5.1.2 Building a mapping table from the wordlist

In this approach, we have re-arranged the Romanized word list to form a key-value data structure. The Roman-mapping was case sensitive. Hence, we have formed a new data structure in which the key is a lower-cased word and its value had all the possible case-variated words to the key word, which existed in the dictionary. For example, against an entry kavali, the entries, in RTS mapping will be: kAvAlI;kAvAli;kAvali;kavAli;khavAlI (కావాలి;కావాలి;కావలి;కవాలి;ఖవాలి), which are the various words that exist in the word list, that are case-variated versions of the entered word. Though this approach was better than the previous one, it had one drawback. It works only if all the vowels and consonants entered were correctly spelt by the user, which is not always the case. For example, the word Burma (another name for the country -Myanmar) is actually pronounced barma and hence written so in Telugu. But, since this approach does not save those variations in its data structure, it fails to transliterate in this case. This obviously resulted in a less efficient system since this is a common problem in text input for Indian languages.

5.1.3 Fuzzy string matching based approaches

Since the above mentioned approaches have not proved to be efficient enough for the task, we have tried out some experiments with string similarity metrics and other fuzzy string matching approaches. First, we have experimented with Jaro-Winkler distance. It is a measure of similarity between two strings. It is a variation of Jaro distance metric, which is used for duplicate record detection. It is explained in greater detail in [4]. Our procedure is based on the assumption that the first letter entered is the correct letter. We calculate similarity between the words in the word list starting with that letter and the input word. All the words, which cross a certain threshold level are given as predictions for the input word. The approach scaled well and was fast enough. However, as mentioned by Winkler in [68], the Jaro-Winkler similarity works well for proper names. Its initial purpose was to get the spelling variations of first names and last names. But our dataset is a generalized data set,

which has both named entities as well as non-named entities, since this is the problem of a general purpose text input. Hence, this approach did not give a good efficiency to provide a better text input method.

Hence, we have tried to experiment with normalization methods like Soundex[1] Originally designed to get the spelling variations of names, Soundex achieves its purpose by grouping the letters in to respective classes. The Soundex algorithm is explained to some extent in [61]. In essence, it assigns a code to the given word based on its grouping. Similar names will possess the same Soundex code to a large extent. We have customized the Soundex grouping of letters as per the needs of a transliteration approach. We have altered the grouping of classes to some extent and extended the ”Alphabet+3 digit code” of Soundex to ”Alphabet+N digit code” where N is dependent on the length of the input word, since there will be so many words that have a similar 3 digit code as the input word. However, this did not improve the efficiency much.

Edit distances have been used for a variety of matching tasks. They have been used widely for the purpose of detecting spelling variations in named entities. Freeman et.al. [61] used an extension of Levenshtein edit distance algorithm for Cross-linguistic name mapping task in English to Arabic transliteration. Cohen et.al. [64] have performed a comparison of different string distance metrics for name matching tasks. They have also implemented a toolkit of string matching methods for general purpose application. Fuzzy string matching methods for Named entity recognition and other related tasks have been explored in greater detail by [45, 40, 43, 52] in the context of Indian languages, especially. However, though all these approaches have used edit-distance and other string matching based approaches widely, this work differs from them in the application scenario. While all of them have used those methods for name-matching tasks in mono-lingual as well as cross-lingual retrieval, we used those methods to develop an input method for Telugu language.

[1] http://en.wikipedia.org/Soundex

Hence, we have tried using edit-distance as a method to devise an efficient text input method for Indian languages. We have used Levenshtein distance [2] for this purpose because it calculates the cost involved in converting one word in to another by considering the number of insertions, deletions and substitutions required for the source word to be converted to target word. Typically, insertions/deletions/substitutions are the three common factors involved in typing one language using the script of the other language. This approach has worked extra-ordinarily well and was proved to be very efficient. Our approach is explained below.

5.2 Levenshtein based English to Telugu Transliteration

Levenshtein distance between two strings is defined as the minimum number of operations (which comprise of insertion, deletion or substitution of a single character) required to transform a string to another. In the scenario of transliteration based text input, the intended string differs from entered string typically in one or many of the above mentioned operations. Hence, Levenshtein distance is a suitable metric to experiment with, in the development of a text input method. The following are the steps involved in our approach:

Step 1-Pre-processing the input word: The preprocessing step involves applying a few heuristics on the input word to make it as close as possible to the internal mapping. For example, say a user enters the input word as ”raamaayanam” (రామాయణం), which is one of the Indian epics. The word list actually saves such a word as ”rAmAyaNaM” according to its mapping. Hence, one of the rules applied in the preprocessing step is to replace ”aa” by ”a”. Since we compare the edit distance between two words only after converting the words in to lower case, the two words match. This is a simple example. Such rules are written to do some pre-processing on the input word to convert it in to the required form.

Step 2-Prediction: As mentioned before, based on the intuition that the first letter entered by the user is the correct letter, we search through only the words starting with that alphabet

[2] http://www.let.rug.nl/ kleiweg/lev/levenshtein.html

in the word list. The words which satisfy the requirements are returned as suggestions. In a couple of experiments, we have considered that the last character should also match. However, this is not a mandatory requirement. The requirements also include calculation of three Levenshtein distances. They are:

1. Levenshtein distance between the two words (i)
2. Levenshtein distance between the consonant sets of the two words (j)
3. Levenshtein distance between the vowel sets of the two words (k)

Here, the terms consonant set and vowel set refers to the concatenation of all consonants and all vowels in the word respectively.

Step 3-Conversion to Unicode: Finally, we have to convert the Romanized predictions back to Unicode notation. This is done by reversing the process followed to convert the word list in to a Roman notation, which was performed in the initial stage.

We have conducted several experiments by varying the Levenshtein distance thresholds of the three distances mentioned above. Our experiments and the results are explained in the following section.

5.3 Experiments and Results

We have performed some experiments by varying the threshold levels of the three Levenshtein distances we have considered in designing the system. We have tested this approach on three types of datasets, to estimate the efficiency of this system in offering good predictions. The datasets are explained below.

1. A general dataset of 500 words, which contains Telugu text typed in general. It includes Telugu words typed in Roman script, named entities like place names and country names and English words adapted in to Telugu. This data was collected from Telugu websites including Telugu Wikipedia, news sites and blog pages. Since the

data was collected from a free-text rather than a collection of words, the text in this dataset is the representative of a real word scenario .

2. A dataset of country/place names, which had around 200 words. This was collected from Telugu Wikipedia pages on the respective places.

3. A dataset of person names. This was again collected from Wikipedia pages on famous people. It had 150 words. Some Indian names were also included in the dataset, which mostly consisted of sportsmen, politicians and writers.

