DESIGNING EFFICIENT VIRTUAL KEYBOARD FOR TEXT COMPOSITION IN BENGALI

Soumalya Ghosh

DESIGNING EFFICIENT VIRTUAL KEYBOARD FOR TEXT COMPOSITION IN BENGALI

Thesis submitted to the
Indian Institute of Technology Kharagpur
for award of the degree

of

Master of Science (by Research)

by

Soumalya Ghosh

Under the guidance of

Dr. Debasis Samanta

School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721 302, India
May 2013

CERTIFICATE OF APPROVAL

31/05/2013

Certified that the thesis entitled **Designing Efficient Virtual Keyboard for Text Composition in Bengali** submitted by **Soumalya Ghosh** to the Indian Institute of Technology, Kharagpur, for the award of the degree Master of Science has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Member of DAC) (Member of DAC) (Member of DAC)

(Member of DAC) (Member of DAC) (Member of DAC)

(Supervisor)

(Internal Examiner) (Chairman)

CERTIFICATE

This is to certify that the thesis entitled **Designing Efficient Virtual Keyboard for Text Composition in Bengali**, submitted by **Soumalya Ghosh** to Indian Institute of Technology Kharagpur, is a record of bona fide research work under my supervision and I consider it worthy of consideration for the award of the degree of Master of Science (by Research) of the Institute.

Date: 31/05/2013

Dr. Debasis Samanta
Associate Professor
School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721 302, India

DECLARATION

I certify that

a. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Soumalya Ghosh

Dedicated to

My parents and other family members

ACKNOWLEDGMENT

First and foremost I wish to convey my deep sense of gratitude to my mentor Prof. Debasis Samanta. It has been my blessed opportunity to be his student. I appreciate all his contributions in the form of time, idea, and greater vision to make my research experience productive and cherishable. I basically learned an approach of humanity, patience and hard working from him.

I would like to thank Prof. Jayanta Mukhopadhyay, Head of SIT for extending me all the possible facilities to carry out the research work. I also wish to thank all of my departmental academic committee members Prof. A. Gupta, Prof. C. R. Mandal, Prof. S. Sural, Prof. S. K. Ghosh, Prof. K. S. Rao, Prof. S. Misra, Prof. R. R. Sahay for their valuable suggestions during my research. I also like to thanks Prof. D. Roychowdhury and Prof. I. Sengupta for their valuable guidance and emotional support during the difficult times of my research work. I sincerely remember the support of office staffs Mithun Da, Soma Di, Pratap Da, Malay Da, Vinod Da and others. I am also grateful to all members of School of Information Technology.

I owe my deepest gratitude to Sayan Da, Debasish Da, Somnath Da, Barik Da, Sajal Da, Kamalesh Da, Priyankar Da for strengthening my research by constant moral support and providing necessary guidance when required. I really learned a lot from them. I wish to convey my heartfelt thanks to Pradipta Kumar Saha, Manoj Kumar Sharma, Santa Maiti, Arindam Dasgupta, Jayeeta Mukherjee, Sankar Narayan Das, Indira Mukherjee, Barun Saha, Sudhamay Maity, Sai Praveen Kadiyala, Pushpita Chatterjee, Soumya Maity, Kanchan Manna, Uttam Ghosh, Sudipta Bhattacharjee, Pramit Roy and many more.

I am greatly indebted to many of my friends for their constant inspiration. I used to receive frequent boosting calls from Avi, Tua, Khuku Di, Tanu Di, Dola Di, Rituparna Di and many more.

Nothing would have been possible without the moral support of my parents, uncles, aunts, brothers, sisters and their families. I deeply indebted to them. I would also thank to all the volunteers who participated in the user testing.

Soumalya Ghosh

Abstract

Virtual keyboard, an on-screen graphical display, where keys are arranged spatially and can be tapped by finger-tip, mouse pointer or stylus, is treated as an alternate text entry mechanism to primitive hardware keyboard such as *QWERTY* keyboard. An effective design of virtual keyboard is required to ensure the faster text entry with lower typing error rate. Existing virtual keyboard design methodologies consider text entry rate maximization as their primary objective. In other words, existing design approaches grossly ignore the users' typing error into account. In fact, occurrences of typing errors are more significant during text entry, particularly in Indian languages due to the linguistic features. Indeed, there is a need to design virtual keyboard which enhances the text entry rate along with minimum typing error rate. This work aims to model the typing error rate based on the virtual keyboard design parameters followed by an optimum design of virtual keyboard with respect to text entry rate maximization and error rate minimization. In this work, we have considered Bengali language; however the approach is applicable to many other Indian languages. We outline the different works carried out in the following.

First, we have identified different virtual keyboard design parameters, which influence the occurrence of typing error. We have performed several user experiments on those identified parameters. Statistical analysis on those experimental results concludes the significant impact of each parameter on typing error occurrence.

Next, we develop a model to predict typing error rate based on these parameters using Artificial Neural Network - Genetic Algorithm (ANN-GA) based hybrid approach. To prove the efficiency of our proposed model, similar model using ANN approach is also developed. Experimental results substantiate that our proposed ANN-GA model outperforms than ANN model.

Finally, we design a virtual keyboard based on optimization of two objectives namely maximize text entry rate and minimize typing error rate. The performance of the proposed virtual keyboard design approach is compared with existing work and we observe that, on an average, the text entry rate and error rate achieves 17% increment and 44% decrement over existing keyboards, respectively.

Keywords: Human computer interaction, virtual keyboard design, typing error modeling, soft computing applications, automatic design evaluation, text entry mechanism.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

List of Symbols

TE_{ctc}	Text entry rate for correctly typed characters
p_c	Crossover Probability
η	Learning Rate
μ	Momentum Constant
p_m	Mutation Probability

List of Abbreviations

AAC	Augmentative and Alternative Communication
ANN	Artificial Neural Network
ANN-GA	Artificial Neural Network-Genetic Algorithm
ANOVA	Analysis of variance
CPS	Character Per Second
ER	Error Rate
GA	Graphical Algorithm
GR	Grouping
GSC	Graphical Similar Character

HL	Hard-limit
InScript	Indian Script
KS	Key Size
LS	Log-Sigmoid
MAD	Mean Absolute Difference
MAPE	Mean Absolute Percentage Error
MaxTER	Text Entry Rate Maximized Virtual Keyboard
MLP	Multi-layer Perceptron
MSE	Mean Square Error
MT_{Mean}	Mean Motor Movement Time
MOGA	Multi-objective Genetic Algorithm
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
P	Purelin
PAES	Pareto-Archived Evolution Strategy
PDA	Personal Digital Assistant
SDE	Standard Deviation Error
SK	Space Between Keys
SPEA	Strength Pareto Evolutionary Algorithm
SpErOpT	Speed and Error Optimized Virtual Keyboard
TE	Text Entry Rate
TS	Tan-Sigmoid
WPM	Words Per Minute

Chapter 1

Introduction

Texts are combination of symbols used for communicating one person to other and also keeping as a record for future purpose. With the enormous advancement in digital world, text based communication has increased rapidly. Text composition thus become popular as people all over the world prefer communicating among themselves by e-mailing, messaging, chatting, blogging etc. through digital devices. Further, there is a necessity to interact in users' own mother languages. According to statistics, *Wikipedia* contains 475,163 articles in major Indian languages, and around 229,516 edit-operations are done per month on these data[1]. *Bengali Wikipedia* consists of 23,153 articles, out of which around 2,108,207 pages are viewed per month. Also, for Bengali language, around 15 new articles are added per day and 11,652 existing data are edited per month[2]. The above mentioned statistics substantiates the fact that text composition in users' mother language is becoming popular and we need a robust text composition mechanism.

Virtual keyboard, also called soft keyboard [9,11,24] is an on-screen graphical image map where character keys are arranged spatially and can be tapped by finger tip, mouse pointer or stylus, is treated as the most primitive alternate text entry mechanism [9,25]. Compare to its hardware counterpart, virtual keyboard can easily be designed for text composition for any language. To make virtual keyboard more usable and efficient different text entry rate enhancement strategies such as word prediction [26], completion [27], adaptation and personalization [28] etc.

[1] `http://stats.wikimedia.org/EN_India/TablesCurrentStatusVerbose.htm`
[2] `http://stats.wikimedia.org/EN_India/ReportCardIndia.htm`

can be incorporated with it. In addition to the text entry rate enhancement, we should look for the virtual keyboard design, which minimizes user typing error rate during text composition. We may note that in traditional virtual keyboard design approaches, only text entry rate improvement has been taken account. However, typing error rate influenced by virtual keyboard design parameters [29] should also be considered as an evaluation parameter along with movement time. This is particularly important in case of Indian languages, where the text entry becomes more error prone [30]. So, designing an Indian language compatible virtual keyboard without considering typing errors must narrow down the usefulness of the design. In fact, there is a genuine need to design a virtual keyboard which enhances the text entry rate along with diminished typing error rate. This work aims to model the typing error rate based on the virtual keyboard design parameters. Further, an optimum design of virtual keyboard is targeted which can maximize text entry and minimize typing error rate. In this work, we have considered Bengali as the target language; although, the approach is applicable to many other Indian languages.

The rest of the chapter is organized as follows. Section 1.1 gives a review of different text entry mechanisms. The advantages in virtual keyboard based text entry is stated in Section 1.2. The issues and challenges for Indian language compatible virtual keyboard design are discussed in Section 1.3. Section 1.4 describe the scope and objectives of our work. Finally, the outline of the thesis is presented in Section 1.5.

1.1 Different Text Entry Mechanisms

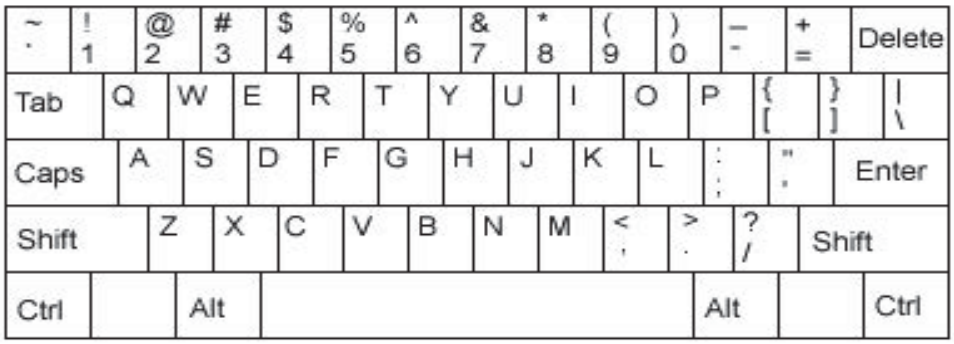
Recently, people are using a variety of devices and mechanisms to compose texts. In this regard, different mechanisms like hardware keyboard [1], virtual keyboard [12], speech-to-text systems [11], gesture-based [31] and eye gaze based text composition [32] are gaining popularity. A brief introduction regarding the different text entry mechanisms are described below.

- **Hardware Keyboard-based text composition:** The most popular method for composing text is the traditional hardware keyboard ($QWERTY$

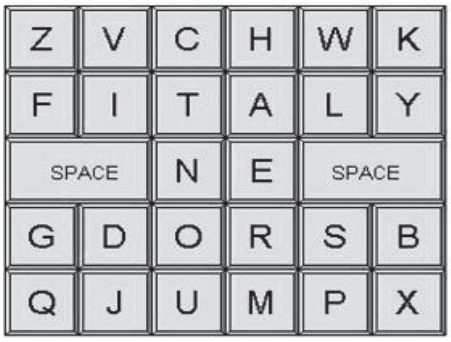
keyboard). The *QWERTY* keyboard (Fig. 1.1a), which was introduced in 1868 by Sholes et al. [1] was designed for text composition using ten fingers. However, *QWERTY* is not the effective design for text entry through single pointer based text composition in many devices like: hand-held mobile devices, PDA, iPod, Palmtops etc.

- **Virtual Keyboard-based text composition:** Virtual keyboard (Fig. 1.1b) is an on-screen graphical display, where keys are arranged spatially and can be tapped by finger-tip, mouse pointer or stylus. Virtual keyboard has several advantages such as it can be easily implemented in any language, easy to personalize, can be applied in the system with size restriction and it is portable compared to traditional hardware keyboard. All this advantages make the virtual keyboard more promising alternative than any other mode of text entry.
- **Gesture-based text composition:** It allows user to draw the shape of characters by pen stroke or stylus tapping (Fig. 1.1c). Gesture-based text entry method is very useful in small handheld devices where display area is too small to accommodate a virtual keyboard. However, in this method, users require to memorize gesture for each alphabet which increases the cognitive load [33].
- **Speech-based text composition:** Users compose texts by the method of speech to text conversion (Fig. 1.1d). This mechanism is suitable for the user who can not express their thought by writing. However, the accuracy of speech-to-text conversion becomes a real concern. Moreover, it has been reported that the text composition rate is lower than the keyboard-based text entry rate [12].
- **Eye gaze-based text composition:** In this mechanism, user enters text by the focus of gaze with the help of an eye-tracking device integrated with computing device (Fig. 1.1e). This mechanism is helpful for physically challenged users who are unable to move their hands and figures.

It may be noted that a particular mechanism of text composition facilitates a type of users. For example, speech and eye gaze-based text entry methods are suitable for users who are language illiterate and physical disabled respectively,



(a) Hardware keyboard-based text composition [1]



(b) Virtual keyboard-based text composition [2]

(c) Gesture-based text composition [3]

(d) Speech-based text composition [4]

(e) Eye gaze-based text composition [5]

Figure 1.1: Different mechanisms of text composition

whereas hardware keyboard-based text composition is suitable for literate able bodies users. Further, a mechanism may be suitable only in a particular situation. For example, the gesture-based text entry method is useful in small handheld devices where display area is the constraint.

Hardware keyboard-based text composition is not feasible in many cases like hand-held mobile devices, PDA, iPod, Palmtops etc. Addressing the size and portability issues, designers advocate virtual keyboard. Being a software component, a virtual keyboard can easily be enabled to support text entry in different languages and can also easily be augmented with different text entry enhancement strategies like word prediction, adaptation, personalization etc. More detail discussion on advantages of virtual keyboard-based text composition is given in the next section.

1.2 Advantages of Virtual Keyboard-Based Text Entry

In Section 1.1, we have discussed different text entry mechanisms. In this section, we pointed out some advantages of text entry through virtual keyboards.

- **Easy to implement:** Virtual keyboard is a software counterpart of hardware keyboard, which can be easily implemented for any device with out the use of any extra hardware.
- **Different language support:** One of the major benefits of virtual keyboard is language support. The virtual keyboard can easily be developed to support any language depending on users' need. Moreover, it can also be easily implemented for supporting multiple languages [34], which is not possible for hardware keyboard.
- **Easy to personalize:** Virtual keyboard interface can be easily personalized depending on individual user's need. For example, experienced users can use optimized keyboard layout whereas novice users can switch to their convenient one.
- **Adaptability:** It is also possible to provide adaptive virtual keyboard interface which can be changed in accordance with individual users

performance. If a user is unable to perform with a specific virtual keyboard interface, then an easier interface can automatically be provided to him.

- **Portability:** The virtual keyboard can easily be incorporated with small screen handheld devices like mobile phone, PDA, iPod etc. without any extra hardware. So there is no need of carrying extra hardware for text entry.
- **Augmenting text entry rate enhancement strategy:** The text entry rate can be improved incorporating text entry rate enhancement strategies into virtual keyboard like word prediction, abbreviation expansion, adaptation and personalization etc.
- **Reducing the risk of keystroke logging:** Virtual keyboards are used to enter secure data into web document to reduce the risk of keystroke logging which is possible in case of hardware keyboard.
- **Support for physically impaired:** Virtual keyboard is not only restricted as a text composition tool for able-bodied users, it is also found suitable for physically impaired users in the field of *Augmentative and Alternative Communication* (AAC). Scanning keyboard is one of such examples which is specially designed for motor impaired users [35].

1.3 Issues in Designing Virtual Keyboards in Indian Languages

The virtual keyboard design approaches in English are broadly confined within basic methodologies of language processing tasks like character frequency calculation (unigram and bigram) from a corpus and defining an association rule of characters in forming meaningful words in the English language. In contrast, the structure of Indian languages are completely different from English language (having indiscriminate usage of complex characters ("yuktakshar") and inflexions ("matra")); and hence it is quite complex to compose texts in Indian languages [36]. In addition to that, Indian languages have a large number of alphabet (50-80) which essentially has led to the formation of a large number of keyboards consisting of different combination of characters in the design space. It is, in

fact, a challenge to search an optimal design from such a large number of design alternatives. Moreover, typing errors are more significant for Indian languages compatible virtual keyboards [36]. It is mainly because of a large number of alphabet, along with many intricate linguistic features such as "matra", complex characters, graphically similar characters etc. All the above mentioned issues make the Indian language compatible virtual keyboard design problem more challenging.

Here, we are concerned about developing a keyboard in Bengali, the second most spoken language in India [37] and the primary language in eastern India and neighboring country Bangladesh. Further, it is also the world's seventh most spoken language [38]. We critically analyze different issues in the context of Bengali virtual keyboard design and summarize in the following.

- **Design space exploration:** In Bengali, the number of characters is more than double of English. More specifically, Bengali language contains 61 characters in contrast to only 26 characters in English. Hence, the number of possible layouts for a keyboard in design exploration space is higher than that of in English. It is quite challenging to find out a good design from such a large design space.
- **Large alphabet set** Compared to English (26 characters), Bengali language has a very large alphabet set (61 characters). Moreover, the size of alphabet set further increases with inclusion of different punctuation symbols and numbers. Accommodation of such a large alphabets set into a space constraint virtual keyboard interface is an issue.
- **Character diversity:** English script only contains two types of characters like vowels and consonants. However, Bengali script is divided into five different categories like inflexions ("matra"), vowels, consonants, sign characters and complex characters. Proper arrangement of so many different types of characters in a keyboard is quite tedious.
- **Presence of matra:** Unlike English, Bengali language contains many vowel inflexion ("matra") which are basically dependent vowels and cannot appear without a consonant. Those characters are frequent with respect to other character in Bengali text and also associated with almost every consonant characters with a high frequency.

- **Existence of complex characters:** Complex characters, which are constructed by combining two or more consonant characters together, are present in Indian languages. In fact, a special effort is required to compose complex characters.

The above mentioned issues make the text entry rate lower and typing error rate higher. A preliminary investigation of our work reveals that adopting the design principle of any popular English virtual keyboard does not give the effective result for Bengali text composition. So a different design approach for Bengali text entry is required.

1.4 Scope and Objectives

Significant advancement in Information and Communication Technologies makes a huge demand to design Indian language compatible virtual keyboard. In Indian language based text entry, due to the presence of many linguistic aspects, fluent text entry is difficult in bounded time. This situation, however, incurs many typing errors in composed text chunk. Apart from the user's ability, the cause of committing errors also depends on several design parameters of virtual keyboard. However, the traditional virtual keyboard design approaches consider only improvement of text entry rate. In existing literatures, no work is reported on predicting typing errors based on keyboard design parameters which, in fact, should be an important criteria while designing virtual keyboard. This work proposes to address this limitation. Our main objective is to design a keyboard layout to achieve higher text entry rate with lesser typing error. However, as every language has its own linguistic issues, we have limited our work to Bengali language. The objectives of our research are as follows.

- Identification of error influencing keyboard design parameters.
- Modeling of error rate based on identified parameters.
- Keyboard layout design with optimization of both text entry rate (maximization) and typing error rate (minimization).

1.5 Organization of the Thesis

The this thesis is organized into chapters as follows.

Chapter 2: Related Work

Chapter 2 provides a review of related work. It includes state of the arts for error classification and error modeling for virtual keyboard design. It also describes many virtual keyboard design approaches for English, other non Indian languages and Indian languages.

Chapter 3: Error Influencing Virtual Keyboard Design Parameter Identification

This chapter discusses typing error classification for Bengali virtual keyboard and user's typing error behavior. We then identify different virtual keyboard design parameters that have significant influence on typing error rate. Significance of each identified parameter is determined through different user experiments followed by the statistical test.

Chapter 4: Modeling of Typing Error Rate

The proposed typing error rate modeling based on the identified parameters using Artificial Neural Network-Genetic Algorithm (ANN-GA) hybrid approach is discussed in this chapter. The model is capable of predicting the user typing error rate for a virtual keyboard instance with different combinations of design parameter values. To prove the efficiency of our proposed model, similar model using ANN approach is also developed. Both the models are critically examined and tested through in-domain and out-domain data.

Chapter 5: Virtual Keyboard Design with Multi-Objective Optimization

In this chapter, we discuss about our Bengali virtual keyboard design approach based on multi-objective optimization using Non-dominated Sorting Genetic Algorithm II (NSGA-II). We use this multi-objective technique to develop a virtual

keyboard layout based on two optimality metrics: maximize text entry rate and minimize typing error rate. To judge the effectiveness of our multi-objective design approach, we design another alternate virtual keyboard using Genetic Algorithm (GA) for maximizing text entry rate only. Then, our proposed design is validated through different user experiments.

Chapter 6: Summary and Conclusion

This chapter concludes our study in the domain of virtual keyboard design. We also discuss the future research directions in this field.

Chapter 2

Related Work

The research work on error classification or categorization, error prediction model for virtual keyboards and virtual keyboard design approaches are reviewed in this chapter. First, we discuss reported work on error classification. It also provides an understanding about why users commit different typing errors. Then, we present existing error predictive models for virtual keyboard based text composition. Those models are tried to find out the relation between typing errors and virtual keyboard design parameters. Finally, we illustrate various notable work on virtual keyboard design methodologies in English and other non-Indian languages as well as Indian languages.

The organization of the chapter is as follows. Section 2.1 reviews different work on error classifications for text composition. Existing error predictive models for virtual keyboard based text composition are discussed in Section 2.2. Section 2.3 includes several work on virtual keyboard design approaches. Finally, the chapter is summarized in Section 2.4.

2.1 Error Classification for Text Composition

White [39] analyzed the accuracy of typing and highlighted twelve types of errors. This proposed categorization, however, did not formally define the errors. Dvorak et al. [6] extended White's work and introduced several other error types. The main contribution of Dvorak's work was introducing *Adjacent* error (falsely key stroke on the adjacent key of intended key such as typing; e.g. "amvition" for "ambition"),

Copy Reading error (read incorrectly from presented text; say, "admiration" for "ambition"), *Transposition* error (interchange of two consecutive strokes for two consecutive characters; e.g. "tiem" for "time"), *Transposed Doubling* error (two successive transposition errors such as 'thses' for 'these') etc.

MacNeilage [40] in his work tested text typing on 5 college students having average typing speed of 30-45 words per minute and identified a total of 623 errors. He then divided the identified typing errors into two groups namely *Spatial* and *Temporal* errors. The *Spatial* errors occur mainly due to keyboard design difficulty and *Temporal* errors come from language knowledge of users. Further, he divided the *Spatial* errors in three sub categories namely *Horizontal* error (typing a letter immediately to the left or right of the correct letter, of the same row of the keyboard; e.g. "e" for "r") , *Vertical* error (typing a letter immediately to above or below of the correct letter, of the same column of the keyboard; e.g. "f" for "r") and *Diagonal* errors (a letter is typed in a row and column adjacent of that correct letter; e.g. "d" for "r"). *Temporal* errors were sub-grouped as *Reversal* error (order of two letter is reversed; e.g. "ht" for "th"), *Omission*, *Equivocal* error (the letter one stoke ahead of the required in the copy is typed, after which the subject stops; e.g. "stiml-" for "stimulus") and *Anticipation* errors (a letter is typed which is required more than one stoke ahead of the place where it is mistakenly typed; e.g. "ext" for "expected"). Moreover, he also defined a set of miscellaneous errors such as *Interpolation*, *Phonemic*, *Type*, *Contralateral*, *Dynamic*, *Multiple Classification* and *Unclassifiable* errors.

Gentner et al. [41] introduced nine different typing errors such as *Misstrokes* error (an error which can be traced to inaccurate motion of the finder, as when one finger strikes two keys simultaneously), *Transposition*, *Interchange across I letters* error (two non-consecutive letters are switched with I letters intervening (I>0)), *Migration across M letters* error (one letter moves to a new position, with M letters intervening between its correct position and its end position (M>1); e.g. "orrecection" for "correction"), *Omission* error (a letter in a word is left out), *Insertion* (an extra letter is inserted into a text) error, *Substitution* error (a wrong letter is typed in place of the correct letter), *Doubling* error (word containing a repeated letter and the wrong letter is doubled instead; e.g. "caleed" for "called") and *Alternating* (a letter alternates with another but the wrong

alternation sequence; e.g. "threr" for "there") error. The *Alternating* error however, was already defined by Dvorak et al. [6] as *Transposed Doubling* error.

Bhagat et al. [42] analyzed on Punjabi text (a language in western India) and classified six types of errors in it. These are *Insertion error (IE)*, *Deletion error (DE)*, *Substitution error (SE)*, *Transposition error (TE)*, *Run-on Error (ROE)* (space missing between two or more valid words), *Split Word error (SWE)* (some extra space is inserted between parts of a word). This is a very basic level error classification and moreover, does not give any clue about why errors occurred.

Wobbrock et al. [43] returned back to the basic level of error classification by classifying it into *Insertion*, *Omission* and *Substitution* errors only to reduced the ambiguities in error classification. This type of high level classification also lacks in detailing of understanding the reason behind the occurrence of error.

Kano et al. [44] found total 1060 typing errors from a three days copy typing test of 112 children. Based on the analysis, they defined several new error types in addition to the previously defined error types. They extended the work of White [39] by introducing more word-level and phrase-level errors. The categorization method classified altogether 33 different types of errors at letter, word- and phase-levels.

Chen et al. [45] identified typing and pointing errors that affect mobile web users and motor impaired desktop users. They classified six different typing errors such as *Long key press*, *Bounce*, *Missing key*, *Transposition*, *Additional key* and *Key ambiguity errors*. On the other hand, with pointing task, they identified three other errors such as *Clicking*, *Multi-clicking* and *Dragging errors*. They also suggested to migrate some existing technique to address the errors (like - *Dynamic keyboard* [46]: long key press and bounce errors, *TrueKeys* [47]: additional key, missing key and transposition error, *SUPPLE++* [48]: pointing task errors). But those techniques are mostly device specific and not related with keyboard design parameters (*TrueKeys*) and moreover, useful for motor impairment users (*Dynamic keyboard*, *SUPPLE++*).

Our survey on the existing work on typing error reveals that the error categorization methods are highly dependent on language and device specific parameters. Also, it may be noted that the most of the work are based on English text composition through hardware keyboard. Our literature survey also reveals

that work on different parameters' influence on typing error rate, in the context of virtual keyboard design, has been scarcely reported.

2.2 Error Modeling

Bhattacharya et al. [49, 50] proposed a model to predict errors for different level of scanning in the context of scanning virtual keyboard design. There, text entry task was done by automatic scanning mechanism. The elements were highlighted one after another and user have to select the desired element through activation switch. They also identified three types of error for scanning keyboard based text entry such as *Timing* error, *Selection* error and *Transcription* error. *Timing* error occurs when a user fails to activate access switch when the element is highlighted. *Selection* error occur due to wrong element selection and *Transcription* error for entering a character that closely resembles with intended character. They modeled the user typing error for different level of scanning for motion impairment users. This error modeling approach is, however, not relevant to virtual keyboard with direct input that is, tapping with mouse pointer or stylus.

Jain et al. [29] proposed a predictive error model on virtual keyboard using curve fitting technique [51]. In their work, they only considered distance between the keys, and in fact, ignore many other critical design parameters. Moreover, their model is developed in the context of English virtual keyboard and thus limiting its usefulness to other languages, particularly Indian languages.

2.3 Virtual Keyboard Design

In this section, we review existing design principles of available virtual keyboards for text composition in English followed by some other major languages and mainly Indian languages. We also review the existing multi-objective virtual keyboard design approaches, which have been reported in the recent literature.

2.3.1 Virtual Keyboards in English

The design principles of existing English keyboards are reported below.

- ***QWERTY* keyboard:** *QWERTY* keyboard is the most commonly used text composition tool. The keyboard layout was invented by Sholes et al. [1] in 1868 for text composition through ten fingers and the name comes from the first six characters of keys in a keyboard [52]. The keys are arranged primarily to overcome the mechanical limitations inherited in typewriters and the layout (shown in Fig. 2.1(a)) has been adopted as a standard in 1971 by American Standards Institute. However, it is not an efficient design for text entry through single pointer based virtual keyboard [11].

- ***Dvorak* keyboard:** Dvorak et al. [6] developed a keyboard in 1932 through an analysis of the relative frequency of letters (Fig. 2.1(b)). The layout was designed for minimizing hand and finger movements [53]. The design principle of it also includes placing of vowels and frequently used consonants in such a way that it aids in typing with alternate hand. Moreover, the layout distributes the letters based on the strength of each finger, keeping most frequently used letters on the home row. However, *Dvorak* keyboard has not completely replaced the *QWERTY* keyboard due to long-term adaptation with the *QWERTY* keyboard, through it provides better text entry rate compared to *QWERTY* keyboard [53].

- ***FITALY* keyboard:** This keyboard was designed by *Textware Solution* to optimize pointer movement during text entry with mouse or stylus [2]. The design of *FITALY* layout includes dual sized double space bars at the two end of the home row (see Fig. 2.1(c)). The most common letters in English (e.g., E, T, A, H) are placed in proximity to the space bars. The keyboard's name is taken from the letter sequence along the second row of keys. Text entry through *FITALY* keyboard is faster than *QWERTY* keyboard [11].

- ***Chubon* keyboard:** Chubon keyboard [7] was designed to accelerate text entry rate for single-digit typists with a stylus (shown in Fig. 2.1(d)). It was designed by centralizing most commonly used letters in English language and placing frequently used letters together in their close vicinity.

- ***Cirrin* keyboard:** The layout was proposed by Mankoff and Abowd [8] to facilitate stylus based text entry. Input is generated from the coordinates of the points where the stylus crosses the interface of the circumference of

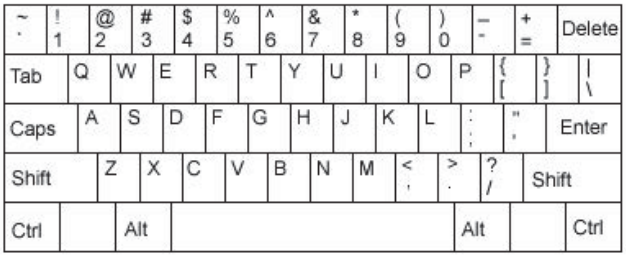
the inner circle in the layout. For example, composition of word "cirrin" is shown in Fig. 2.1(e).

- ***OPTI* keyboard:** MacKenzie and Zhang [9] proposed one optimized virtual keyboard, *OPTI* (shown in Fig. 2.1(f)), for English text typing. The keyboard layout was optimized to increase the typing speed using Fitts' law [54] and digram frequencies of characters in English. The key arrangement of the keyboard was done by trial-and-error method. *OPTI* keyboard [9] is 35% faster than *QWERTY* and 5% faster than compared to *FITALY* keyboard.

- ***Lewis* keyboard:** Lewis et al. [10] introduced a virtual keyboard layout in English alphabetical sequence (see Fig. 2.1(g)), which claim to perform better than *QWERTY* layout for single-finger text entry. However, the key arrangement is suffered in the alphabetical discontinuity that is caused by row breaking problem [55].

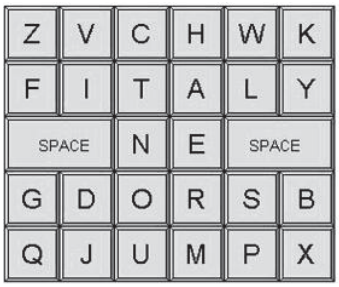
- ***Metropolis* keyboard:** Zhai et al. [11] designed virtual keyboard using Metropolis algorithm, a *Monte Carlo* method, which is widely used in searching for the minimum energy state in statistical physics [56]. The keyboard design problem can be mapped to searching for the structure of a molecule (keyboard) at a stable low energy state determined by the interactions among all the atoms (keys). The approach followed *Random walk* in the virtual keyboard design space. In each step of the walk, the algorithm picked a key and moved it to a random direction by a random amount to create a new configuration, which is followed by the evaluation of the new configuration. Then, the new configuration was considered as the starting position based on the metropolis function. *Metropolis* keyboard (shown in Fig. 2.2(a)) is 40% faster than *QWERTY* and 10% faster than *OPTI* keyboard [11].

- ***ATOMIK* keyboard:** ATOMIK keyboard [12] (shown in Fig. 2.2(b)) was designed to address the problems with *QWERTY* keyboard and get better text entry speed for single-finger typing. It achieves better movement efficiency than any other existing stylus keyboards in terms of mouse or stylus typing. The layout was alphabetically tuned having general trend

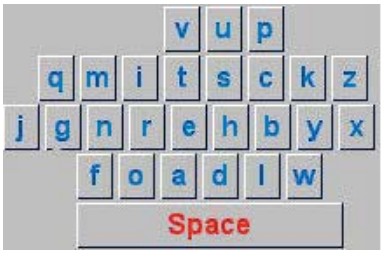


(a) *QWERTY* keyboard [1]

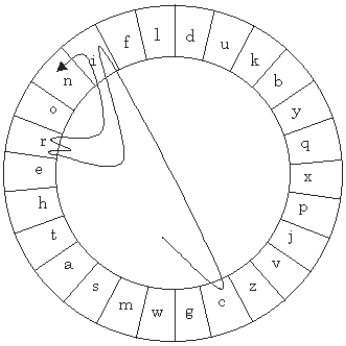
(b) *Dvorak* keyboard [6]



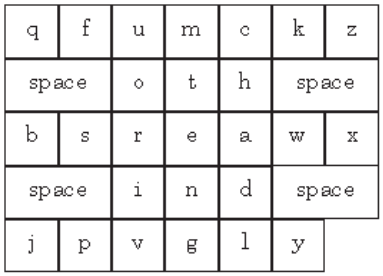
(c) *FITALY* keyboard [2]



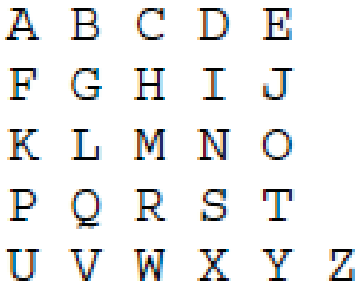
(d) *Chubon* keyboard [7]



(e) *Cirrin* keyboard [8]



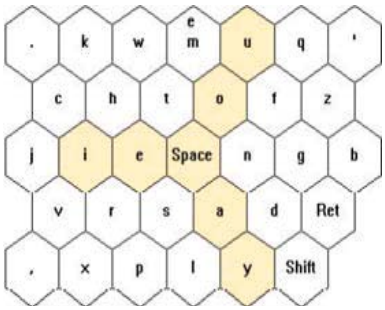
(f) *OPTI* keyboard [9]



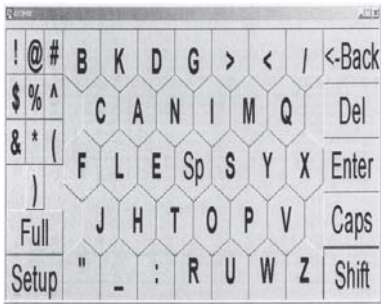
(g) *Lewis* keyboard [10]

Figure 2.1: Some virtual keyboard layouts in English

that letters from A to Z are placed from the upper left corner to the lower right corner of the keyboard. The arrangement reduces the mental load of novice users in finding keys.



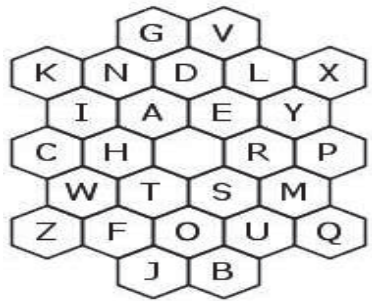
(a) *Metropolis* keyboard layout [11]



(b) *ATOMIK* keyboard layout [12]

Figure 2.2: *Metropolis* and *ATOMIK* virtual keyboard layouts

- ***GAG* keyboard:** Raynal et al. [13] used the concept of genetic algorithm [57] to solve the virtual keyboard optimization problem in English language. They considered the average mouse movement time as fitness function and tried to maximize the text entry speed. They took two existing layouts (*Metropolis* and *OPTI*) as inputs for the algorithm and obtained two solution (based on key arrangement) which are called *GAG I* and *GAG II* keyboard (shown in Fig. 2.3(a) and Fig. 2.3(b) respectively).