There might have been some overlap of the last two datasets with the first dataset since it is a general dataset which encompassed the other two categories. The results are summarized in the following tables. The first column indicates the experiment number, the second, third and fourth columns indicate the different Levenshtein distances explained in the previous section and the last column shows the efficiency. Efficiency is calculated as the number of words for which the system gave correct predictions for a given input word. Results using Dataset-1 is are shown in Table 5.1. Results using Dataset-2 and Dataset-3 are shown in Tables 5.2 and 5.3 respectively.

General Observations:

1. The system has not been very efficient with an exclusively named entity dataset compared to the generalized dataset.

2. The person list had the largest accuracy for a particular experiment, more than the generalized dataset.

3. The accuracy with the general dataset has been considerably good.

The variation of accuracy of the system with word Levenshtein distance threshold has been plotted in the graph shown in Figure 5.1. As it can be seen from the graph, the system performed very well with the generalized test data, which proved that this can be an efficient solution to text input for Telugu. This approach is language independent and hence can be

Expt No	Word Levenshtein distance (i)	Consonant Levenshtein distance (j)	Vowel Levenshtein distance (k)	Accuracy (%)
1	5	2	1	90.6%
2	4	2	1	90.6%
3	3	2	2	91.0%
4	4	2	2	91.8%
5	6	2	2	92.0%
6*	4	2	2	92.4%
7	4	2	-	92.6%
8	6	2	-	93.2%

Table 5.1 Transliteration accuracy with Dataset-1

(*: Difference between experiment 4 and experiment 6 lies in the fact that while experiment 4 considers the last character matching condition, experiment 6 does not.)

applied to any other Indian language, owing to the similarity between Indian languages. The success of the system with the Levenshtein distance approach can be attributed the relationship between the nature of Levenshtein distance and the process of typing Telugu in English, as mentioned in the previous section.

The failure of this system with Named Entities can be owing to the fact that the way they are written is different from the way they are spelt. But, our approach is based on how the words are spelt since the dictionary is converted to a Roman mapping based on the same factor. But, the system worked better with Indian named entities i.e., places, person names which are of Indian origin. It can be attributed to the same reason, since Indian words are spelt in English in the same way as they are pronounced, to a large extent. It is natural that the system will not perform very efficiently with named entities compared to generalized test data, since our purpose is not named entity transliteration but a general purpose text in-

Expt No	Word Levenshtein distance (i)	Consonant Levenshtein distance (j)	Vowel Levenshtein distance (k)	Accuracy (%)
1	5	2	1	75.0%
2	4	2	1	75.29%
3	3	2	2	79.25%
4	4	2	2	79.25%
5	6	2	2	79.25%
6*	4	2	2	83.27%
7	4	2	-	79.24%
8	6	2	-	84.69%

Table 5.2 Transliteration accuracy with Dataset-2

(*: Difference between experiment 4 and experiment 6 lies in the fact that while experiment 4 considers the last character matching condition, experiment 6 does not.)

put system. We did not work on specifically transliterating named entities in this approach. Hence, in this perspective, the results on the named entity datasets are encouraging enough. However, the results prove that using Levenshtein distance is a workable approach to build a text input method for Indian languages. Possible alternatives to improve efficiency with named entities can be trying with a different metric, using a combination of metrics or using some heuristics in the existing method.

5.4 Summary

In this chapter, we have proposed a simple, yet efficient technique for text input in Telugu, which can be easily extended to other non-Roman script based languages. Since it does not involve any learning on the part of the system, it does not put heavy weight on the system.

Expt No	Word Levenshtein distance (i)	Consonant Levenshtein distance (j)	Vowel Levenshtein distance (k)	Accuracy (%)
1	5	2	1	88.11%
2	4	2	1	88.11%
3	3	2	2	89.10%
4	4	2	2	89.5%
5	6	2	2	91.1%
6*	4	2	2	96.0%
7	4	2	-	91.10%
8	6	2	-	96.09%

Table 5.3 Transliteration accuracy with Dataset-3

(*: Difference between experiment 4 and experiment 6 lies in the fact that while experiment 4 considers the last character matching condition, experiment 6 does not.)

In this approach, we have exploited the availability of a large word list for the language to design a new text input method. Having tried out different approaches in using this word list, we have finally concluded that a Levenshtein distance based approach is most efficient of all. This is because of the relation between Levenshtein distance and the nature of typing an Indian language through English. The approach offers only valid words as suggestions and it gives the results instantaneously.

We have to work on further improving the approach so as to make it work better with named entities too, since it will also help in creating gazetteers of named entities in Telugu. We have to test the system on larger datasets before making a full fledged application which uses this system. We plan to explore other distance metrics and/or using a combination of distance metrics to achieve a good working system. The system offers too many suggestions for words of smaller length. We have to look at alternative methods to deal with this

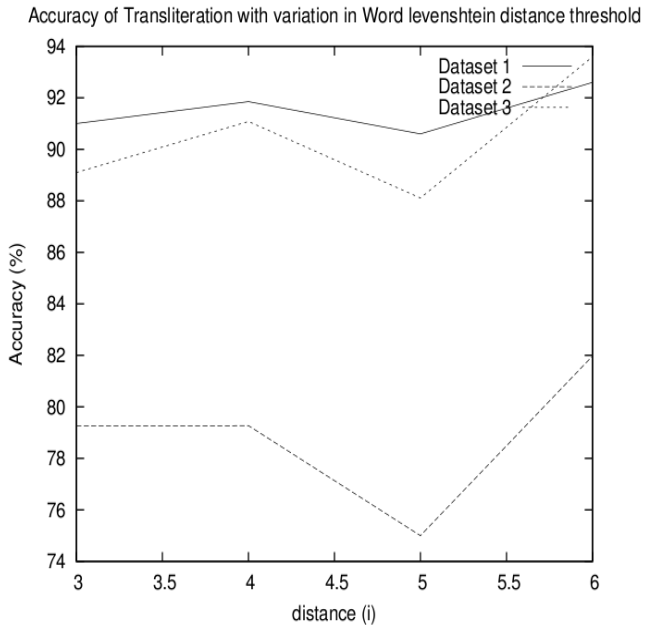

Figure 5.1 Variation of Transliteration accuracy with Levenshtein distance

problem. A proper ranking function needs to be designed to rank the results in an optimal order.

Chapter 6

Soft Keyboards

The term *soft keyboard* refers to any kind of keyboard that serves as an input method in place of an actual hardware keyboard. They are also referred to as *virtual keyboards* [1] or *on screen keyboards*. A soft keyboard can be understood as a software with an image map on the screen, which enables the user to enter data. Kolsch et.al. [32] define soft keyboard as a typing device which does not have a physical manifestation. A "button" on a soft keyboard is not exactly a button but is programmed to behave so. Pointing devices like mouse or stylus are used to enter data using soft keyboards. Though a soft keyboard can mean anything like a touch-screen type pad, track ball control[2], eye-tracking device or a projection keyboard [3], we use it in the context of usage on desktop and laptop computers. The initial purpose in introducing a soft keyboard was to facilitate typing for people with disabilities or special needs. However, it is typically used in hand held devices as a means of data input, instead of a physical keyboard.Though there are many alternative ways of data input for such needs, like Graffiti [4] and Dasher[5], soft keyboard is the simplest way to enter data and its accuracy depends only on the user entering the right key. Further, a

[1] http://en.wikipedia.org/wiki/Virtual_keyboard

[2] http://en.wikipedia.org/wiki/Trackball

[3] http://en.wikipedia.org/wiki/Projection_Keyboard

[4] http://en.wikipedia.org/wiki/Graffiti_%28Palm_OS%29

[5] http://en.wikipedia.org/wiki/Dasher

soft keyboard does not have the problem of keyboard logging which makes it a convenient method of data input in secure systems like banking. They can also be used as typing aids for people who are not accustomed to typing. They can also be used in mobile devices for text messaging and other input needs.

In case of Indian languages, the need for soft keyboards arises majorly as an alternative data input method, since there is a lack of standardized input method, which is convenient to use for novice users in Indian languages. For example, consider the case of an Indian language search engine. The searchers' input needs are short and are restricted to a few word query. Further, the searchers need not be familiar with Indian language typing tools. In this kind of scenario, the presence of a soft keyboard can be of immense help since it requires absolutely no training for a novice user to enter text using it. Though the speed of typing may not be comparable with a trained usage on a normal keyboard, for short typing needs, speed has a lesser impact on typing. In this context, design of soft keyboards for Indian languages obtains significance.

6.1 Design Principles

Chapter 2.4 has surveyed some of the relevant literature on soft keyboard design for English as well as other languages, including Indian languages. However, incase of Indian languages, there has been no authentic study yet on the design of soft keyboards. All the currently existing tools are adhoc approaches without a structured study and evaluation. Hence, in our work, we study the problem of design of soft keyboards, formulate some design rationale based on the previous work done for English and apply the same to design new soft keyboard layouts.

Mackenzie et.al and Zhai et.al. [37, 70] have worked widely on the design of soft keyboards for English. We have formulated our design principles based on their rationale. Our design principles are:

Ease of remembrance of the layout: The layout should be easy to remember so that even

novice users will not face any problem in using it. The aim was to reduce the visual scan time of the user without compromising on the efficiency to tap using the keyboard.

Lessening mouse movement on the screen: The alphabets are placed in such a way that the keyboard can be used with minimum mouse movement possible.

More than one space button: Space is the most commonly used button since it is very likely to occur after every word. Hence, we have placed space buttons at more than one location on the keyboard so that the users can prefer the nearest space key.

Placing phonetic variant buttons in the middle of the keyboard: The phonetic variant buttons are the most likely ones to be used for every alternate key press. Hence, they should be placed in the middle of the keyboard since this navigation reduces the distance travelled by the mouse.

Alphabetical ordering: As it can be seen from the previous works, alphabetically ordered virtual keyboards performed better than the non-alphabetically ordered ones. Hence, we tried to stick to alphabetical ordering as much as possible.

As mentioned before, Telugu soft keyboard requires a minimum of (54+16) 70 keys. Further, addition of numbers, punctuation markers and other keys makes it a longer keyboard compared to its English counterpart. This increased number of buttons is valid for all Indian languages with little difference. We tried to implement the keyboards which include all these buttons without compromising on the design principles. On testing the possible designs, we have come up with three most natural designs possible, which stick to the design principles as closely as possible.

6.2 Design

The following are the soft keyboards that we have designed based on the above mentioned design principles.

Figure 6.1 Layout-1

6.2.1 Layout-1

Figure 6.1 shows the first layout. It has 90 (9*10) buttons on the whole. The vowels are arranged on the top side. Phonetic variant buttons are placed in the middle of the layout, color coded. They are placed in the middle since they are most likely to be chosen on every alternate click. All the consonants are arranged around this phonetic variant part. The space key has been repeated twice, on top left side and the bottom right side. Numbers and some punctuation markers are added. This keyboard conforms to the design principles mentioned previously, in all aspects except alphabetical ordering. However, color coding improves the ease of remembrance of the layout and makes it more visually appealing.

6.2.2 Layout-2

Figure 6.2 shows the second layout. This is arranged in the order of Telugu alphabet. The first row is occupied by numbers and the rest of the rows by alphabets and punctuation markers. It has vowels and phonetic variant buttons arranged on the right side and the consonant buttons on the left side. This is designed for a better ease of remembrance. The

Figure 6.2 Layout-2

Figure 6.3 Layout-3

phonetic variant buttons are again arranged towards the middle of the keyboard as much as possible, for an easy navigation.

6.2.3 Layout-3

Figure 6.3 shows the final layout of our design. This is also arranged in the natural order of the alphabet. However, the difference between Layouts 2 & 3 is that while Layout 2 has vowels and phonetic variants on the right and consonants on the left, Layout 3 has vowels

Figure 6.4 Google's soft keyboard for Telugu

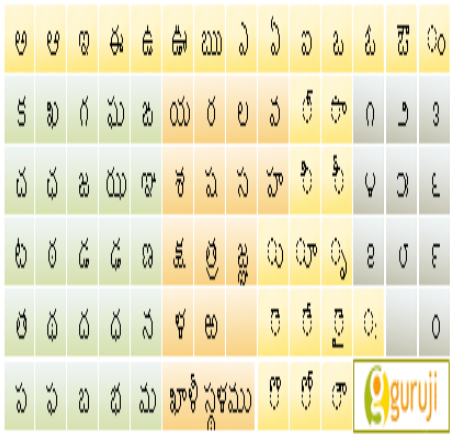

Figure 6.5 Guruji.com's soft keyboard for Telugu

and phonetic variants on the left and consonants on the right. In that way, its a reverse implementation of Layout 2. It has conformed to the design rationale in terms ease of remembrance and lesser navigation time.