(a) *GAG I* keyboard layout [13]

Q	Y	P	U	B	J
X		S	T		F
K	L	A	H	O	M
C	I	N	E	R	V
	G	D		W	Z

(b) *GAG II* keyboard layout [13]

Figure 2.3: *GAG I* and *GAG II* virtual keyboard layouts

2.3.2 Virtual Keyboards in other Non-Indian Languages

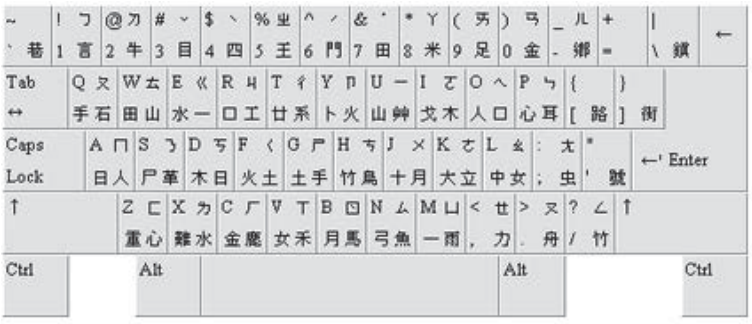
We also review literatures of some non-Indian (other than English) keyboard layouts. In our survey, we choose four popular languages (written and spoken) of the world namely Arabic, Chinese, Japanese and Russian. Descriptions of each layout is given in the following.

- ***Arabic* keyboard:** This layout is used to enter text in Arabic language and is developed by *Microsoft*. It contains both Arabic and Latin, as Latin characters are necessary for composing URLs and Email addresses (Fig. 2.4(a)) [14].
- ***Chinese* keyboard:** The Chinese traditional keyboard [14], shown in Fig. 2.4 (b), incorporates three input methods in a single layout namely *Zhuyin* (upper right), *Cangjie* (lower left) and *Dayi* (lower right). Here, *Zhuyin* follows *bopomofo* style that is, lexicographical order (top-to-bottom left-to-right) of designing keyboard and *Cangjie* is the standard method for speed typing in traditional Chinese.
- ***Japanese* keyboard:** Japanese keyboard (as shown in the Fig. 2.4(c)) [14] uses both *Hiragana* and Roman letters. The Japanese Industrial Standard keyboard layout keeps the Roman letters in English *QWERTY* layout. Many of the non-alphanumeric symbols are the same as in English language keyboards.
- ***Russian* keyboard:** The most common keyboard layout in Russia which is used in the Windows operating system, is shown in Fig. 2.4(d) [14]. This layout allows using keyboards of the same physical design as in many other countries. Here, the comma and full stop symbols are on the same key and users need to hold the *Shift* key every time they enter a comma.

2.3.3 Virtual Keyboards in Indian Languages

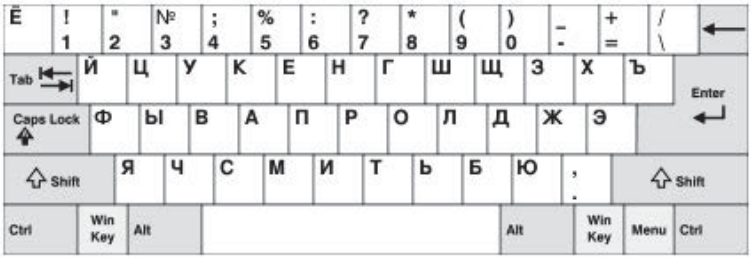
Various layouts are proposed to enter texts in Indian languages which have been incorporated in hardware as well as in virtual keyboards. This section highlights some of the designs implemented in Indian language virtual keyboards.

(a) Arabic keyboard layout



(b) Chinese keyboard layout

(c) Japanese keyboard layout



(d) Russian keyboard layout

Figure 2.4: Virtual keyboard layouts in non-Indian languages [14]

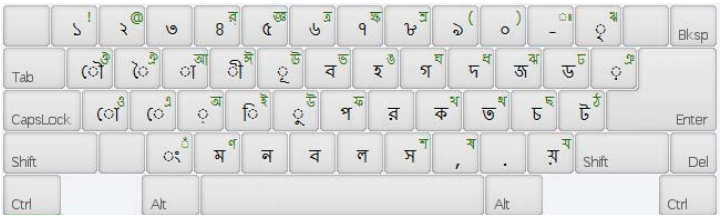
InScript (Indian script) keyboard layout (Fig. 2.5) [15, 58], developed by *Centre for Development of Advanced Computing* (CDAC), has been standardized by Government of India for composing text in Indian languages [59]. This is the standard keyboard available for 12 Indian language scripts including Devanagari, Bengali, Gujarati, Kannada, Malayalam, Oriya, Tamil, Telugu etc. Further, *Microsoft* also have designed an *InScript* on-screen virtual keyboard which supports text entry task in Hindi as well as in other Indian languages [60].

Figure 2.5: InScript keyboard layout [15]

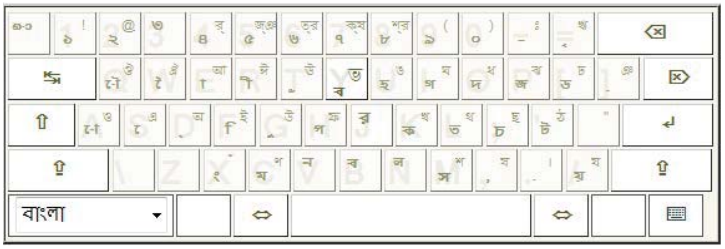
As the hardware keyboards are mostly made with English, stickers can be stuck on top of the keys so that one can understand character mapping for a particular language. Now, as the number of characters present in Indian languages are more than English, mapping all characters into keyboard is an issue.

Virtual keyboards, following *InScript* layout [58], contain each key having two character mapping which can be accessed using "Shift" key alternatively (Fig. 2.6a - 2.6c) [16–18]. However, *InScript* keyboard layout is not a text entry efficient design for single pointer based virtual keyboards. The text entry rate for those keyboards varies form $3-5$ *WPM* approximately [21].

A popular virtual keyboard layout *Avro* [19] for Bengali language is shown in Fig. 2.7(a). In this layout, all consonants are alphabetically placed into two parts and all vowels are presented sequentially in a row at the bottom of the keyboard. *Matras* are placed in a separate row in between consonant panel and vowel row. *Guruji* [20] has developed alphabetical keyboard to support users in composing Indian language texts. This keyboard uses different colors to distinguish consonants, vowels and numeric keys (shown in Fig. 2.7(b)) to efficiently search the keys during text typing.



(a) *Gate2Home* keyboard [16]



(b) Lipik keyboard layout [17]

(c) Lookeys keyboard layout [18]

Figure 2.6: Different *InScript* Bengali virtual keyboard layouts

Samanta et al. [21] developed virtual keyboards for text composition in Hindi (*iLiPi-H*), Bengali (*iLiPi-B*), and Telugu (*iLiPi-T*) languages, where keys are arranged according to their frequency. The arrangement of keys is carried out by using Genetic Algorithm [57]. Figure 2.8 shows the *iLiPi-T* keyboard layout.

2.3.4 Virtual Keyboard Design with Multi-Objective Optimization

Majority of the work on virtual keyboard design have attempted to maximize text entry rate by arranging the key positions. There are some approaches to design virtual keyboard considering the problem of optimizing multiple objectives, which are discussed in the following.

(a) *Avro* Bengali keyboard [19]

(b) *Guruji* Hindi keyboard [20]

Figure 2.7: Alphabetical virtual keyboard layouts

Figure 2.8: *iLiPi-T* keyboard layout [21]

Eggers et al. [61] designed the keyboard by optimizing six ergonomic criteria, namely, tapping load distribution, number of keystrokes, hand alternation, finger alternation, finger posture, and hit direction, using Ant Colony Optimization algorithm [62]. Deshwal et al. [63] used Eggers criteria as the performance indexes and developed an optimal Hindi keyboard using Genetic Algorithm [57]. We may note that those ergonomic criteria are not applicable to virtual keyboard design.

Yin et al. [64] optimized ergonomic criteria and disambiguation/prediction effectiveness simultaneously using Cyber Swam method [65] for keyboard designing. Their keyboard also outperforms *QWERTY* and *Dvorak* keyboard.

Quasi-Qwerty [22] keyboard layout(Fig. 2.9) was designed for optimizing both

average motor movement time to enhance text entry rate and designing layout close to *QWERTY* to reduce initial visual search time. The keyboard achieved 12% faster text entry rate than *QWERTY* keyboard. Though *Quasi-Qwerty* is slower than *ATOMIK*, but it requires less initial text entry time.

q	w	d	r	t	u	y	l	k	p
z	a	s	e	h	n	i	o	m	
	x	f	v	c	g	b	j		

Figure 2.9: *Quasi-Qwerty* keyboard layout [22]

Dunlop et al. [23] used multi-objective optimization to design keyboard layout for touchscreen phone based on three design metrics: minimizing finger travel distance in order to maximize text entry speed, a new metric to maximize the quality of spell correction by reducing tap ambiguity and maximizing familiarity through a similarity function with the standard *QWERTY* layout. Using the methodology, they designed two new keyboard namely, *Sath-Trapezoidal* (Fig. 2.10(a)) and *Sath-Rectangular* (Fig. 2.10(b)).

(a) *Sath-Trapezoidal* keyboard [23] (b) *Sath-Rectangular* keyboard [23]

Figure 2.10: *Sath* keyboard layouts

2.4 Summary

In this chapter, we have discussed various typing error classification approaches. Proper classification and understanding of errors help designers to find out the

relationship between typing errors and keyboard design parameters. This also supports designers to come up with effective error prevention technique. Our survey reveals that the error categorization methods are highly dependent on language and device specific parameters. As an example, *Pointing* errors are not occurred in hardware keyboard based text entry. It may also be noted that most of the works are based on English text composition through hardware keyboard.

Our literature survey concludes that work on modeling typing error rate based on error influencing virtual keyboard design parameters, has been scarcely reported. However, existing work are based on scanning keyboard or English virtual keyboard only taking distance between keys, which are not applicable to single pointer based Bengali virtual keyboard context. Moreover, there is also a need to check how other virtual keyboard design parameters put impact on typing errors.

We have surveyed, we also describe different virtual keyboard design approaches in English and other languages. It may be noted that the English virtual keyboard design approaches are not directly applicable to other language compatible keyboard due to linguistic issues like large number of characters, presence of complex and "matra" characters etc. With reference to virtual keyboard for Bengali text entry, most of the existing Bengali virtual keyboard are designed by either in alphabetical order or following the *InScript* mapping technique. Both techniques do not provide virtual keyboards having higher text entry rate. Moreover, we also analyze several multi-objective virtual keyboard design principles for English keyboard, which optimize text entry rate along with tap ambiguity. However, those do not consider user typing error rate as an objective to design virtual keyboard having less error rate. In the next chapters, the limitations in existing work have been addressed to develop typing error prediction model and design a multi-objective Bengali virtual keyboard to increase text entry rate and decrease typing error rate. In order to develop the error model, it is necessary to identify the virtual keyboard design parameters which have significant influence on typing error. Identification of error influencing virtual keyboard design parameters is discussed in the next chapter.

Chapter 3

Error Influencing Virtual Keyboard Design Parameter Identification

Towards the automatic evaluation of virtual keyboard designs, designers mainly emphasize on text entry rate. To the best of our literature survey, none of the existing works on virtual keyboard design evaluation consider typing errors as another metric for the design evaluation. Nevertheless, in practical situation, it has been observed that typing errors highly influence the text entry rate. Moreover, typing errors are more significant for Bengali compatible virtual keyboard as the language contains a large number of characters with many intricate linguistic features such as diacritics, complex characters, graphical similar characters etc. It may also be noted that in the context of Bengali, a single character error may require multiple number of edit primitives to correct the error, affecting the text entry rate. So, designing a Bengali compatible virtual keyboard without considering typing errors narrow down the practical usefulness of that design.

This work attempts to bridge this gap. As a first step towards an automatic typing errors evaluation, a model is necessary to predict user typing error behavior. To achieve this, it is essential to identify the virtual keyboard design parameters which significantly influence typing errors. With this objective, in this chapter, we have conducted an empirical study to analyze the effects of various parameters on typing errors. Finally, we critically examine the significance of each parameter on typing error rate or percentage of error occurred during typing.

This chapter consists of four sections. Section 3.1 states typing error

classification for Bengali text composition through virtual keyboard. Then analysis of users' typing error behavior to identify error influencing parameters is discussed in Section 3.2. Experiments and experimental details in order to analyze the significance of each parameter is presented in Section 3.3. Finally, Section 3.4 contains the summary of the chapter contents.

3.1 Error Classification for Bengali Virtual Keyboard

In this section, we classify the different types of errors that may occaur while typing through a single tap Bengali virtual keyboard. We first enlist all possible errors that may occur during text composition, from earlier literatures, and then classify them in the context of Bengali. Moreover, we also introduce several new error types like *Omitted Halant or Matra (OHM)*, *Inserted Halant or Matra (IHM)*, *Substituted by Graphically Similar Character (SG)* errors, which are specially related to Indian languages. A summary of classification is shown in Table 3.1.

Table 3.1: A classification of Bengali virtual keyboard text typing errors

Omission Errors	Substitution Errors	Insertion Errors	Other Errors
Omitted Letter (OL)	Alternating Error (AE)	Duplicated Letter (DL)	Unknown (U)
Omitted Halant or Matra (OHM)	Close to Error- Substitution (CT-S)	Inserted Letter (IL)	
	Doubling Error (DE)	Inserted Halant or Matra (IHM)	
	Next to Error- Substitution (NT-S)	Close to Error-Insertion (CT-I)	
	Substituted by Graphically Similar Character (SG)	Next to Error-Insertion (NT-I)	
	Substituted Letter (SL)		
	Transposition Error (TE)		

Omission Errors

- *Omitted Letter (OL)* : When a letter is omitted from the word when it is typed, it is classified as an *OL* error (example: একন [ekana] for একজন [ekajana]).
- *Omitted Halant or Matra (OHM)* : When a halant or matra character is omitted from the word while typing (example: তপত [tapata] for তপ্ত [tapta], গতম [gatama] for গৌতম [gautam]).

Substitution Errors

- *Alternating Error (AE)* : A letter alters with another but in a wrong alternation sequence (example : টকট [TakaTa] for কটক [kaTaka]).
- *Close to Error-Substitution (CT-S)* : Intended letter is substituted by a letter close to the intended letter (above or below adjacent keys) of the keyboard (example : দলকে [dalake] for দশকে [dashake]).
- *Doubling Error (DE)* : The word contains a repeated letter, however a wrong letter is doubled instead (example : শধধর [shadhadhara] for শশধর [shashadhara]).
- *Next to Error-Substitution (NT-S)* : Intended letter is substituted by a letter next to the intended letter (left or right adjacent keys) of the keyboard (example : দচকে [dachake] for দশকে [dashake]).
- *Substituted by Graphically Similar Character (SG)* : When an incorrect letter having graphical similarity with intended letter is substituted (example: ঘব [ghaba] for ঘর [ghara]).
- *Substituted Letter (SL)* : When an incorrect letter substitutes the intended letter which do not belong to any other substitution error types (example: অভিষেদ [abhiSheda] for অভিষেক [abhiSheka]).
- *Transposition Error (TE)* : When two consecutive letters are switched by each other (example: একনজ [ekanaja] for একজন [ekajana]).

Insertion Errors

- *Duplicated Letter (DL)* : A character is erroneously repeated twice (example: রততন [ratatan] for রতন [ratan]).

- *Inserted Letter (IL)* : An extra letter (not a duplicate of the pervious letter) is inserted (example: অবস্থান [abasthaana] for অবস্থা [abasthaa]).
- *Inserted Halant or Matra (IHM)* : An extra "halant" or "matra" letter is inserted in the typed text (example: কখনি [kakhani] for কখন [kakhan], আলপ্না [AlapnA] for আলপনা [AlapanA]).
- *Close to Error-Insertion (CT-I)* : An extra letter close to the intended letter in the keyboard is inserted along with the intended letter (example : দশলকে [dashalake] for দশকে [dashake]).
- *Next to Error-Insertion (NT-I)* : An extra next to letter of the keyboard is inserted along with the intended letter (example : দশচকে [dashachake] for দশকে [dashake]).

Other Errors

- *Unknown (U)* : When the error does not fit into any of the above mentioned category, it is classified as an *Unknown error*.

3.2 Analysis of Users' Typing Error Behavior

Users' typing errors are influenced by virtual keyboard design parameters. According to Salthouse [66], 30.1% of substitution errors are involved with horizontally (NT-S error) or vertically (CT-S error) adjacent keys and more than 54% of insertion errors are occurred due to an adjacent key placed in the same row (NT-I error) or same column (CT-I error) in keyboard. Those substitution and insertion errors are directly related with different parameters like: space between keys and size of the keys. In addition to this, possibly every type of error occurrence may be influenced by space between keys and key size. Moreover, parameters like varying key size and space between keys in a particular keyboard, multiple characters assigned in a particular key also put impact on error occurrence. Substitution error with graphical similar character (*GSC*) pairs (SG error) are directly associated with key positioning and distance between *GSC* pairs in the keyboard. So, there is a need to examine how distance between those characters pairs put impact on the error occurrence. Typing errors are also influenced by the

size of a keyboard [67]. Several other design parameters like coloring, number of keys, key grouping, multiple space keys and number of rows and columns in the keyboard may put an impact on the occurrence of typing errors.

Out of the several parameters as pointed above, the area of keyboard is directly related with the size of keys and space between keys. Further, varying key sizes and/or space between keys is/are not a prudent choice of designers. Considering the above, we ignore layout area of a keyboard, keys of variable sizes and different separation among keys. The effect of coloring is very much subjective to a particular user. So, we do not consider it as a parameter for the experiment. Typing errors are also influenced by number of keys present in the keyboard [67]. But, it remains fixed in a specific language virtual keyboard. So, it is not possible to evaluate the impact on errors by varying the number of keys present in a keyboard. In contrast, alphabetical grouping of keys may have moderate impact on error rate in a keyboard [52]. Some virtual keyboards like *OPTI* [9], *FITALY* [2] etc. has multiple space keys to reduce the average mouse movement time. We need to investigate the impact of number of space keys on typing error rate. We list the parameters which are applicable in our context as below.