6.3 Evaluation

We have evaluated the above mentioned three soft keyboards with three pre-existing soft keyboards. They are shown in Figure 6.4, Figure 6.5 and Figure 6.6 respectively

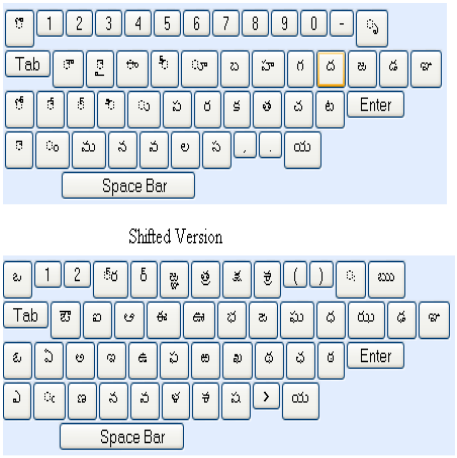

Figure 6.6 The Inscript soft keyboard

6.3.1 Novice User experience

In the evaluation of input devices, learning effects are considered a confounding factor ([38]). There are two ways to deal with it - one way is to have a single interaction which gives an overview of the efficiency of the system and the other is to subject the user to a series of sessions and obtain the learning rate. However, human performance inevitably improves with practice in both simple and complex tasks as well ([30]). As mentioned in [70], initial performance determines the success and future of any new interaction technique. They have also concluded that the users are in general reluctant to go through the ordeal of learning, before appreciating a new interface. Mackenzie et.al. [38] also mentioned that despite the fact that the predictions for novice users are less striking than those of experts, it is this novice experience factor that determines the acceptance of a new technology.

Hence, we have evaluated our soft keyboards along with some of the existing ones by conducting the Novice user experience experiment as proposed by ([26]). The goal of this experiment is to understand a first-time user's performance with the system. Our use of the term "novice behaviour" simply implies that the user is working with a soft keyboard with an unfamiliar letter arrangement ([26]). We have evaluated 6 keyboards in total 3

keyboards designed by us and the other three are Google (Figure 6.4),Guruji.com (Figure 6.5) and Inscript Figure (6.6) Mackenzie et.al ([37]) refers to this kind of experiment as quick test owing to the simplicity of technique used.

6.3.2 Learning curve

Learning curve refers to the process of conducting a longitudinal study on an input device, to estimate its usability in a long run with users. This can be used to get an understanding of the ability of the users to learn efficient usage of a particular input device. Hence, we have conducted this experiment on our users in a multi-session interaction to evaluate the soft keyboards and draw conclusions on their usability.

6.4 Experiments & Results

6.4.1 Novice user experience

Novice User Experience refers to the process of evaluating a given soft keyboard in terms of the comfort level of its first time users. The users of this experiments, though are regular users of normal QWERTY keyboard, are first time users for soft keyboards in general and Indian language soft keyboards in particular. The experiment's task is that the users will be asked to type a test passage on all the 6 soft keyboards under test. The time taken for the user to type the given passage is calculated which is in turn used to calculate the WordsPerMinute(WPM) -typing speed of a given keyboard.

Users

The experiment has been conducted with 15 users. All the users are native speakers of Telugu language, who are regular users of Computers. There are 8 Male and 7 Female users. All users are new to the usage of soft keyboards in general and Telugu soft keyboards in particular.

The Task

There are six keyboards considered for this evaluation. The task is described as follows: each user was given the test passage and was asked to type it using each of these keyboards. The test passage was the same for all the users and for all the keyboards for an unbiased evaluation. However, in this case, there is a possibility of the user learning the passage and typing it at a quicker rate in the last keyboards that he encounters than the first ones. Hence, the keyboards were randomly ordered for different users so that a user's learning of test passage will not affect the overall mean speed of a particular keyboard.

The participants were given instructions about their task and the goal of the experiment. The time taken to type the test passage on each of the keyboards was noted in seconds and was used to calculate the typing speed of the keyboard from a novice user perspective. No practice sessions were given and the text was entered only once-per user, per keyboard. The typing speed is calculated as per the method proposed by Mackenzie et.al.([38]), as follows:

Entry speed in Words Per Minute (WPM) = (No of letters in the test passage/entry speed in seconds)(60/Average number of letters per word)*

The test passage had 75 characters and was picked randomly from the news corpus of a Telugu news daily. It took around half-an-hour of an individual user's time to perform this test, on an average.

Results

The results of this test are shown in Table 1. The result table has three columns. First column shows the Keyboard layout, second column shows the mean entry speed in words per minute and the third column indicates the standard deviation.

Keyboard Layout	Mean entry speed in WPM	Standard deviation in WPM
Layout 1	3.27	1.05
Layout 2	3.91	0.55
Layout 3	3.596	1.17
Google keyboard	3.79	0.59
Guruji Keyboard	3.64	1.00
Inscript	2.71	0.97
overall mean	3.49	

Table 6.1 Novice user experience experiment: entry speeds for different keyboards (in words per minute)

Observations

The mean entry speed across all users and keyboards was 3.49 wpm. As it can be seen from Table 1, Layout-2 has the highest average Words Per Minute, closely followed by Google's keyboard. Though the difference appears lesser, it gains significance at a higher level. The relatively low standard deviations suggests that the mean scores for each of the keyboard layouts were consistent across the subjects.

The Inscript keyboard, which is known as the fastest typing layout on the *hard-keyboards* had the least score as an soft keyboard. This again confirmed our design rationale that an on screen keyboard need not stick to the QWERTY layout or its equivalent. Layout-1 was assumed to score better since it was designed in a visually appealing manner, with color codes for easy identification. Layout-3 was implemented as a reverse version of Layout-2. In essence, one should move from right to left in using Layout-3. Hence, it was expected to perform low. But, it performed better than Layout-1. Though, as mentioned by Norman et.al.([2]), for a *hard keyboard*, alphabetical ordering might not matter much, more alphabetically ordered layouts performed better, in the case of soft keyboards, as mentioned by Zhai et.al. ([70]). Perhaps, the failure of Layout-1, despite its visual appeal, can be ex-

plained considering ([70])'s conclusion that alphabetically ordered soft keyboards perform better.

The novice user experience experiment was conducted by Mackenzie et.al. ([37]) on English soft keyboards.. The speeds were in the range of 7-8 words per minute on an average. The low text entry rates on these keyboards compared to their English counterparts can be because of the increased alphabets in Telugu. Further,several letter combinations are possible in Telugu, which makes the typing task even more time-consuming.

6.4.2 Learning Curve

The Learning curve experiment aims to estimate the learning rate for a particular keyboard layout. Mackenzie and Zhang [37] have conducted learning curve experiments on their OPTI soft keyboard and QWERTY keyboard and analyzed their learning rates. We have used their experimental methodology to perform this experiment on Telugu soft keyboards.

Users

Since estimation of learning curve is a longitudinal study, usually, fewer participants are considered and tested for longer periods of time. We have used five participants in this experiment. All the users were native speakers of Telugu. They were all computer science students, three male and two female. All of them are new to usage of soft keyboards in general and Telugu soft keyboard usage in particular, at the start of the experiment. They are all familiar with using Telugu on computers, though not regular users. We have tested them for 5 sessions, on each keyboard, which meant, each user had 30 sub-sessions of keyboard typing (5 sessions, 6 keyboards). All were informed about the time commitment required for participation in this experiment.

The Task

The experiment was a 5*6 within subjects factorial design. The two factors are:

- Keyboard layout (Totally 6 in number)
- Number of Sessions (Totally 5 in number)

Each session lasted for about 40 minutes, with each keyboard taking around 6-8 minutes of the user-time. The order of the layouts per user, per session was alloted in a random manner so that the sessions will be un-biased. The test passages were small and had around 20 words on an average. The passages were chosen at random from a Telugu news daily corpus. The phrases chosen represented the language and were relatively simple and easy to remember. The sessions were scheduled in a gap of not more than three to four days each. This was to make sure that the test users use the system regularly. They were also instructed not to use any of these keyboards for any other purposes in between the sessions since we wanted to calculate the learning rate in an unbiased manner. Each participant was given clear instructions on the goals of the experiment and their task. They were asked to aim for both text entry speed and accuracy.

Results

The results of the learning curve experiment are tabulated below in Table 6.2. While the first three keyboards were designed by us, the last three keyboards were already existing keyboards, evaluated in this experiment.

The results have been plotted on a graph, with X-axis indicating the sessions and the Y-axis indicating the text entry speed. This can be seen in Figure 6.7

Observations

The following observations were noted from this experimental result.

- All the keyboards showed increase in text entry speeds as the sessions progressed.
- All the keyboards showed a fall after the fourth session. Hence, fourth session had the highest text entry speed.

Keyboard Number	Session 1	Session 2	Session 3	Session 4	Session 5
Layout 1	3.294	3.552	4.978	5.156	5.126
Layout 2	3.328	3.988	5.368	5.946	5.318
Layout 3	3.986	4.07	4.82	5.012	4.86
Inscript	2.404	2.94	4.562	4.236	4.14
Google keyboard	3.53	3.894	4.97	5.558	5.178
Guruji Keyboard	3.526	3.894	4.396	4.814	4.604

Table 6.2 Learning curve experiment - Session wise entry speeds for different keyboards

- Keyboard layout-2 was proved to be a better performer in terms of text entry rate. It achieved both the maximum text entry speed as well as a better learning rate.

From this experiment, it can be concluded that Layout-2 has the highest learning rate. It was already mentioned that Layout-2 had a better text entry rate over other keyboards with novice users too.Hence, it can be concluded to be the optimal soft keyboard for Indian languages.

6.5 Statistical Significance

6.5.1 ANOVA

Analysis of Variance or ANOVA([15]) is a statistical procedure used to infer about the differences in means of two or more groups of samples. ANOVA infact tests the null hypothesis that the groups represent random samples from populations with the same means. ANOVA was developed by R.A.Fisher in 1920s and is known as Fisher's ANOVA since it uses his F-distribution in the process of testing for statistical significance. ANOVA is performed by calculating the total variation of the data samples, variation within class and variation between different classes and using them to determine the statistical significance

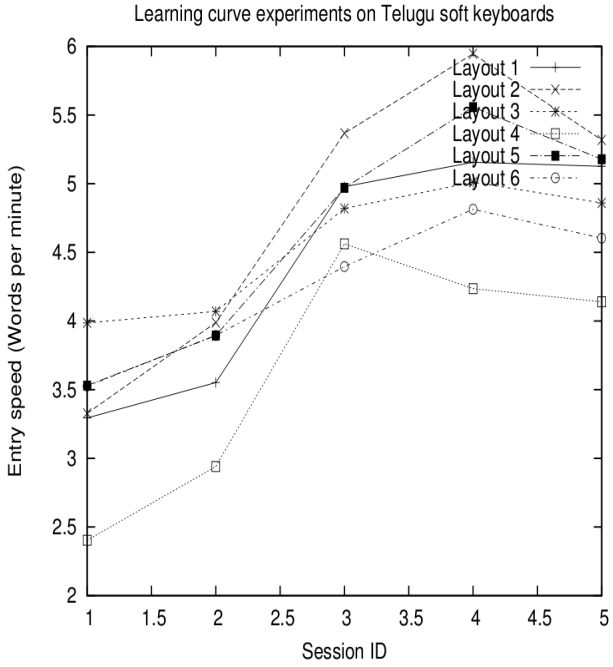

Figure 6.7 Learning Curve

of a sample by performing some test like F-Test over it. An ANOVA test ends in computation of a F-ratio, whose numerator reflects the differences among the group means and the denominator reflecting the differences within the group. If the numerator is sufficiently larger than the denominator, then the null hypothesis is rejected and the results are termed to be statistically significant. ANOVA is of two kinds, depending on the number of independent variables in the data. If there is only one independent variable, then One-way ANOVA is used to verify the statistical significance. Else, Two-way ANOVA is used.

6.5.2 One-way ANOVA

One-way ANOVA is the simplest case of ANOVA, which is used to analyze an experiment in which there is only one independent variable.It is used to test for significant differences between class means, which is done by analyzing the variances. In the present context, the keyboard layout is an Independent variable. Hence, one-way ANOVA is used to understand

the impact of keyboard layout on the typing speed of the users. This is performed by first estimating the total variation, within class variation and across group variation from the obtained experimental data. Later, tests like F-Test are performed using these estimations and levels of significance, to determine the statistical significance of an experiment.

6.5.3 Two-Way ANOVA

Two-Way ANOVA, also known as factor ANOVA is used when there is only one measurement variable but two independent nominal variables. In the present context, the measurement variable is the text entry speed. The two independent variables are: the keyboard layout and session id. A two way ANOVA can be done with or without replication. With replication indicates that there is more than one observation for each combination of the nominal variables. Without replication indicates that there is only one observation for each combination of the nominal variables. It is one of the commonly used test by biologists to verify statistical significance of their experimental data. It is useful in demonstrating how the independent variables interact and the combined effect that they have on the dependent variable.It is conducted in a similar way as One-Way ANOVA, followed by F-Tests to determine the statistical significance of the experiment.

6.5.4 ANOVA and F-Test on the experimental data

We have performed One-Way ANOVA test over the data obtained from Novice user experience experiment. This experiment has only one independent variable - Keyboard layout. Hence, One-way ANOVA has been performed, followed by F-test. It has been found that there is a significant effect of keyboard layout ($F(5, 78) = 4.71$, $p < 0.01$) on the text entry rate.

To verify the statistical significance of our Learning curve experiment, we have conducted the Analysis of Variance test over the data. In this experiment, the session and the keyboard are two independent variables. Hence, we performed two-way ANOVA, followed

by the F-test. It has been found that there is a significant effect of session [$F(4, 20) = 47.91$, $p < 0.01$] as well as the keyboard layout ($F(5, 20) = 10.17$, $p < 0.01$) on the text entry rate.