- Key size
- Space between keys
- Distance between *GSC* pairs
- Key grouping
- Number of space keys

Another important parameter which influences user's typing error behavior is the multiple characters on a key. In this work, we limit our investigation to virtual keyboards consisting of single character per key. In the following sections, we discuss our experiments and observation of user's typing error behavior on the above mentioned design parameters.

3.3 Experiments and Experimental Details

Several past work identify different typing error influencing keyboard design parameters [52, 66, 67]. However, none of these work analyzed or quantified the

significance of impact of those parameters on typing error. In this work, we tried to bridge this gap. In order to identify the significance of each parameter, several user experiments have been carried out which are discussed in this section.

3.3.1 Experimental Setup

Desktop PCs with 17" color monitor display with 1280×1024 resolution are used to conduct all the experiments. Each PC has Windows 7 running on Intel®Core^{TM}2 Duo processor with 2.4 GHz clock speed. The developed virtual keyboard interfaces for these experiments are designed by C# using Visual Studio 2008. All user typing tasks are automatically stored in a log file. The experiments are conducted in the HCI Lab, School of Information Technology, IIT Kharagpur.

3.3.2 Virtual Keyboard Interfaces Used

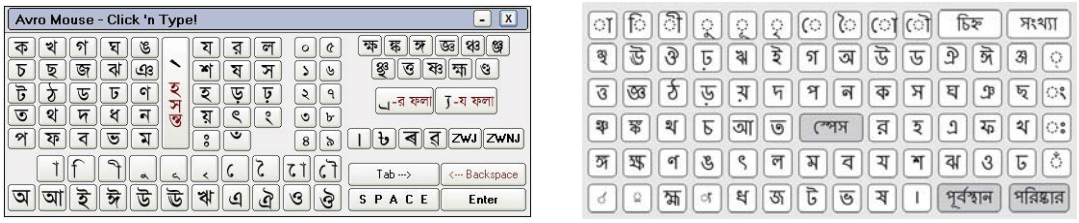
We have considered two different types of virtual keyboard for our experimental studies namely *Avro* [19], a popular Bengali virtual keyboard developed by OmricornLab, *iLiPi-B* keyboard proposed by Samanta et al. [21]. Both the keyboard interfaces are shown in Fig 3.1.

Avro **:** This keyboard is a single tap Bengali virtual keyboard and keys are arranged in alphabetical sequence. All consonants are alphabetically placed in two parts as shown in Fig. 3.1a. All vowels are presented sequentially in a row at the bottom of the keyboard. *Matras* are placed in a separate row in between consonant panel and vowel row.

iLiPi-B **:** This is also a single tap Bengali virtual keyboard where keys are arranged according to their frequency(Fig. 3.1b). The arrangement of keys is carried out by using Genetic Algorithm [57].

3.3.3 Participants

Users play most important role in the design and evaluation phase of human computer interaction (HCI). User classification poses a great challenge to the HCI researcher because of a large variety of user profiles based on tasks performed. Here, our target users are Bengali literate people who can read and write Bengali.



(a) *Avro* keyboard layout [19] (b) *iLiPi-B* keyboard [21] layout

Figure 3.1: Virtual keyboard used in the experimental studies

Forty eight unpaid volunteers from our Institute, local area and market, school and offices have participated in our experiments (28 males and 20 females). Depending on their background, professional information and familiarity with computer, users are classified into three categories: expert, intermediate and novice. Familiarity with computer is determined through a questionnaire session on Bengali typing task using virtual keyboard and general window based applications. Users belong to expert category are post-graduate and under graduate students from engineering colleges, school teachers and office staffs having prior knowledge of using computers. The intermediate users are high school students, shopkeepers, and security guards. Users in this category use computer occasionally (e.g. once in a week or less than that). On the other hand, primary school students, and domestic helpers belong to novice category. They have never used computer earlier. Moreover, to make the proper mixture in each category, we select equal number of users in each category (16 users). Details are shown in Table 3.2.

3.3.4 Text Selection

We select four different text corpus from popular Bengali novels, story book and Bengali "Wikipedia" by taking care of the occurrence of almost all characters of Bengali. We have selected portion of texts from *"Kapalkundala"*, a Bengali novel written by *Bankim Chandra Chattopadhyay* (TB_1), restrained with a large number of complex and inflexed characters, *"Srikanta"*, another novel with a large number of simple words, written by *Sarat Chandra Chattopadhyay* (TB_2), *"Ashamanjababur Kukur"*, a short story of *Satyajit Ray* (TB_3) and *Wikipedia* corpus (TB_4), inscribed with very simple words. Details are shown in Table 3.3.

Table 3.2: Users profile

User Category	Background Information	Professional Information	Score in Questionnaire Session	Number of Subjects
Expert	Well educated, good knowledge in Bengali computer applications	Post and under graduate students, college and school teachers, office staffs	more than 80%	16
Intermediate	School level educated and average familiarity with computer	Shopkeepers, security grads and high school students	$50\% - 80\%$	16
Novice	Primary school level educated and poor familiarity with computer	Primary school students, domestic helpers	below 50%	16

Table 3.3: Selected texts in the experiments

Text Under Test	Reference Text	Word Count	Character Count			Inflexion Count
			No Space	With Space	Complex	
TB_1	"Kapalkundala" by Bankim Chandra Chattopadhyay	171	1035	1205	75	337
TB_2	"Srikanta" by Sarat Chandra Chattopadhyay	186	885	1070	39	285
TB_3	"Ashamanjababur Kukur" by Satyajit Ray	196	997	1192	34	303
TB_4	"Wikipedia" Bengali	167	861	1027	37	291

3.3.5 Experiments

The experimental study consists of two parts; training session and testing session. The training session is required to make the user familiar and accustom with text composition task using virtual keyboards. In this session, initially, we give half an hour introductory session on Bengali text composition using the virtual keyboard. After that, a fifteen minutes short break is given to them. Then, keyboards are given to each user separately for typing practice.

After completion of training session, testing session with same keyboard interfaces is started. To study the impact of each identified parameter, we redesign the keyboards by changing the value of each parameter separately while keeping all other parameters fixed. Moreover, for a specific experiment, we maintain equal value for that parameter. In each experiment, a randomly selected keyboard interface is given to each user for typing the selected text corpus freely. They do not have to think about the occurrences of errors during text composition. A fifteen minute break is also provided in between two consecutive sessions for relaxation. Moreover, a user can attend at most four typing sessions in a day. Through the experiments, we want to analyze how typing errors are influenced by different design parameters. So, we disable the modifier keys (backspace, delete keys) of the keyboards not giving any provision for correcting the errors which are already committed. During experiments, the composed text through each keyboard are stored into a log file for each user from which user typing error rate is calculated for any particular design parameters value. Typing error rate is basically the percentage of error committed by the user in the time of typing as shown in Eqn. 3.1, where length of the typed text by the user is represented by $|T|$.

$$Error\ Rate = \frac{Number\ of\ errors}{|T|} \times 100 \tag{3.1}$$

3.3.6 Experimental Results

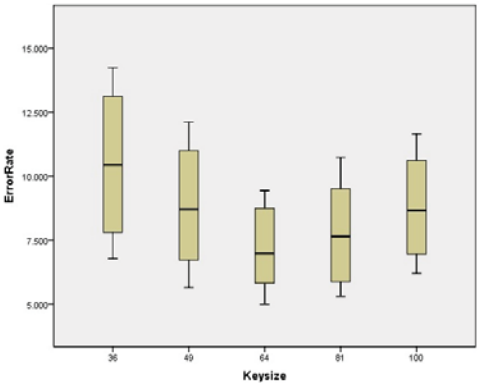
We have listed the design parameters which influence user error rate in Section 3.2. Impact and significance of those parameters are described in this section.

Key Size: In the keyboard interface, key size refers to the area occupied by the key and font size is specified as the maximum size displayed on a key. Here, we consider only square shaped keys. A number of experiments have been performed to observe the influence of key size on user typing error rate during text composition. The experiments are performed by varying size of the keys keeping all other parameters fixed. Also, for a particular experiment, we maintain same key size in a keyboard. In this scenario, key size is varied from 36 mm^2 (6×6) to 100 mm^2 (10×10) with an increment of 1 mm in both key length and width.

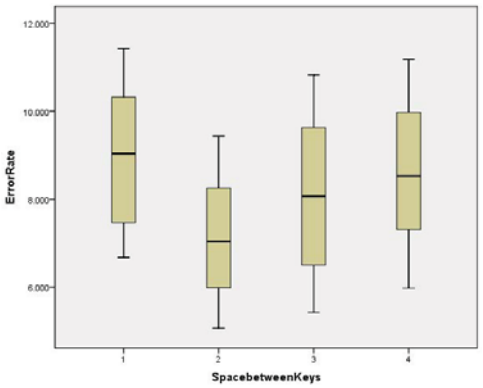
The analysis of the log file is illustrated graphically in Fig. 3.2(a) where key size and corresponding typing error rates are represented through x- and y-axis, respectively. The experimental outcomes of all users show that typing error rate is quite high for small key size (like 36 mm^2). The error rate decreases with increasing key size (upto a moderate value (64 mm^2)). But when the key size is high enough (81 mm^2), the curve corresponding to typing error rate slightly increases. It can be concluded that user error rate for most of the users is minimum with moderate size of the keys. Further, Analysis of variance (*ANOVA*) test reveals that, there is a significant difference between typing error rate on different values of key size, $F_{(4,175)} = 16.058$, where $p < 0.05$.

Space between Keys: In the context of virtual keyboard, space between keys refers to the horizontal or vertical gap or space between adjacent keys. Here, we consider equal horizontal and vertical space for any adjacent key for a particular keyboard interface. The experiments are performed by varying space between keys keeping all other parameters fixed. For these experiments, the space between keys varies from 1 to 4 mm with an increment of 1 mm.

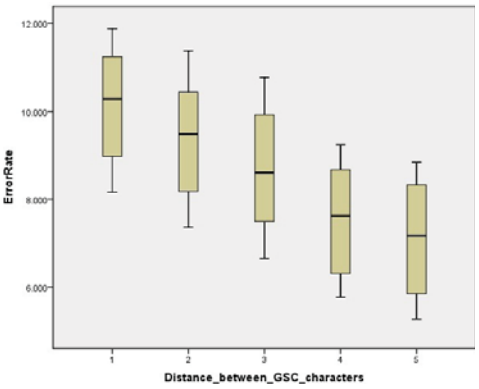
Results of the experiments are graphically shown in Fig. 3.2(b) where space between keys and corresponding typing error rate are represented through x- and y-axis, respectively. It has been observed from the experimental results that typing error rate is affected by space between keys, and it is less when the keyboard interface has 2 mm blank space between all adjacent keys. We also perform *ANOVA* test on typing error rate for different space between keys. The analysis concludes that typing error rate during text entry for different space between keys are significantly different. The observed $F_{(3,104)}$ from experimental data is 10.719 with $p < 0.05$.



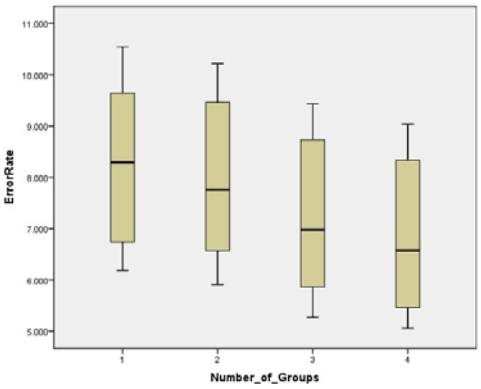
(a) Typing error rate for different key sizes



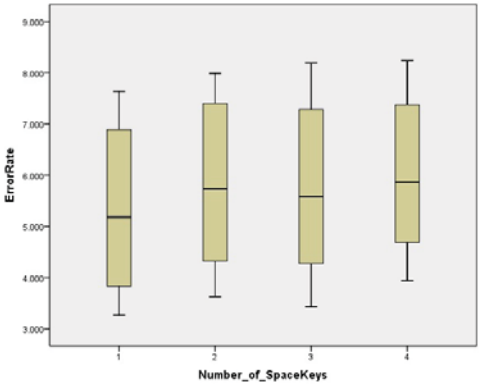
(b) Typing error rate for different space between keys



(c) Typing error rate for different distances of *GSC* character pairs



(d) Effects of different key grouping on typing errors



(e) Typing error rate for multiple number of space keys

Figure 3.2: Typing error rate influence by different virtual keyboard parameters

Distance between *GSC* Pairs: In the context of Bengali language, there are many characters which look alike, that is, *graphically similar character* (*GSC*). For example, characters ব [ba] and র [ra] in Bengali are very similar. Many a times it is confusing to distinguish between these two *GSC* characters and hence it results in typing errors in text composition. A set of graphically similar characters are shown in Table 3.4.

Table 3.4: Bengali GSC

G.S.C	Base Shape	G.S.C	Base Shape
অ [a], আ [A], ত [ta]	ত [ta]	ঢ [Dha], ট [Ta], চ [cha], ছ [chha], ঢ় [.Dha]	ঢ [Dha]
জ [ja], ড [Da], ড় [.DA], উ [u], ঊ [U]	ড [Da]	ফ [pha], য [ya], য় [Ya], ষ [Sha]	য [ya]
এ [e], ঐ [ai], ঞ [n^a]	এ [e]	স [sa], ম [ma]	স [sa]
ও [o], ঔ [au]	ও [o]	ণ [Na], ন [na]	ণ [Na]
ক [ka], ধ [dha], ব [ba], র [ra], ঝ [jha], ঋ [R^i]	ব [ba]	খ [kha], থ [tha]	থ [tha]

To quantify the graphical similarities among characters, we take characters in standardized '*Vrinda*' Bengali font and place them into a 120×120 matrix grid structures. Then for each character set, we calculate the MAD (Mean Absolute Difference) values of all sub blocks in the grid using Eqn. 3.2, where $f(a, b, i)$ denote the pixel value of a sub block at spatial coordinate (a, b) in i^{th} frame. N denotes the sub block size and $(d1, d2)$ denotes the motion or displacement of a sub block.

$$MAD = \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} \frac{|f(a, b, i) - f(a + d1, b + d2, i + 1)|}{N^2} \tag{3.2}$$

We have calculated MAD value for each character mentioned in Table 3.4. Then we calculate the modular difference between different character pairs and take those pairs which are having moderately less MAD value (MAD value difference within 250). The grid structures for characters ব [ba] and র [ra] are shown in Fig. 3.3. The selected *GSC* pairs and MAD value differences between them are displayed in Table 3.5.

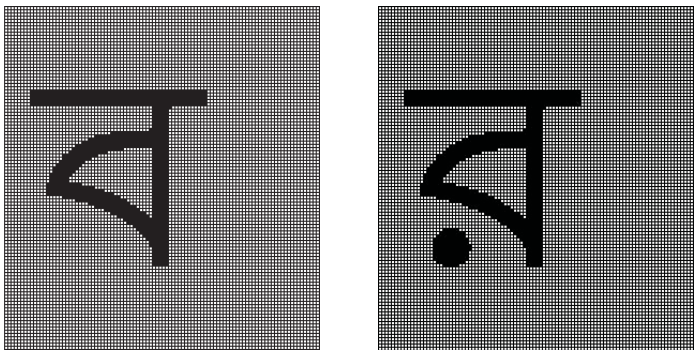


Figure 3.3: The character in grid structure

Table 3.5: Bengali Character MAD Differences

GSC set	MAD value	GSC set	MAD value
য [ya], ষ [Sha]	61	ব [ba], র [ra]	115
য [ya], য় [Ya]	85	ষ [Sha], য় [Ya]	146
ঢ [Dha], ঢ় [.Dha]	86	ট [ta], ঢ [Dha]	231
ড [Da], ড় [.Da]	89	খ [kha], থ [tha]	243

We analyze typing errors influenced by distance between *GSC* pairs. In a particular experiment, we choose a virtual keyboard maintaining specific distance between any two *GSC* pairs keeping all other parameters fixed. We consider distance between any two adjacent keys as 1 unit. In this scenario, distance of *GSC* pairs are varied from 1 to 5 unit with an increment of 1 unit. Users were requested to type the text corpus using those virtual keyboards.

The analysis of log file is shown graphically in Fig. 3.2(c) where distance between *GSC* pairs are represented through x-axis and corresponding typing error rate is shown in y-axis. The experimental results show that typing error rate almost linearly decreases with increase in distance of *GSC* pairs. It is also observed that typing error rate becomes minimum while distance is maximum (5 unit). *ANOVA* test proves that, there is a significant difference between typing error rates for different distance between *GSC* pairs ($p < 0.05$). The observed value of $F_{(4,55)}$ is 11.049.

Key Grouping: In the context of virtual keyboard, grouping is referred to as alphabetical grouping of the keys. To study the impact of key grouping on the typing error rate, we have reformed the keyboard layouts by maintaining same

type of keys (i.e. consonants, vowels, punctuations, matras) in a group with an allowance of maximum 4 groups. Here, number of groups 1, 2, 3, 4, respectively stand for no group, all character placed in a same area of the keyboard; consonants and vowels in a group and punctuations, matras in other groups; consonants, vowels in separate groups and punctuations, matras in other groups; consonants, vowels, punctuations, matras all are separate groups.

The experimental results are represented graphically in Fig. 3.2(d). It establishes that meaningful alphabetical key grouping plays vital role in typing error rate. Typing error rate linearly decreases with the number of groups. Further, *ANOVA* test also reveals the fact that typing error rate is not equal for different number of alphabetical groups. There is a significant difference between the typing error rate on different number of key groups ($F_{(3,140)} = 8.241$ where $p < 0.05$).

Number of Space Keys: Virtual keyboard like *OPTI*, *FITALY* etc. contains multiple space keys in different locations of the interface to increase the text entry rate. Here, we investigate the impact of multiple space keys on typing error rate. The experiments are performed by adding the alternate space keys in different location keeping all other parameters fixed. The number of space keys varies from 1 to 4 and placed them in different positions in the keyboard (center, first and last row). Users were requested to type the text corpus using those virtual keyboards.