6.6 Summary

In this chapter, we have explained the design, implementation and evaluation of soft keyboards for Indian languages. We have surveyed the relevant literature for English language soft keyboards and formed a set of design rationale for Indian language soft keyboards applying the previous work to Indian language context. There have been some implementations of Indian language soft keyboards, though there has not been a proper study of the problem and evaluation of solutions. In this chapter, we have explained three soft keyboards we have designed and compared them with three exisiting soft keyboard layouts. In this process, we have also proved experimentally for Indian languages, what Zhai et.al have proved for English that alphabetically arranged virtual keyboards perform better than non-alphabetically ordered ones. We have also proved our design rationale that the layout for a soft keyboard need not stick to its normal counterpart. We have evaluated the keyboards by conducting two experiments involving the users.

1. 1. Novice user experience

2. 2. Learning curve experiment.

While the first experiment aims at understanding the ease of use of a soft keyboard for novice users, the second experiment aims at estimating the user-friendliness of a keyboard layout in terms of long term usage. The first experiment estimates the immediete usability of the tool, which in turn decides the acceptance of a new technology by users. We conducted this experiment with 15 users, who are all new to using soft keyboards in general and Indian language soft keyboards in particular. Our experiment has shown that one of our layouts (Layout-2) performed better compared to all other layouts. We have tried colour coding on one of our layouts, sacrificing the alphabetical layout condition, estimating that

visual appeal might increase the ease of remembrance of a keyboard layout. However, the layout did not perform any better with novice users, which proved once again that alphabetical ordering is a must for a soft keyboard. The results of this experiment are shown in Table 6.1

The second experiment is to estimate the learning curve of a layout. This is a longitudinal study, in which less number of users are tested for a longer duration. We have chosen five users and tested them for five sessions. Using the data obtained from these experiments, we have plotted the learning curve for each of the keyboard layouts. From the learning curve, it can be understood that not only was one of our layouts (Layout-2) the best performer in terms of speed, it was also the best performer in terms of the learning rate. The same keyboard performed well in the previous experiment too. Hence, it can be concluded that this layout is the optimized layout for Indian language soft keyboards. Though the design has been tested only for Telugu, owing to the similarity among Indian language scripts, the same designs can be applied to other Indian languages as well. A graph presenting the results is seen in Figure 6.7

We have performed statistical significance tests on the experimental data. Analysis of variance and F-Test were conducted on the data obtained from both the experiment. While One-way ANOVA was used for the first experiment since it has only one independent variable (keyboard layout), Two-way ANOVA was used in the second experiment since it had two independent variables (Keyboard layout and sessions). Our results were found to be statistically significant in both the cases. Soft keyboards can be used as an alternative data input method for Indian language technologies, in certain kind of application scenarios like a search engine user interface, where the data entry needs of end user are short. They may not achieve a speed comparable to normal keyboards. But, they require very less learning and absolutely no prior experience in usage. Hence, they can also be easily used as language tutor tools too.

Chapter 7

Conclusions and Future work

In this thesis,we have studied the problem of text input for Indian languages, taking Telugu as a test case. We have also classified, designed and evaluated various text input methods for Telugu. Owing to the similarity among Indian languages, the ideas can be applied to other Indian languages as well. Section 7.1 explains the contributions of the thesis and section 7.2 mentions possible directions for future work.

7.1 Contributions

In this thesis, we have proposed and evaluated different input methods for use in the context of Indian language computing. We have performed a detailed study of the available tools for data input both in Indian as well as other languages and formed our design rationale based on the previous work in English and other non-Indian languages. We have classified input methods for computer users in to four categories:

- Keyboard layouts
- Transliterators
- Prediction systems

- Soft keyboards

We have classified our study accordingly and continued with the same classification throughout our work. Within each category, we have proposed new designs and evaluated them by comparing them with existing designs. The evaluation techniques that we have applied are based on simple heuristics and have been used in this kind of applications in different works, cited appropriately throughout the thesis.

We have proposed new keyboad layouts for Telugu and evaluated them by comparing them with the existing keyboard layouts for the same. Our evaluations, which were explained in Chapter 3 have proved that our layouts performed better over the existing layouts. Our designs aimed at keeping the ease of use intact but improving on the ease of remembrance at the same time. In this work, we have successfully provided an efficient alternative to the existing keyboard layout for Indian languages, called *Inscript*. Our results and their comparision with Inscript in terms of their performance are shown in Chapter 3.4. For the users who are not used to a QWERTY keyboard, the proposed keyboard layouts can be directly useful to enable easy typing in Telugu. For others, these methods provide a direct input in Input in their language, instead of a transliteration based input.

We have studied the problem of lack of standardization of mapping between Indian language scripts and roman script during transliteration. In search of an optimized mapping, we conducted some simple experiments on different mappings available, apart from proposing a new mapping. These experiments have proven that there is not much of a difference in performance among the different mappings, though RTS(Rice Transliteration Scheme) had a considerably better ease of remembrance amongst the users, compared to other mapping schemes considered for this evaluation.

We have worked on the problem of transliteration based text prediction as a data input method. We observed that simple string matching based prediction can give good results

with a very light weight on system resources. We have conducted different experiments on prediction with Levenshtein distance, Soundex algorithm and Jaro-Winkler distance. While Levenshtein distance based approach performed very efficiently with the test data both independently and as a combination with Soundex algorithm, it was observed that Jaro-Winkler distance is not of great use in this scenario. With these experiments, we have successfully presented an alternative text input method for users who are comfortable typing in English, by providing them a transliteration based input system.

We have worked on designing and evaluating soft keyboards for Telugu, which can be used for other Indian languages as well. The initial motivation in working towards designing soft keyboards was an attempt to come up with an alternative text input method which does not require any learning/training on the part of the user. This was done keeping in mind the scenarios in which data input needs of a user are short, like giving a query to an Indian language search engine. We have designed three soft keyboards and evaluated them along with three existing soft keyboards. We have performed statistical significance tests on the results obtained and our results were proven to be statistically significant. One of our soft keyboards performed the best in the six designs tested. The results have been explained in greater detail in Chapter 6.4.

To conclude, in this thesis, we have studied and analyzed the problem of data input for Indian languages. We then classified different methods for data input, formulated some design rationale and designed new systems. We have also evaluated them and compared their performance with the existing systems. We have successfully proposed alternatives to existing data input methods in Indian languages.

7.2 Future Work

In this work, all the designs and evaluations performed are restricted to Telugu language only. Development of efficient input methods for other Indian languages should follow since this work provided efficient input methods for Telugu. Since there is a good deal of similarity among Indian languages, the approach to design and evaluation might not vary drastically. Further, more designs can be worked out providing new means of data input for Indian languages. None of the solutions here can claim to be a panacea for data input in Indian languages. They only provide better alternatives to the existing solutions. Thus, a more efficient keyboard layout which performs better than the proposed and already existing ones, a prediction system which offers better predictions than the current solutions, a soft keyboard which is easier to use than present solutions these are some of the directions that are to be worked upon as a part of our future work.