Results of the experiments are graphically shown in Fig. 3.2(e) where number of space keys and corresponding typing error rate are represented through x- and y-axis, respectively. The experimental results reveal that typing error rate minutely changes while the altering the number of space keys. Further, ANOVA test also concludes that, typing error rates for different numbers of space keys are not significantly different. The observed $F_{(3,116)}$ from experimental data is 1.341 with $p < 0.05$.

3.3.7 Summary of Observations

The experimental results convey that the typing error rate varies significantly with variation of key size. The experimental outcomes of all users show that typing error rate is quite high for small key size (like $36mm^2$). The typing error rate highly decreases with increasing key size up to a moderate value ($64mm^2$). But when the

key size is high ($81mm^2$), the typing error rate is slightly increased. This concludes that user get accustomed with moderate size of keys. Similarly, the space between keys also significantly contributes to the occurrence of typing errors. It is less when the keyboard interface has $2mm$ blank space between all adjacent keys. Analysis of log file reveals that typing error rate is significantly influenced by distance between *GSC* pairs. Typing error rate almost linearly decreases with the increment of distance between *GSC* pairs. We have also found that meaningful alphabetical key grouping plays vital role in occurrence of typing errors. It also linearly decreases with number of groups. In contradiction, typing error rate is almost constant irrespective of the number of space keys. The statistical analysis (*ANOVA*) of our observed data for different parameters are summarized in Table 3.6. Here, DF_n and DF_d represent degrees of freedom numerator and degrees of freedom denominator, respectively.

Table 3.6: Summary of statistical analysis for different parameters

Design Parameter	DF_n	DF_d	F value	Significant ($p < 0.05$)
Key size	4	175	16.058	Yes
Space between keys	3	104	10.719	Yes
Distance between *GSC* pairs	4	55	11.049	Yes
Key grouping	3	140	8.241	Yes
Number of space keys	3	116	1.872	No

Table 3.6 establishes the fact that there are 4 parameters, which influenced typing error rate significantly. The parameters are:

- Key size
- Space between keys
- Distance between *GSC* pairs
- Key grouping

3.4 Summary

In this chapter, the virtual keyboard design parameters which significantly influence the user typing errors during text composition task have been identified. To accomplish the task, first we have listed typing error influencing design parameters. Then, we have performed several experiments on those identified parameters with users having different level of expertise. The experimental study concludes that typing error rate varies with variation of design parameters. It shows that typing error rate of all users are quite high for small key size (like $36mm^2$). The typing error rate highly decreases with increasing key size up to a moderate value ($64mm^2$). But when the key size is high ($81mm^2$), the typing error rate is slightly increased. Similarly, typing error rate is less when the keyboard interface has $2mm$ blank space between all adjacent keys. Further, statistical analysis (*ANOVA* test) on those experimental results has been carried out, which establishes that there are four virtual keyboard design parameters which play the significant role on user typing errors during text entry. The parameters are; *key size*, *space between keys*, *distance between GSC pairs* and *key grouping*. Outcome of this chapter would be helpful to develop a computational model to predict typing error rate based on virtual keyboard design parameters, which is presented in the next chapter.

Chapter 4

Modeling of Typing Error Rate

In the previous chapter, we have discussed about the identification of virtual keyboard design parameters, which put significant influence on occurrence of user typing error. In this chapter, we develop a predictive error model based on those identified parameters. This model is capable of predicting typing error rate for a virtual keyboard instance with different combination of design parameters' values.

In order to develop the typing error predicting model, we have collected the information regarding typing errors during text composition for different combination of parameters' values. The experimental setup, interfaces used and experimental procedure we have followed are already discussed in Chapter 3. Using these experimental data, a model has been developed using Artificial Neural Network-Genetic Algorithm (ANN-GA) hybrid approach. Here, values of virtual keyboard design parameters are input vectors and corresponding error rate is the output vector for the model. To prove the efficacy of our proposed model, similar model using ANN approach is also developed. Experimental results substantiate that our proposed ANN-GA model is superior than ANN model predicted values both for in-domain and out-domain data.

This chapter consists of four sections. Basic of Artificial Neural Network (ANN) is discussed in Section 4.1. Theoretical background on Genetic Algorithm (GA) is given in Section 4.2. Modeling typing error rate using ANN and ANN-GA hybrid, both are discussed in Section 4.3. Section 4.4 describes validation of our proposed model both for in-domain and out-domain data. Finally, Section 4.5 contains the summary of the chapter contents.

4.1 Artificial Neural Network (ANN)

Artificial neural network (ANN) is a mathematical tool designed by mimicking the working principle of the human brain information processing to simulate any complex and not properly defined problem. The neural network research is initiated with the pioneering work done by McCulloch and Pitts [68]. However, Rosenblatt's perceptron convergence theorem [69] and work of Minsky et al. [70] shows the limitations of a simple perceptron. The resultant of those works dampened the research interest on ANN for almost 20 years. Research interest on this topic again gained momentum since early 1980, by Hopfield's energy approach [71] and the back-propagation learning algorithm for multilayer perceptrons (multilayer feedforward networks) first proposed by Werbos [72], and then popularized by Rumelhart et al [73]. Since past two decades it is widely used in many applications like: forecasting/prediction [74,75], pattern classification [76], clustering [77], function approximation [78] in engineering.

Different types of ANNs exist. In our present work, here, we use feed forward multi-layer perceptron (MLP). In this network, the input data forward input layer to output layer through hidden layer(s) without any feedback. Moreover, the architecture and parameter of the network are generally determined by trial-and-error method [75].

Fig. 4.1 shows a typical three layer feed forward MLP network used for prediction purposes. Here, each neuron is operated by taking the sum of its connecting weighted inputs and bias value and passing the results through a transfer function (hard limit, linear and sigmoid function etc.). Output of each hidden layer's neuron and output layer's neuron are calculated by Eqn. 4.1 and Eqn. 4.2, respectively. Output of j^{th} hidden neuron and output layer neuron are represented by X_j and $\bar{Y}$ respectively. x_i is the value of i^{th} neuron in input layer. w_{ij}, w_{jk} are the connection weights between i^{th} neuron of input layer to j^{th} neuron of hidden layer and j^{th} neuron of hidden neuron to k^{th} neuron of output layer. w_{j0} and w_{k0} are the biases for j^{th} and k^{th} neuron respectively. Number of neurons for input, hidden layers are n and m, respectively.

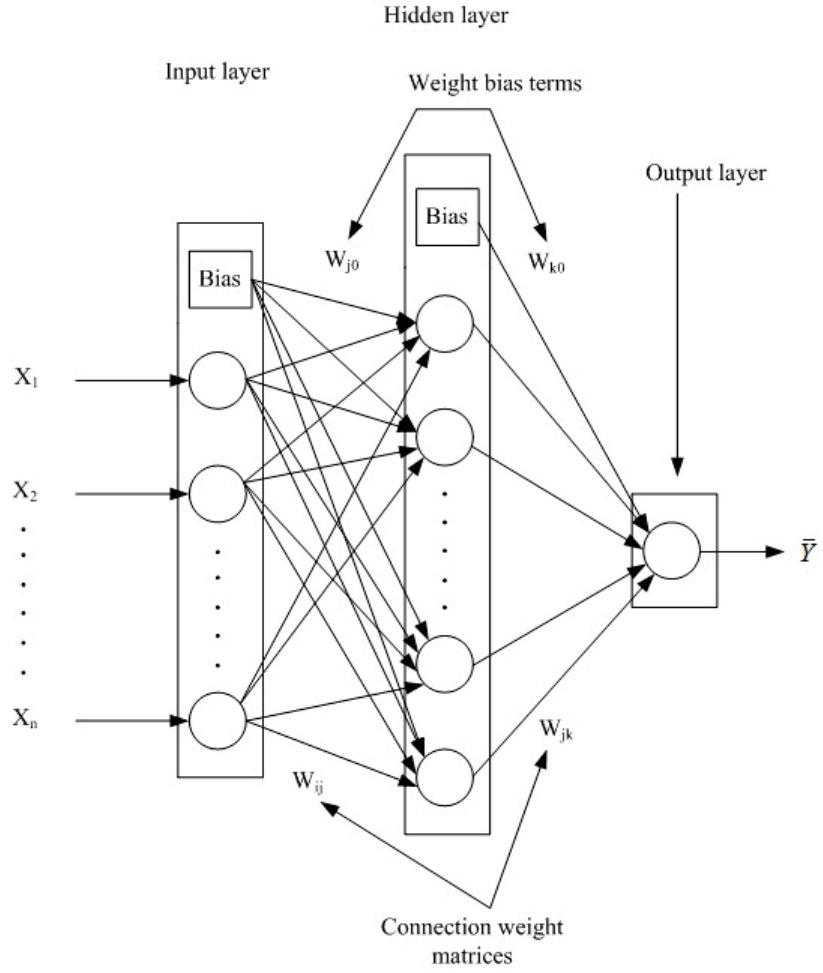


Figure 4.1: Three layer feed forward MLP network

$$X_j = f(\sum_{i=1}^{n} x_i w_{ij} + w_{j0}) \tag{4.1}$$

$$\bar{Y} = f(\sum_{j=1}^{m} X_j w_{jk} + w_{k0}) \tag{4.2}$$

Here, we use gradient descent back propagation learning algorithm [79]. At the beginning of the back propagation learning, all connection weights of the network are initialized with small random values. The learning set, consists of sufficient data with input and desired output pattern for proper learning of the network. The network output value ($\bar{Y}$) is compared with the desired output (Y) of the

training set by an error function E as defined in Eqn. 4.3, where the number of data in the training data set is N.

$$E = \frac{1}{N}\sum_{t=1}^{N}(Y_t - \bar{Y_t}) \tag{4.3}$$

The main goal of the training procedure is to minimize the error function (E) by adjusting connecting weights and biases. The updated or adjusted weights and biases are computed by the Eqn. 4.4 and Eqn. 4.5, respectively, where w_{ij}^n and w_{ij}^p represent the new (updated) and the previous connected weight from i^{th} to j^{th} neuron. Similarly, w_{j0}^n and w_{j0}^p are the new and previous bias value for j^{th} neuron.

$$w_{ij}^n = w_{ij}^p + \Delta w_{ij} \tag{4.4}$$

$$w_{j0}^n = w_{j0}^p + \Delta w_{j0} \tag{4.5}$$

The changes in connecting weight and bias are represented by Δw_{ij} and Δw_{j0} respectively, and also determined by Eqn. 4.6, and Eqn. 4.7, where $\frac{\partial E}{\partial w_{ij}^p}$ and $\frac{\partial E}{\partial w_{j0}^p}$ are the error gradient with reference to the weight and bias respectively. η represents the learning rate and it value lying between 0.0 to 1.0. If the learning rate is very small, the training algorithm takes relatively long time to converge. In contrast, if it is very high, it will converge relatively fast but network may be unstable. The learning rate is decided through trial and error for proper combination of fast converge and relatively low processing time.

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}^p} \tag{4.6}$$

$$\Delta w_{j0} = -\eta\frac{\partial E}{\partial w_{j0}^p} \tag{4.7}$$

Moreover, momentum constant (μ) is used to make the network stable at high learning rate. The change in connecting weight in a particular iteration (t) is determined through the sum of error gradient multiplied by learning rate, and momentum constant multiplied by connection weight change in previous iteration (t-1), as shown in Eqn. 4.8. Moreover, momentum value is also selected by trial and error and it lies between 0.0 to 1.0.

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w_{ij}} + \mu \Delta w(t-1) \tag{4.8}$$

The main goal of the training phase is to minimize the error function (E) by adjusting connection weights and biases using gradient descent back propagation algorithm. After training a sufficient numbers of the epoch, it is hoped that the network learns or generalizes and opens up the relationships between input and output vectors by optimizing the error function.

4.2 Genetic Algorithm (GA)

Genetic algorithm (GA) is a powerful optimization or search technique designed by mimicking the principle of biological evolution and selection, and is introduced by John Holland [80]. It is a heuristic based approach, particularly applicable to problems which have a large set of solutions, non-linear and possibly discrete in nature [81, 82]. Due to unavailability of exact mathematical formulation, it does not guarantee to reach the optimality. However, it is likely to give the near to optimal solution. Other major advantages of genetic algorithm includes: parallelism, liability, robustness, easily finding the global optimum, handles noisy functions, no knowledge requirement about the response surface etc. [81].

The algorithm manipulates a population of potential solutions to search the optimal solution based on Darwin's 'survival of the fittest' strategy of biological genetic system. Initial population is generated with predefined number of individuals or chromosomes (N). The individuals are survived and selected for next generation according to objective/fitness function. At each generation, GA selects individuals from the current population to produce children or offspring by performing crossover with crossover probability (p_c) for next generation. To bring more diversity in the population, newly created offspring are randomly mutated with mutation probability (p_m). The newly generated offspring are then inserted into the new population. After that, the crossover and mutation operations are iteratively executed until the size of new population is equal to the previously predefined number of individuals (N). Then, the current (parent) population is replaced with newly created offspring. After that, next generation is started with those offspring. Optimal solution comes out by iteratively executing this procedure

over sufficient numbers of generations till it satisfies the termination condition or the problem gets converged. To provide more clear understanding, the outline of GA is illustrated in Fig. 4.2.

4.3 Modeling Typing Error Rate

Typing error model has been carried out by two different approaches namely ANN and ANN-GA hybrid approach. In both the approaches, we use the same experimental data. In our work, the previously identified virtual keyboard design parameters are input vectors and corresponding error rate are the output vectors. In depth description about these models are presented in this section.

4.3.1 Input-Output Vectors for Typing Error Prediction Model

To model typing error rate based on those identified design parameters, we need to collect the information regarding typing error during text composition for different combination of parameter values. So, we conduct several experiments with participants on different variation of parameter values. The experimental setup and procedure, participants and virtual keyboard interfaces used are same as discussed in Chapter 3. We collect total 1680 experimental data samples with 210 different combinations of design parameters values. Note that different values of keyboard design parameters are the input vectors and corresponding typing error rate are the output vectors in our model. So, total number of input vectors (key size, space between keys, distance between eight *GSC* pairs, grouping) are 11 and output vector is one (typing error rate).

Data Normalization: Before starting of training phase, data need to be normalized over a specific range. This is required to keep all entries in the same range and the range must be limited by neurons' transfer functions. We normalize entire data using min-max normalization over the range $[-1, 1]$ as shown in Eqn. 4.9. The range is decided based on the nature of the sigmoid function of ANN, which is used in both the models as a transfer function.

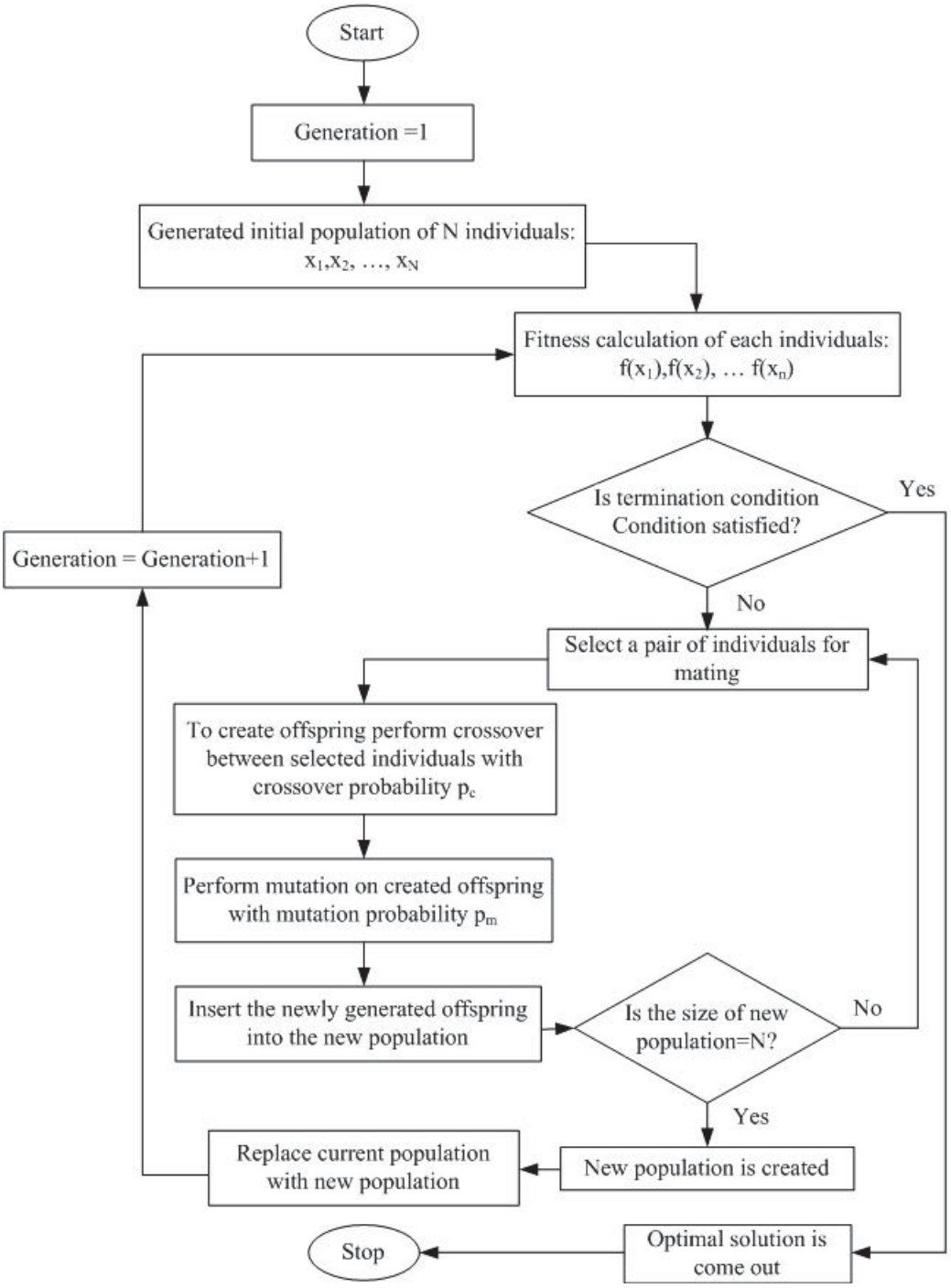


Figure 4.2: Flowchart of GA

$$X' = \frac{(X'_{max} - X'_{min})}{X_{max} - X_{min}}(X - X_{min}) + X'_{min} \tag{4.9}$$

Here, X' is the normalized value of variable X. X_{max} and X_{min} are the expected maximum and minimum values of concerned variable. X'_{max} and X'_{min} specify the desired values of the range for the transformed variable.

4.3.2 Statistical Performance Metrics

In order to compare and quantitatively determine the accuracy of model, four statistical performance metrics are employed, which are: (1) Mean Absolute Percentage Error ($MAPE$), (2) Mean Square Error (MSE), (3) Standard Deviation Error (SDE) and (4) Correlation Coefficient (R). The $MAPE$, MSE, SDE and R are calculated using the below mentioned Eqn. 4.10 − 4.13, respectively.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i - \bar{Y_i}\right|}{Y_i} \times 100 \tag{4.10}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y_i})^2 \tag{4.11}$$

$$SDE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y_m})^2} \tag{4.12}$$

$$R = \frac{\sum_{i=1}^{n}(Y_i - Y_m)(\bar{Y_i} - \bar{Y_m})}{\sqrt{\sum_{i=1}^{n}(Y_i - Y_m)^2 \sum_{i=1}^{n}(\bar{Y_i} - \bar{Y_m})^2}} \tag{4.13}$$

Here, n is the total number of data sets. Y_i and $\bar{Y_i}$ are the actual and model predicted typing error rate corresponding to the i^{th} data set. Mean of actual and model predicted typing error rate are represented by Y_m and $\bar{Y_m}$, respectively. The smaller value for $MAPE$, MSE, SDE and larger value for R are preferable, which means that the predicting data are closer to the actual data.

4.3.3 Model Implementation

Initially, we generate the error prediction model on the basis of virtual keyboard design parameters using ANN. Here, the architecture parameters of the network (number of hidden layer(s) and neurons in hidden layer(s), transfer functions, learning rate, momentum, epochs) are determined by the trial-and-error method. To address this limitation, ANN-GA hybrid method is utilized. Here, GA is applied for searching the optimal ANN parameters value to provide more acceptable predictive results. Details, implementation and corresponding results of both ANN and ANN-GA models are discussed afterwards in this section.

ANN Model Implementation:
Several MLP networks with various network architecture parameter specifications are generated and tested on that input-output data (Section 4.3.1). The size ratio of the training validation and testing data set are 70%, 15%, and 15% respectively. Here, we create ANN with maximum two hidden layers as more than two may not give significant improvement in performance [83, 84]. The transfer function for input layer to first hidden layer is chosen as Purelin (P), for all other layers, the selected function is Tan-Sigmoid (TS). Before the training, all data are normalized over the range $[-1, 1]$. The back propagation algorithm is used to adjust the connection weights. Network structure and parameters are decided by the trial-and-error method. The parameters for the best network are found out by the trial-and-error approach as shown in Table 4.1. The model is developed and simulated using MATLAB tool.

ANN-GA Model Implementation:
In ANN, it is mathematically impossible to determine the value of different network parameters [75, 85] and in this context, these are decided by the trial-and-error approach. This is one of the foremost limitations of ANN model. To address this limitation, ANN-GA hybrid approach is employed, which is developed by the combination of Genetic Algorithm (GA) with ANN. Here, GA is used for searching the optimal network parameters values for the ANN to provide optimal predictive value, instead of trial-and-error method in traditional ANN forecasting approach. Optimal solution or optimal parameters values from the solution space come out by

Table 4.1: Parameters of the network created by ANN model

Parameter	Value
Number of neurons (nodes) in Input layer	11
Number of neurons (nodes) in Hidden layer 1	11
Number of neurons (nodes) in Hidden layer 2	24
Number of neurons (nodes) in Output layer	1
Transfer Function (Tf)	P-TS-TS
Learning Rate (Lr)	0.003
Momentum (Mn)	0.003
Epochs	1000

iteratively applying GA in consecutive generations. The major steps or strategies involved in ANN-GA approach is elaborated in this subsection.

Chromosome Representation: In GA, each individual of a population is described through chromosome representation or encoding. Here, we use binary encoding scheme to represent the chromosome. The network parameters are encoded in a 25 bit binary chromosome. In our study, we use ANN having the maximum of two hidden layers. The first and second hidden layers are represented by first and second consecutive five bits respectively of the chromosome. Therefore, the number of neurons in each hidden layer is varied within the range of 0 to 31 in the integer number domain. The transfer functions for one layer to subsequent layer are represented by next successive 6 bits. Here, initial 2 bits of those successive 6 bits denote the transfer function for input to first hidden layer. Similarly, next 2 and last 2 bits represent the transfer function for first hidden layer to second hidden layer and second hidden layer to output layer, respectively. Moreover, we use four different types of transfer functions namely, Hard-limit(HL), Purelin (P), Log-Sigmoid (LS), Tan-Sigmoid(TS), which are symbolized by 00, 01, 10, 11, respectively. Learning rate, momentum rate and number of epochs are represented by 3 consecutive bits among rest of 9 bits. Here, learning rate and momentum rate are varied from 0.0005 to 0.0040, where 000, 001 are denoted by 0.0005 and 0.0010 respectively and so on. Similarly, number of epochs varies from 500 to 1200, where 000, 011 symbolize for 500 and 600, respectively and so on. Detail encoding for those are shown in Table 4.2.

Table 4.2: Encoding for Learning Rate, Momentum Rate and Epochs

Encoding	Learning Rate	Momentum Rate	Epochs
000	0.0005	0.0005	500
001	0.0010	0.0010	600
010	0.0015	0.0015	700
011	0.0020	0.0020	800
100	0.0025	0.0025	900
101	0.0030	0.0030	1000
110	0.0035	0.0035	1100
111	0.0040	0.0040	1200

Encoding scheme is described by an example to provide more clear understanding. If the encoding string of a chromosome is represented by "010011010001100010101011100" as shown in Fig. 4.3, then corresponding ANN structure and parameters are described in Table 4.3.

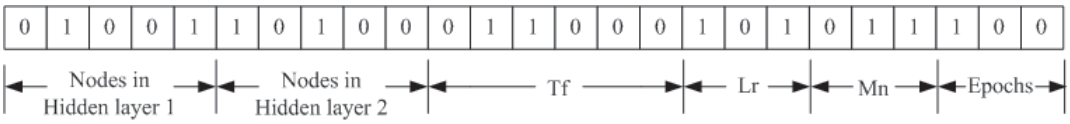


Figure 4.3: Chromosome encoding for ANN structure and training parameters

Table 4.3: ANN structure according to the chromosome shown in Fig. 4.3

Parameter	Value
Number of nodes in Hidden layer 1	9
Number of nodes in Hidden layer 2	20
Transfer Function (Tf)	P-LS-HL
Learning Rate (Lr)	0.003
Momentum (Mn)	0.002
Epochs	900

Selection: Individuals are selected according to their fitness or objective function values, shown in Eqn. 4.3. Here, fitness function is calculated after decoding the encoded chromosome string. After calculating fitness function for

each individual, we select the chromosomes for crossover using the 'rank-based selection' [86] method as shown Eqn. 4.14. In this method the chromosomes are arranged in ascending order of their fitness and chromosome having the lowest fitness is assigned rank 1 and other chromosomes are ranked accordingly. Then a proportionate selection scheme based on the assigned rank is followed as shown in Eqn. 4.14, where r_i indicates the rank of i^{th} chromosome and n is the size of the population. Selection probability of each i^{th} individual is represented by p_i.

$$p_i = \frac{r_i}{\sum_{i=1}^{n} r_i} \times 100 \tag{4.14}$$

Genetic operators (Crossover and Mutation): After selecting the individuals, new generation is created by crossover operation. Here, we use two-point crossover, first and second crossover points are randomly selected from the range of 3 to 7 and 16 to 20 from the chromosome to create new offspring for the new generation. To bring more diversity in the population, newly created offspring are randomly mutated. In this case, each bit of a chromosome is mutated 0 to 1 and vice-verse with mutation probability μ_p.

ANN-GA Model: The outline of the ANN-GA optimization approach is shown in Fig. 4.4. Initially the experimental data sets are collected and normalized in a specific range $[-1, 1]$. Then it is subdivided into training, validation and testing data set with the ratio of 70%, 15%, 15% correspondingly. More Detailed specification about data collection and data normalization is given in Section 4.3.1. Individuals or chromosomes are created randomly in binary encoded form. Each chromosome is decoded and corresponding structure is simulated and trained by training data. Then, the network is validated, tested and fitness values are calculated using fitness function (Eqn. 4.3). We then iteratively execute this procedure up to predefined number of maximum individuals (20) to create new individuals and calculate their fitness values. After that, selection, crossover and mutation procedures are used to reproduce a new generation. After executing 50 generations, the optimal network parameter values for the typing error modeling come out. The value of the GA and ANN parameters for the ANN-GA model are shown in the Table 4.4 and 4.5. This model is also developed and simulated using MATLAB.

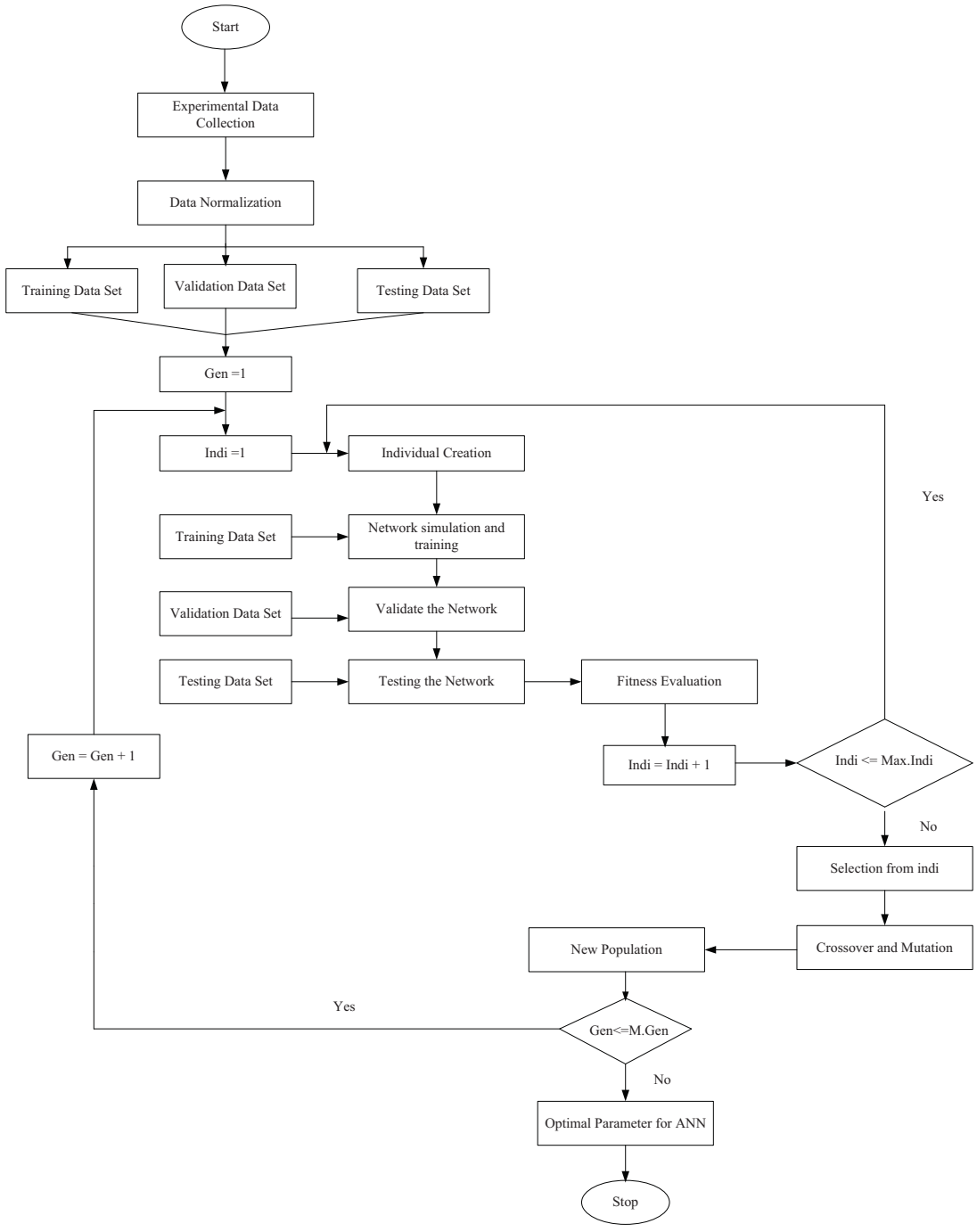


Figure 4.4: Flowchart of ANN-GA model

Table 4.4: GA parameters for ANN-GA model

Parameter	Value
Encoding	Binary
Population size	20
Generation	50
Selection	Rank-based
Crossover	Two point
Mutation	Uniform
P_c	0.9
P_m	0.1

Table 4.5: Parameters of the network created by ANN-GA model

Parameter	Value
Number of neurons (nodes) in Input layer	11
Number of neurons (nodes) in Hidden layer 1	26
Number of neurons (nodes) in Hidden layer 2	20
Number of neurons (nodes) in Output layer	1
Transfer Function (Tf)	P-LS-TS
Learning Rate (Lr)	0.0035
Momentum (Mn)	0.002
Epochs	1100

4.4 Model Validation

In this section, initially we compare the predicted results (typing error rate) of our proposed ANN-GA model and ANN model both with the experimental data (in-domain data). Then we also perform another empirical study with different interfaces and participants (out-domain data). In this study, performances predicted by the proposed models for a set of design parameters are compared with the observed user typing error rate. Detail results of our proposed model for both in-domain and out-domain data are reported in the following subsections.

4.4.1 Results for In-domain Data

Here, the testing data are taken from the same source on which the model is trained. Actual user typing error rate for different combination of design parameters and corresponding typing error rate is also calculated from the models. The performance comparison of the ANN-GA with the ANN prediction model is shown in Table 4.6. The values of $MAPE$, MSE, SDE metrics are smaller and value of R is larger for ANN-GA model compared to ANN model. It concludes that ANN-GA model performs better than ANN model based on the statistical performance metrics.

Table 4.6: Result comparison of ANN and ANN-GA models for in-domain data

Model	Performance			
	MAPE	MSE	SDE	R
ANN	8.461	0.940	0.0179	0.890
ANN-GA	5.848	0.401	0.0097	0.939

4.4.2 Results for Out-domain Data

The accuracy and generalization capability of our proposed ANN-GA model can better be established if similar types of results are observed for out-domain data. In order to determine the efficacy of our proposed model, another separate empirical study with different interface and participants has been carried out. We have mimicked the keyboard used in Google Bengali transliteration [87] for the study. We designed the keyboard interface using Visual Studio C# 2008. The keyboard interface is shown in Fig. 4.5.

In our experiment, 9 other participants (5 males, 4 females) of expert, intermediate and novice categories are selected. We follow the same user categorization as discussed previously. We select four other text corpora from the earlier mentioned corpora sources to validate our model. Detail statistics about these selected corpora are illustrated in Table 4.7.

We observe typing error rate for different combination of design parameters from user experiments. The corresponding typing error rate for each combination is also calculated from the models. The performance comparison of ANN-GA with

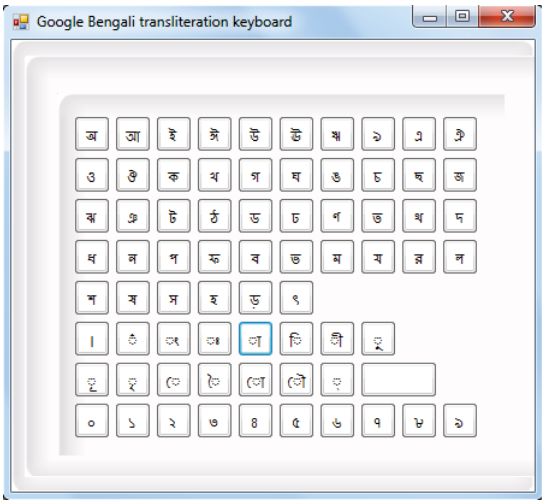


Figure 4.5: *Google Bengali transliteration* keyboard layout

Table 4.7: Selected texts in the experiments

Text Under Test	Reference Text	Word Count	Character Count			Inflexion Count
			No Space	With Space	Complex	
TB_{v1}	"Kapalkundala" by Bankim Chandra Chattopadhyay	163	984	1146	69	326
TB_{v2}	"Srikanta" by Sarat Chandra Chattopadhyay	178	846	1023	35	270
TB_{v3}	"Ashamanjababur Kukur" by Satyajit Ray	191	969	1152	36	307
TB_{v4}	"Wikipedia" Bengali	174	892	1065	40	293

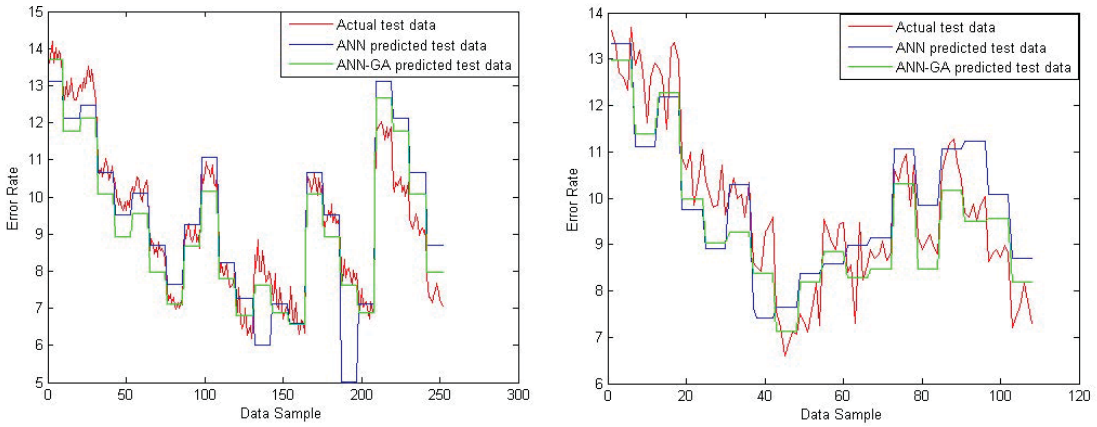
ANN model is shown in Table 4.8. Here, the values of *MAPE*, *MSE*, *SDE* metrics are smaller and value of R is larger for ANN-GA model compared to ANN model. It concludes that for each statistical metrics, ANN-GA model is superior than ANN model predicted values for out-domain data also.