Innovative user-interaction strategies need to be designed as an extension to the problem, there by extending the work in to different application scenarios like user interfaces for multilingual and cross-lingual applications, including those in Information Retrieval. One immediate goal in this direction is extending the present work to the context of mobile text input. Mobile text input for Indian languages is a relatively unexplored area whose future appears promising. Hence, this work can be applied in that direction. Indian language text input using a 3*4 mobile phone keyboard is an interesting area of research. Hypothesizing that in the ”SMS” way of typing, vowels are the most likely ones to be skipped,[1] the problem of mobile text input in the context of Indian languages becomes the one of ”word prediction in the absence of vowels”. This can be viewed as a variant of Transliteration. However, obtaining training data, handling multiple spelling variations, indexing the corpus are some of the questions that arise. Further, the possibility of writing an SMS language in Indic scripts is another matter of doubt. All these questions make the problem

[1] http://en.wikipedia.org/wiki/SMS_language

more challenging.

The evaluations that we have performed in this thesis are mostly based on simple heuristics and have worked out well so far. However, more detailed study needs to be conducted on evaluating text input methods for computing devices. The possibility of evaluation techniques that utilize the characteristics and similarity among Indian languages also needs to be explored. User studies need to be conducted using different user-interface designs for data input in Indian languages. Screen monitoring softwares and eye-tracking devices can be used to understand and analyze the user behavior. The results of these studies may be useful in developing full fledged applications using the data input techniques proposed in this thesis. Finally, different scenarios in which these methods can be used needs to be studied. Possibility of using these input methods in applications like language teaching aids also can be explored.

Bibliography

[1] Report of the committee for standardization of keyboard layout for indian script based computers. *Electronics Information & Planning Journal*, 14(1):2444–2448, October 1986.

[2] Norman D. A. and Fisher D. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. In *Human Factors*, volume 24, pages 509–519, 1982.

[3] Rathod A, Joshi A, and Athvankar U. Devnagari text input device. Master of design project thesis, Indian Institue of Technology, Bombay, India, 2002.

[4] Elmagarmid A.K., Ipeirotis P.G., and V.S. Verykios. Duplicate record detection: A survey. In *EEE Transactions on Knowledge and Data Engineering*, volume 19, pages 1–16, 2007.

[5] Akshar Bharathi, Vineet Chaitanya, and Rajeev Sangal. *Natural Language Processing - A Panian Perpective*. Prentice-Hall of India, 1991.

[6] Lee Butts and Andy Cockburn. An evaluation of mobile phone text input methods. In *Australian Computer Science Communications*, volume 24, pages 55–59, 2002.

[7] Michael Capewell. Alternative keyboard layouts - a case study of different keyboard layouts for english. In *http://www.geocities.com/smozoma/projects/keyboard/layout_capewell.htm?20081*.

[8] Michael Capewell. Keyboard evolve - a c++ program/framework for evolving keyboard layouts. In *http://keyboardevolve.sourceforge.net/*.

[9] Zheng Chen and Kai-Fu Lee. A new statistical approach to chinese pinyin input. In *The 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2000.

[10] Zheng Chen and Kai-Fu Lee. Multilingual text entry using automatic language detection. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

[11] Colemak. The colemak keyboard. In *http://colemak.com/*.

[12] Norman D. *Design of Everyday things*. Basic Books, 1988.

[13] Andrew D.Wilson and Maneesh Agarwala. Text entry using a dual joystick game controller. In *CHI*, 2006.

[14] Eatoni Ergonomics. A survey of alternate text entry methods. http://www.eatoni.com/research/alternates.pdf.

[15] Ronald A. Fisher. *Statistical methods for research workers*. Oliver and Boyd, 1925.

[16] Nestor Garay-Vitoria and Julio Abascal. Text prediction systems: a survey. In *Universal Access in the Information Society*, volume 4, pages 188–203, 2006.

[17] Sandeva Goonetilleke., Yoshihiko Hayashi., Yuichi Itoh., and Fumio Kishino. An efficient and user friendly simhala input method based on phonetic transcription. In *Journal of Natural Language Processing*, volume 14, 2007.

[18] Sandeva Goonetilleke, Yoshihiko Hayashi, Yuichi Itoh, and Fumio Kishino. Srishell primo: A predictive sinhala text input system. In *Workshop on NLP for less privileged languages, IJCNLP-2008*, 2008.

[19] Renu Gupta. Technology for Indic Scripts,a user perspective. *Language in India*, 2006.

[20] Hakon Hallingstad. The Arensito keyboard layout. In *http://www.pvv.org/ hakonhal/keyboard/*.

[21] Karin Harbusch and Michel Kuhn. Towards an adaptive communication aid with text input from ambiguous keyboards. In *Proceedings of the tenth conference on European Chapter of the Association for Computational Linguistics - Volume 2*, pages 207–210, 2003.

[22] Yuzuru Hiraga, Yoshihiko Ono, and Yamada-Hisao. An assignment of key-codes for a japanese character keyboard. In *COLING.*, 1980.

[23] http://en.wikipedia.org/wiki/Keyboard_layout. Keyboard layout.

[24] http://www.tamilnation.org/digital/tamilnet99/report.htm. Final report of conference : Tamilnet99.

[25] Scott MacKenzie I. and Janet C. Read. Using paper mockups for evaluating soft keyboard layouts. In *CASCON.*, pages 98–108, 2007.

[26] Scott MacKenzie I. and Shawn X.Zhang. An empirical investigation of the novice experience with soft keyboards. *Behavior & Information Technology*, 20(6):411–418, 2001.

[27] Poika Isokoski. Performance of menu-augmented soft keyboards. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–430, 2004.

[28] Poika Isokoski and Roope Raisamo. Architecture for personal text entry methods. In Morten Borup Harning and Jean Vanderdonckt, editors, *Closing the gaps: Software engineering and Human-Computer Interaction*, 2003.

[29] Anirudha Joshi, Ashish Ganu, Aditya Chand, Vikram Parmar, and Gaurav Mathur. Keylekh: A keyboard for text entry in indic scripts. In *Computer and Human Interaction (CHI).*, 2004.

[30] De Jong J.R. The effects of increasing skill on cycle time and consequences of time standards. In *Ergonomics*, volume 6, pages 51–60, 1957.

[31] Kaplan and Wissink. Unicode and keyboards on windows. Technical report, Windows Globalisation, Microsoft Corporation report.

[32] Mathias Kolsch and Matthew Turk. Keyboards without keyboards: A survey of virtual keyboards. Technical Report 21, University of California - Santa Barbara (UCSB), July 2002.