Our observed user typing error rate for different combination of parameter values and corresponding predicted error rate for both ANN and ANN-GA models for both in-domain and out-domain data are graphically shown in Fig. 4.6. Results comparison for in-domain data and out-domain data are illustrated in Fig. 4.6a,

Table 4.8: Result comparison of ANN and ANN-GA models for out-domain data

Model	Performance			
	MAPE	MSE	SDE	R
ANN	9.057	0.988	0.0492	0.827
ANN-GA	6.232	0.526	0.0277	0.930

Fig. 4.6a, respectively, where x-axis represents the data sample (user typing error rate for different combinations of parameter values) and corresponding error rates are shown by y-axis. Here, actual data and respective predicted data for ANN and ANN-GA models are represented by red, blue and green line respectively. It shows that our proposed ANN-GA model outperformed than ANN model for out-domain data also.



(a) Result comparison for in-domain data (b) Result comparison for out-domain data

Figure 4.6: Comparison between predicted and actual typing error rate on both in-domain and out-domain data for both ANN and ANN-GA models

4.5 Summary

In the context of virtual keyboard interface, developing a typing error prediction model remains a serious problem. This chapter addresses the problem and proposes a model to predict typing error rate employing the ANN-GA hybrid soft computing framework. To judge the efficacy of our proposed model, similar model using ANN

approach is also developed. The models are also validated with both in-domain and out-domain data on the basis of different statistical performance metrics ($MAPE$, MSE, SDE and R). The experimental result substantiates that both for in-domain and out-domain data our proposed model perform better than ANN model for each performance metric. The model is developed for predicting typing error rate during text composition with Bengali virtual keyboard. This work, however, can be extended to other Indian languages as well as other foreign languages. The proposed model is not only useful to predict typing error rate given a virtual keyboard, but also can be applied to design an efficient and error-optimum virtual keyboards that minimizes typing error rate and maximizes text entry rate.

Chapter 5

Virtual Keyboard Design with Multi-Objective Optimization

In the previous chapters, we observe that typing error plays a significant role while composing text through virtual keyboard. Previously, researchers have concentrated on enhancing text entry rate by optimizing average mouse movement time only. However, the development of a virtual keyboard with maximum text entry rate as well as minimum typing error rate is desirable. It may be noted that occurrences of typing errors are specifically more significant during text entry through Indian language compatible virtual keyboard due to linguistic features [30]. It may also be noted that in the context of an Indian languages, a single character error may require multiple number of edit primitives to correct the error [88], affecting the text entry rate. So, designing a Bengali virtual keyboard without considering typing errors narrows down the practical usefulness of that design.

In order to account users' typing errors in the design, a predictive error model has been developed using ANN-GA hybrid approach, as described in Chapter 4. The model is capable to predict typing error rate for a given design. However, it is difficult to evaluate designs on the basis of error rate alone, as the measure does not have any text entry rate component. Hence, in order to arrive at a design methodology for virtual keyboard design, both these measures (i.e. text entry rate and error rate) have been taken into account. In this chapter, we propose an approach to design a virtual keyboard optimizing both text entry rate and

user typing error rate. To judge the effectiveness of our multi-objective design approach, we design another alternate virtual keyboard using GA for maximizing text entry rate only.

This chapter consists of five sections. The objective of our work is discussed in Section 5.1. Our proposed approach to design virtual keyboard using NSGA-II multi-objective optimization is discussed in Section 5.2. Section 5.3 describes the alternate design approach using GA. An empirical study has been carried out to show the validity of the proposed design approach. Results of the study are discussed in Section 5.4. Finally, Section 5.5 contains the summary of the chapter contents.

5.1 Objective of the Work

In this work, we propose an approach to design a Bengali virtual keyboard which achieves higher text entry rate and less typing error rate with proper combination of different virtual keyboard design parameters. Moreover, we consider the single tap virtual keyboard only, defining the following assumptions:

1. All keys are of equal size and square shape.
2. Equal amount of horizontal and vertical spaces in between all adjacent keys.
3. The characters placed in the keys are all different.
4. Each key contains one character.
5. Keyboard layout remains unchanged throughout the experiment.

5.2 Proposed Approach

We design a multi-objective optimized Bengali virtual keyboard layout based on two optimality metrics; maximize text entry rate and minimize typing error rate. Text entry rate is calculated by Fitts-digraph model [12, 24] and typing error rate is computed by our proposed error model discussed in Chapter 4.

It is concluded from Fitts-digraph model that text entry rate (TR) gets increased by decreasing distance between characters having high digram frequency.

In contrast, from the error model, it is observed that inverse of error rate ($1/ER$) is directly proportional to distance between GSC pairs, and moreover, some of those are highly frequent. Moreover, TR gets increased by decreasing space between keys (SK); in contrast, $1/ER$ gets reduced. So, the objectives are conflicting in nature. Therefore, we need to use a multi-objective optimization to optimize both of the objective functions.

5.2.1 Problem Statement

The performance measurement of a virtual keyboard (VK_{PER}) is a function of both TR and ER. For simplicity, the problem has been transformed to maximization of the both objectives, by taking $1/ER$ as one of the optimization criteria as shown in Eqn. 5.1. The TR and ER are stated in the following.

$$maximize \quad VK_{PER} = f(TR, \frac{1}{ER}) \tag{5.1}$$

Text Entry Rate (TR): The average text entry rate is calculated as word per minute (WPM), which can be expressed in terms of character per second (CPS) as shown in Eqn. 5.2, where W_{AVG} represents average word length. Further, CPS is calculated from the inverse of reaction time (RT) and mean movement time (MT_{Mean}) as shown in Eqn. 5.3.

$$TR\ in\ WPM = \frac{CPS \times 60}{W_{AVG}} \tag{5.2}$$

$$CPS = \frac{1}{RT + MT_{MEAN}} \tag{5.3}$$

According to Hick-Hyman law [89, 90], as shown in Eqn. 5.4, where a' and b' are empirically determined constants and N is the total number of keys present in the keyboard interface. So, RT depends on total number of keys present in the keyboard interface. MT_{MEAN} according to Fitts-digraph model [12, 24] is calculated by summing up movement time (MT_{ij}) between all digram multiplied by corresponding digram probability (P_{ij}), as represented in Eqn. 5.5. Here, we analyze the "Wikipedia" Bengali corpus, to compute digram probabilities of occurrences.

$$RT = a' + b' log_2 N \tag{5.4}$$

$$MT_{MEAN} = \sum_{i=1}^{N} \sum_{j=1}^{N} MT_{ij} \times P_{ij} \tag{5.5}$$

Movement time (MT_{ij}) from i^{th} key to j^{th} key is calculated from Fitts's law [54] as shown in Eqn. 5.6, where a and b are empirically determined constants, D_{ij} is the Euclidean distance between the center of both keys and W_j is width of the target key (j^{th} key). P_{ij}, the digraph probability of occurrence of j^{th} character after i^{th} character, is also calculated from Fitts-digraph model.

$$MT_{ij} = a + b log_2(\frac{D_{ij}}{W_j} + 1) \tag{5.6}$$

Error Rate (ER:) Our proposed error model described in Chapter 4 is used to relate inverse of error rate ($1/ER$) and virtual keyboard design parameters. According to it, ER is a function of design parameters namely, key size (KS), space between keys (SK), distance between GSC pairs (D_{GSC}), grouping (GR).

Solving our problem would be subject to some constrains, which are discussed afterwards. Here, we consider only square shaped keys and in the design space, key width of any j^{th} (W_j) is varied from 6 mm to 9 mm. So, KS for any j^{th} (KS_j) is also varied from 36 mm^2 to 81 mm^2. It also be noted that in a particular keyboard we maintain equal size keys in a keyboard.

We consider equal amount of horizontal and vertical space for any two adjacent keys in a particular keyboard interface. In our design space for different keyboard interface, space between any two adjacent keys namely, i^{th} key to j^{th} key is varied between 1 mm to 4 mm. Moreover, centroid distance between i^{th} key to j^{th} key (D_{ij}) is calculated by half of i^{th} and j^{th} keys width and space between them. As all keys are in equal width, so D_{ij} can be calculated by j^{th} key width and space between them. Minimum and maximum distance between i^{th} key to j^{th} key is 7 mm and 13 mm, respectively as shown in Eqn. 5.7.

$$\begin{aligned} & \frac{1}{2}W_{imin} + \frac{1}{2}W_{j_{min}} + SK_{min} \leq D_{ij} \leq \frac{1}{2}W_{imax} + \frac{1}{2}W_{j_{max}} + SK_{max} \\ = \; & W_{j_{min}} + SK_{min} \leq D_{ij} \leq W_{j_{max}} + SK_{max} \\ = \; & 7mm \leq D_{ij} \leq 13mm \end{aligned} \tag{5.7}$$

In our design space, distance between any two GSC pair is varied from 1 to 5 units. Here, we kept vowels and consonants in the same group. Punctuations are kept in a single group and inflexions ("matars") belong to another group.

5.2.2 NSGA-II-Based Optimization

There are many approaches to solve a multi-objective problem using GA like Vector Evaluated GA (VEGA) [91], Multi-objective Genetic Algorithm (MOGA) [92], Niched Pareto Genetic Algorithm (NPGA) [93], Weight-based Genetic Algorithm (WBGA) [94], Non-dominated Sorting Genetic Algorithm (NSGA) [95], improved NSGA (NSGA-II) [96], Strength Pareto Evolutionary Algorithm (SPEA) [97], Pareto-Archived Evolution Strategy (PAES) [98] etc. In our work, we use NSGA-II algorithm, which is treated as a fast (computational complexity $O(MN^2)$) and elitist multi-objective GA proposed by Dev et al. [96]. It has been reported that, compared to other elitist multi-objective GAs (PAES, SPEA etc.), it has better diversity preservation [96, 99]. So, it can compete with them in the context of converging in the true Pareto-optimal front. Moreover, traditional approaches (MOGA, NSGA) use the concept of fitness sharing by niching. The main problem with sharing is that it requires the specification of a sharing parameter and performance of the sharing function method depends largely on the chosen sharing parameter. NSGA-II replaces the sharing function approach with a crowded-comparison approach that eliminates difficulties. Detailed procedure of NSGA-II is explained below.

NSGA-II algorithm follows the usual crossover and mutation operators, but selection operator work differently from simple GA [96]. Selection is done with the help of fast non-dominated sorting procedure, crowded distance estimation procedure and crowded-comparison operator [96]. Initial parent population P of size N is randomly created. The population is sorted based on the non-domination.

In order to identify the non-dominated solutions, each solution is compared with every other solution.

Fast non-dominated sorting procedure: In the first sorting [96], each non-dominated solution ($N1$) is assigned rank or fitness as 1. Then, the remaining $N - N1$ solutions are again sorted and rank 2 is assigned to each non-dominated solution. Iteratively we need to execute this process until all the solutions are ranked. Here, each solution is assigned rank according to its non-domination level. Best solutions belong to rank 1 and next-best as rank 2, and so on. Further, any solution of a particular rank is not better with respect to any other solutions belong to the same rank.

Crowded distance estimation procedure: After ranking, crowding distance for each solution of all non-domination levels are calculated to get density estimation of solutions. For any particular solution, it is calculated by measuring the average distance of two nearest point (solution) on either side along each of the objectives. To compute the crowding distance, initially, the entire population of particular non-dominated set are sorted in ascending order of magnitude according to each objective function value. Then the boundary solutions for each objective function is assigned an infinity value. After that, the rest of the intermediate solutions are assigned a distance value equal to the absolute normalized difference in the objective function value of two adjacent solutions. This calculation is done for all objective functions. Finally, the overall crowding distance value is computed as the sum of individual distance value corresponding to each objective.

Crowded-comparison operator: In order to select the solutions toward a uniformly spread-out Pareto-optimal front, crowed comparison operator is used. Here, if two solutions belong to different non-domination ranks, then the solution having lower rank is selected. Otherwise, if both solutions are in the same front, then the solution with lesser crowded distance is preferably selected.

5.2.3 Virtual Keyboard Design using NSGA-II

We consider the virtual keyboard consisting of three different panels namely character, inflexion and punctuation as shown in Fig. 5.1. All characters including vowels, consonants and two punctuation characters ("space" and ।) are resided

in character panel. All inflexions ("matras") and punctuation, modifiers (like: backspace (পূর্বস্থান [puub^rsthaana]), delete (পরিষ্কার [pariShkaar]) key) are assigned alphabetically in inflexion and punctuation panel, respectively. Moreover, "matras" are appeared in inflexion panel dynamically that is only visible after a consonant key is tapped, as those are unable to appear independently without any consonant. Here, we make modifier keys double in size as compared to other keys, to search those efficiently during text typing. Moreover, to balance the horizontal and vertical mouse pointer movement during text composition, we make the character panel as 7×7 layout. We also place the "Enter" key (পরবর্তী [parabatI^r]) in between character panel and punctuation panel.

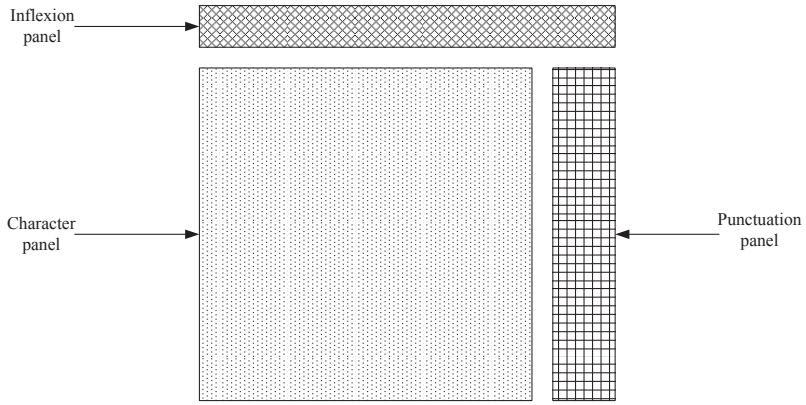


Figure 5.1: Character panel architecture

Key arrangement and other virtual keyboard design parameters of character panel are optimized by employing NSGA-II for maximizing text entry rate along with minimizing typing error rate. Each step of our virtual keyboard design approach using NSGA-II is illustrated in Fig. 5.2.

Here, we randomly generate 20 chromosomes each of 53 bits in initial population. Initial 49 bits of a chromosome use real encoding to represent the key arrangement, where first 7 bits denote the first row, second 7 bits for second and so on. Distance between *GSC* pairs are calculated from this arrangement. Similarly, last 4 bits of a chromosome are encoded in binary form, where first 2 bits and last 2 bits represent key size and space between keys respectively. Encoding for those are shown in Table 5.1.

Single point substring crossover technique [100] with randomly selected

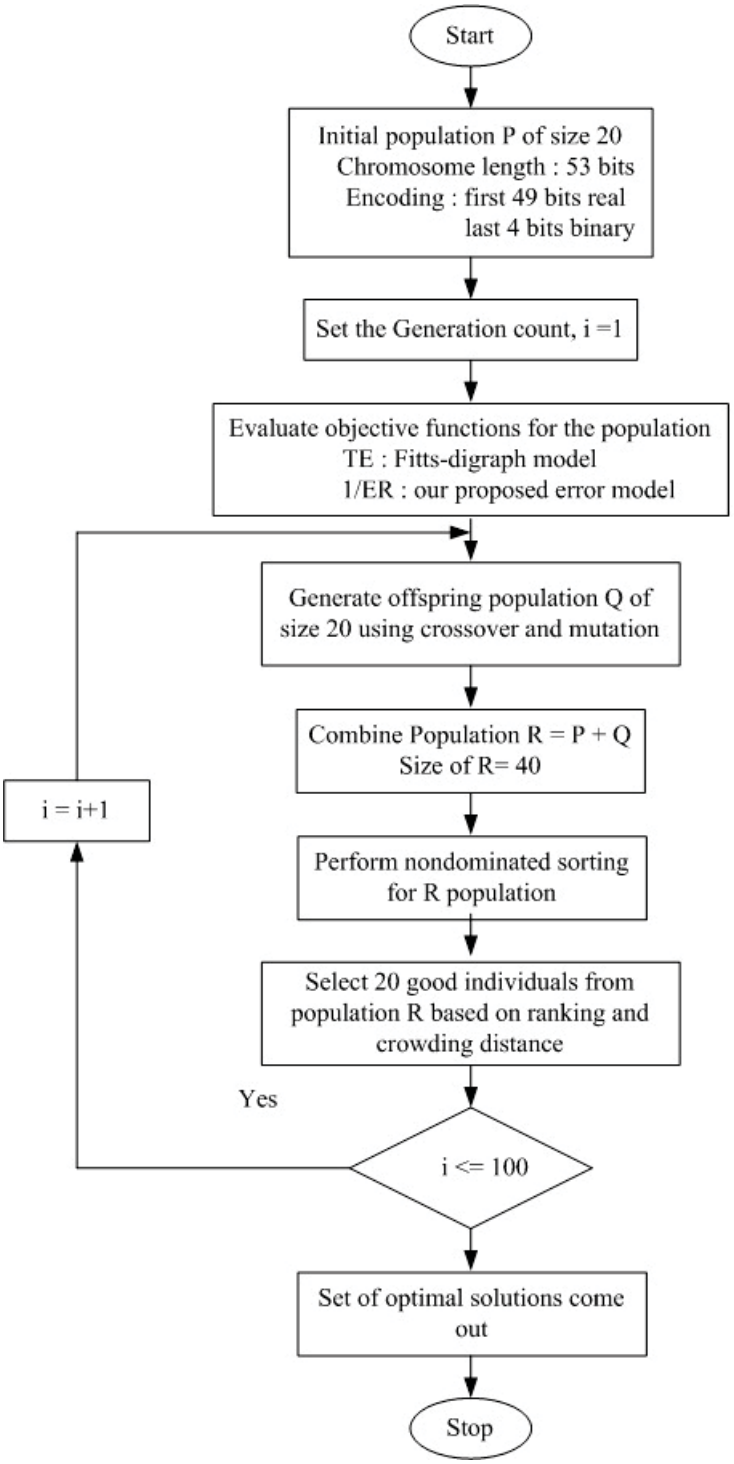


Figure 5.2: Flowchart of virtual keyboard design using NSGA-II

Table 5.1: Encoding for key size and space between keys

Encoding	Key size (in mm^2)	Space between keys (in mm)
00	36	1
01	49	2
10	64	3
11	81	4

crossover point in between 20 to 30 is applied in the real coded portion. Similarly, for binary portion, we also use the single point binary crossover technique [101]. Our crossover approach for both real and binary portion of a chromosome is illustrated in Fig. 5.3.