[33] Per-Ola Kristensson and Shumin Zhai. Shark:a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 43–52, 2004.

[34] Martin Krzywinski. Reports at the carlpax keyboard layout optimizer project website. In *http://mkweb.bcgsc.ca/carpalx/?home*.

[35] Indian language computing from gist to unicode. http://groups.google.com/group/aksharamala/web/indian-language-computing—from-gist-to-unicode. Technical report.

[36] Prahallad Lavanya, Prahallad Kishore, and GanapathiRaju Madhavi. A simple approach to build transliteration editors for indian languages. *Journal of Zhejiang University Science*, 2005.

[37] Scott Mackenzie and Shawn K.Zhang. The design and evaluation of a high-performance soft keyboard. In *Computer and Human Interaction (CHI).*, 1999.

[38] Scott Mackenzie, Shawn K.Zhang, and R.W.Soukeroff. Text entry using soft keyboards. In *Behavior and Information Technology (BIT)*, 1999.

[39] Ganapathiraju Madhavi, Balakrishnan Mini, Balakrishnan N., and Reddy Raj. Om: One tool for many (indian) languages. In *Journal of Zhejiang University Science*, volume 6A, pages 1348–1353, 2005.

[40] Ranbeer Makin., Nikita Pandey., Prasad Pingali., and Vasudeva Varma. Experiments in cross-lingual ir among indian languages. In *International Workshop on Cross Language Information Processing (CLIP)*, 2007.

[41] Peter M.Klausler. Evolutionary algorithm to discover better keyboard layouts. In *http://klausler.com/evolved.html*.

[42] Mtgap. Designing the ultimate keyboard layout. In *http://mtgap.bilfo.com/keyboard.html*.

[43] Animesh Nayan., Ravi Kiran Rao B., Pawandeep Singh., Sudip Sanyal., and Sanyal. Ratna. Named entity recognition for indian languages. In *Workshop on NER for South and South East Asian Languages (NERSSEA), International Joint Conference on Natural Language Processing (IJCNLP)*, 2008.

[44] Prasad Pingali, Jagadeesh Jagarlamoodi, and Vasudeva Varma. Webkhoj: Indian language ir from multiple character encodings. In *International World Wide Web Conference, May 2006,*.

[45] Prasad Pingali. and Varma. Vasudeva. Word normalization in indian languages. In *4th International Conference on Natural Language Processing (ICON)*, 2005.

[46] Prasad Pingali, Suryaganesh Veeravalli, Sreeharsha Yella, and Vasudeva Varma. Statistical transliteration for cross language information retrieval using hmm alignment model and crf. In *Workshop on Cross Lingual Information Access (CLIA),Third International Conference on Natural Language Processing(IJCNLP)*, 2008.

[47] Soukoreff R. and MacKenzie I. Measuring errors in text entry tasks: An application of the levenshtein string distance statistic. In *Computer and Human Interaction (CHI).*, 2001.

[48] Amit Rathod and Anirudha Joshi. A dynamic text input scheme for phonetic scripts like devanagari. In *Proceedings of Development by design (DYD)*, 2002.

[49] Zhai S., Hunter M., and Smith B.A. The Metropolis Keyboard – An exploration of quantitative techniques for virtual keyboard design. In *In the Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 119–128, 2000.

[50] Zhai S., Kristensson P.O., and Smith B.A. In search of effective text input interfaces for off the desktop computing. *Interacting with Computers*, 17(3):229–250, 2005.

[51] Andrew Sears and Julie A.Jacko. *Handbook of Human Computer Interaction*, chapter Input Technologies & Techniques. Lawrence Earlbaum & Associates.

[52] Mauricia. Serva and Filippo. Petroni. Indo-european languages tree by levenshtein distance. In *Exploring the Frontiers of Physics (EPL)*, 2008.

[53] Shrinath Shanbhag, Durgesh Rao, and R. K. Joshi. An intelligent multi layered input scheme for phonetic scripts. In *Proceedings of the 2nd international symposium on Smart graphics*, pages 35–38, 2002.

[54] Anil Kumar Singh, Harshit Surana, and Karthik Gali. More accurate fuzzy text search for languages using abugida scripts. In *Improving Non-English Web Searching (iNEWS07) SIGIR07 Workshop*, pages 71–78, 2007.

[55] Barton A Smith and Shumin Zhai. Optimised Virtual Keyboards with and without alphabetical ordering A novice user study. In *INTERACT 2001 IFIP TC13 International Conference on Human-Computer Interaction.*, pages 92–99, 2001.

[56] Wanda Smith. Research study - ergonomic test of the kinesis contoured keyboard. Technical report, Kinesis Corporation.

[57] R.William Soukeroff and I.Scott Mackenzie. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. In *Conference on Human factors in computing systems*, pages 113–120, 2003.

[58] R.William Soukeroff and I.Scott Mackenzie. Recent developments in text-entry error rate measurement. In *CHI '04 extended abstracts on Human factors in computing systems*, pages 1425–1428, 2004.

[59] Murray Speigel and Larry Stephens. *Statistics*. Tata Macgrawhill, 2000.

[60] Acharya Team. Articles on indian language computing. http://acharya.iitm.ac.in.

[61] Andrew T.Freeman., Sherri L.Condon., and Christopher M.Ackerman. Cross linguistic name matching in english and arabic: A one to many mapping extension of the levenshtein edit distance algorithm. In *Human Language Technology Conference of the North American Chapter of the ACL*, pages 471–478, 2006.

[62] Gihan V.Dias and G Balachandran. Keyboards for Indic languages. Technical report.

[63] Christopher P. Walker. Evolving a more optimal keyboard. Course project: Introduction to evolutionary computation, Missouri University of Science & Technology, 2003.

[64] William W.Cohen., Pradeep Ravikumar., and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *American Association for Artificial Intelligence (AAAI)*, 2003.

[65] Wikipedia. Japanese input methods for computers. http://en.wikipedia.org/wiki/Japanese_language_and_computers.

[66] Wikipedia. Transliteration. http://en.wikipedia.org/wiki/Transliteration.

[67] J. Williamson and R. Murray-Smith. Dynamics and probabilistic text entry. Technical report, University of Glasgow, 2002.

[68] W.E. Winkler. *The State of Record Linkage and Current Research Problems.* Statistics of Income Division, Internal Revenue Service Publication, R99/04.

[69] Shumin Zhai, Michael Hunter, and Barton A. Smith. Performance optimization of virtual keyboards. In *CHI*, 2002.

[70] Shumin Zhai and Barton A Smith. Alphabetically biased Virtual Keyboards are easier to use layout does matter. In *Extended abstracts of CHI2001-ACM conference on Human factors in Computing systems.*, pages 321–322, 2001.