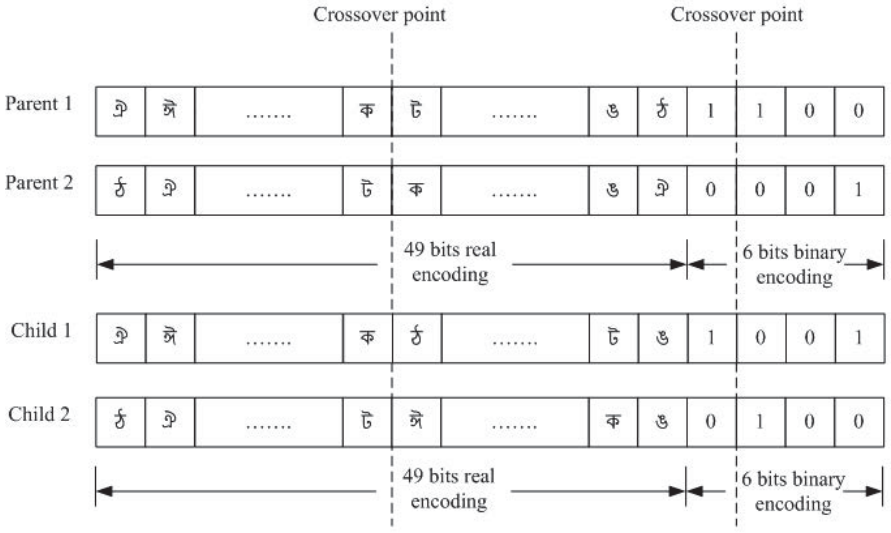


Figure 5.3: Crossover operation for both real and binary portion of chromosome

For mutation of real coded portion, two points randomly select and swap each other. Similarly, for binary portion each bit of the chromosome is mutated 0 to 1 and vice-verse with mutation probability 0.1. The objective functions are calculated from Fitts-digraph model [12, 24] and our proposed error model discussed in Chapter 4. 20 individuals are selected by ranking and crowding distance comparison operators as mentioned in NSGA-II algorithm [96]. After

that, next generation is started with those individuals. We execute the above mention procedure iteratively up to 100 generations to find the set of optimal solutions. Pareto-optimal set of 20 non-dominated solutions are obtained at the end of 100 generations as shown in Fig. 5.4. It has been implemented in MATLAB tool in Windows 7 environment in a PC having Pentium Core2Duo processor with 2.4 GHz clock speed.

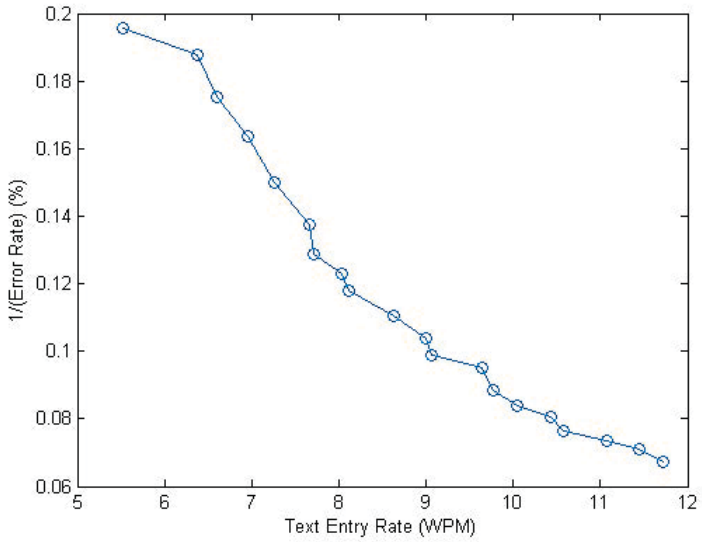


Figure 5.4: Pareto-optimal set

Then, a Fuzzy-based approach [102] is applied to choose the single best compromise solution from the non-dominated solutions. The i^{th} objective function of a solution in the non-dominated set, F_i, is represented by a membership function μ_i as defined in Eqn. 5.8.

$$\mu_i = \begin{cases} 1 & F_i \leq F_i^{min} \\ \frac{F_i^{max} - F_i}{F_i^{max} - F_i^{min}} & F_i^{max} < F_i < F_i^{min} \\ 0 & F_i \geq F_i^{max} \end{cases} \tag{5.8}$$

Here, the maximum and minimum values of i^{th} objective function is denoted by F_{max}^i and F_{min}^i, respectively. For each non-dominated solution j, the normalized

membership function μ^j is calculated as shown in Eqn. 5.9, where n is the number of objective functions considered and m is the number of non-dominated solutions. Here, the value of m and n are 2, 20 respectively.

$$\mu^j = \frac{\sum_{i=1}^{n} \mu_i^j}{\sum_{j=1}^{m} \sum_{i=1}^{n} \mu_i^j} \tag{5.9}$$

The solution having the maximum value of μ^j is the best compromise solution for all objectives. Here, we use the design parameter values obtained from the best compromise solution to design our proposed multi-objective optimized virtual keyboard, named as *SpErOpT* (Speed and Error Optimized virtual keyboard), in Bengali. Here, values of key size, space between keys and number of groups are 8 mm, 2 mm and 3, respectively. The design is depicted in Fig. 5.5.

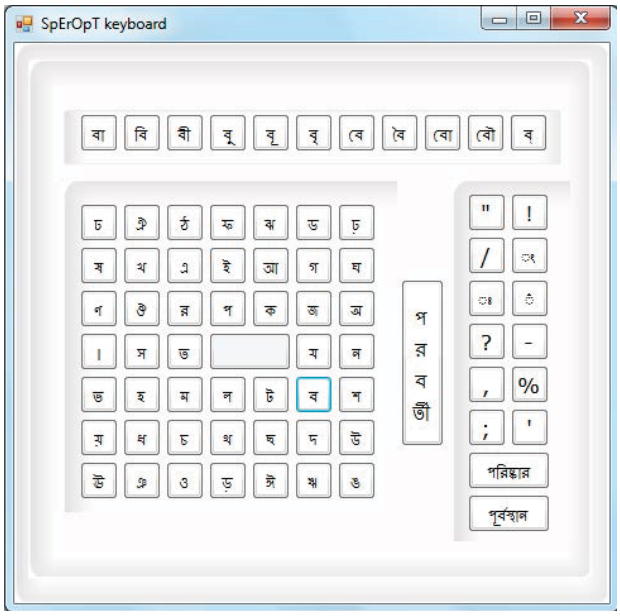


Figure 5.5: *SpErOpT* keyboard

5.3 Alternate Design to Maximize Text Entry Rate

To measure the effectiveness of *SpErOpT* keyboard design, we design another keyboard layout considering maximization of text entry rate only. Here, keyboard layout is divided into three panels, that is, character, inflexion and punctuation panels which is similar to *SpErOpT* keyboard architecture. Further, the character panel is subdivided into three zones namely, Zone 1, Zone 2 and Zone 3 as shown in Fig. 5.6. Then, we analyze the *Wikipedia* Bengali corpus to compute frequency of occurrences of characters. We assign the characters having the higher-frequency value (like: র [ra], ক [ka], ম [ma],স [sa], ত [ta] etc.) in central zone (Zone 1). Medium frequent characters (like: শ [sha], জ [ja], দ [da], ধ [dha], ভ [bha] etc.) are placed in Zone 2 surrounding the central zone. The remaining characters (like: ঈ [ii], ঐ [ai], ঢ [Dha] etc.) are resided at Zone 3 which is outside of the Zone 2. Then, we apply Genetic Algorithm (GA) to arrange the characters of the three zones, which is discussed below.

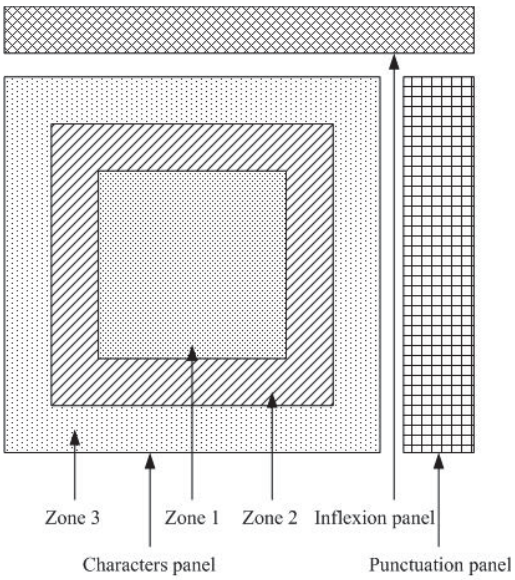


Figure 5.6: Character panel and zonal architecture

5.3.1 Optimal Arrangement of the Keys using GA

We apply Genetic Algorithm (GA) to arrange the characters of the three zones in such a way that it leads to maximizing text entry rate. Mean motor movement time (MT_{MEAN}), calculated from Fitts-digraph model [12, 24], is used as fitness function to maximize text entry rate (Eqn. 5.5).

Here, we randomly generate 20 chromosomes each of 49 bits by arbitrarily arranging all the keys of the character panel according to each zone. Each bit of chromosome represents the position of a particular key where first 7 bits denote the first row, second 7 bits for second and so on. After generating the initial population, fitness function for each chromosome is calculated.

We follow the real coded encoded GA [103] for arranging the characters of each zones. We randomly choose single crossover point in each zone for a pair of selected individual. Then we perform substring crossover [100] for each zone separately with crossover probability (p_c) 0.9. To provide more randomness, we execute mutation operation, which consists of swapping the locations of two randomly selected bits with mutation probability (p_m) 0.1 with respect to a zone. Here, the mutation operation is also performed according to the zone. So, we perform crossover and mutation operation in such a way that zonal arrangement of any character is preserved every time. In other word, mixing between characters belonging to different zones is not possible.

The, objective function MT_{MEAN} as shown in Eqn. 5.5 is calculated using Fitts-digraph model [12, 24]. After calculating objective function for each individual, we select the chromosomes for crossover using the 'rank-based selection' [86] method as shown in Eqn. 4.14 defined in Chapter 4. Individual of a generation is selected according to their fitness value.

The algorithm has been executed iteratively up to 50 generations for finding the optimal solution. Here, we employ GA for each zone separately and final possible solutions for each zone are merged together to find the optimal solution. The Bengali virtual keyboard generated using our proposed approach is shown in Fig. 5.7. We name the optimized keyboard as *MaxTER* keyboard (Text Entry Rate Maximized virtual keyboard).

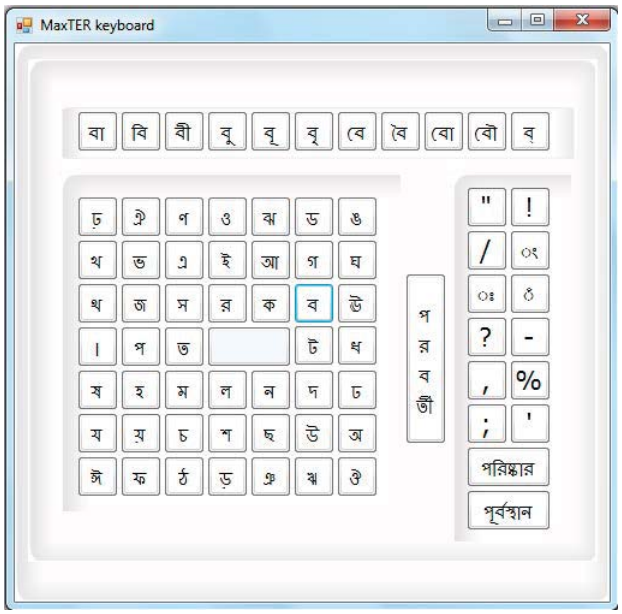


Figure 5.7: *MaxTER* keyboard

5.4 Empirical Study for Design Validation

To ensure the effectiveness of our design approach, an empirical study has been carried out with different virtual keyboards and participants. In this study, we compare user performance in terms of TR and ER during text composition using those keyboards. The method and results of the study are reported in the following.

5.4.1 Experiments

15 other participants (nine males and six females) of expert, intermediate and novice categories as discussed previously in Chapter 3, are selected in our experiments. Previously selected text corpora (Chapter 4) are used in this experimental study. Here, we used total six virtual keyboards namely, *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B*, *MaxTER* and *SpErOpT* keyboard. In each experiment, a randomly selected keyboard interface is given to each user for typing the selected text corpus freely by following previously mentioned experimental setup and procedure as discussed in Chapter 3. During experiments, the composed text through each keyboard are stored into separate log file for each user, from which users' text entry rate and typing error rate are calculated.

5.4.2 Experimental Results

Average text entry rate and typing error rate are calculated from the log files. The average text entry rate is calculated as word per minute (*WPM* [52]) as shown in Eqn. 5.10.

$$TE\ in\ WPM = \frac{|T| - 1}{S} \times \frac{60}{\overline{w}} \tag{5.10}$$

Here, $|T|$ is the length of transcribed text entered by user that is the number of characters entered. S represents the time taken for entering text by a user in seconds, measured from the entry of the first character to the last, including backspaces. $\overline{w}$ of Eqn. 5.10 denotes the average length of words for a language (5.11 for Bengali [104]) and the -1 in the numerator signifies that the time S is measured from the entry of the first character to the entry of the last character (i.e. the total text length $|T| - 1$). Moreover, we also calculate text entry rate for correctly typed characters (TE_{ctc}) by subtracting the number of typing errors (Er_n) from the entire typed text ($|T|$), as shown in Eqn. 5.11.

$$TE_{ctc}\ in\ WPM = \frac{(|T| - 1) - Er_n}{S} \times \frac{60}{\overline{w}} \tag{5.11}$$

User average text entry rate (TE) and typing error rate (ER) for each keyboard is reported in Table 5.2. It substantiates that our proposed *SpErOpT* keyboard achieves higher text entry rate and lesser typing error rate compared to other keyboards except *MaxTER*. Subsequent analysis reveals that *SpErOpT* keyboard has 29.04%, 42.29%, 42.58%, 17.59% more average text entry rate for all corpora in comparison to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B* keyboard, respectively. However, it has 3.26% lesser text entry rate compared to *MaxTER*. Moreover, *SpErOpT* keyboard has 46.07%, 64.28%, 61.62%, 44.76%, 49.74% lesser typing error rate compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B*, *MaxTER* keyboard, respectively.

User performance on each virtual keyboard interface is also graphically presented in Fig. 5.8. Text entry rate on different keyboard interfaces for composing each text corpus and average text entry rate for each keyboard is shown in Fig. 5.8a, where keyboard interfaces and text entry rates are represented by x- and y-axis, respectively. Similarly, typing error rate on different keyboard interfaces for composing each text corpus and average typing error rate for each

Table 5.2: User performance on different virtual keyboards

Text Under Test	Keyboard Interfaces											
	Avro		Lipik		Gate2Home		iLiPi-B		MaxTER		SpErOpT	
	TE	ER	TE	ER	TE	ER	TE	ER	TE	ER	TE	ER
TB_{v1}	5.23	11.58	4.85	17.45	4.81	16.21	5.77	11.37	7.13	12.64	6.87	6.29
TB_{v2}	5.39	11.51	4.92	17.36	4.96	16.16	5.98	11.22	7.31	12.29	7.15	6.22
TB_{v3}	5.72	11.46	5.13	17.31	5.07	16.11	6.18	11.18	7.53	12.26	7.28	6.17
TB_{v4}	5.83	11.40	5.19	17.26	5.21	16.08	6.39	11.09	7.69	12.12	7.39	6.10

TE = Text Entry Rate, ER = Error Rate

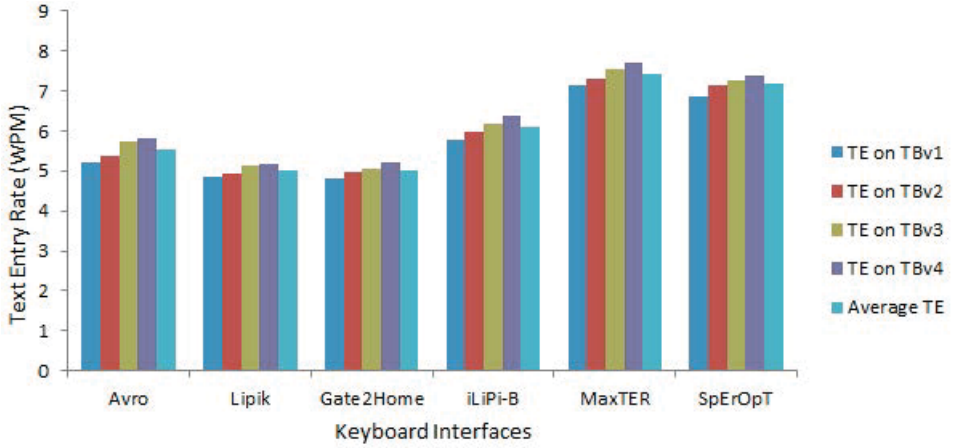
keyboard is shown in Fig. 5.8b, where keyboard interfaces are shown through x-axis and typing error rates are represented through y-axis.

Text entry rate for correctly typed characters (TE_{ctc}) for each keyboard is reported in Table 5.3. It reveals that, *SpErOpT* keyboard has 37.25%, 62.08%, 60.07%, 24.69%, 3.50% more average text entry rate for correctly typed characters compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B* and *MaxTER* keyboard, respectively. It concludes that *SpErOpT* out perform all other keyboards.

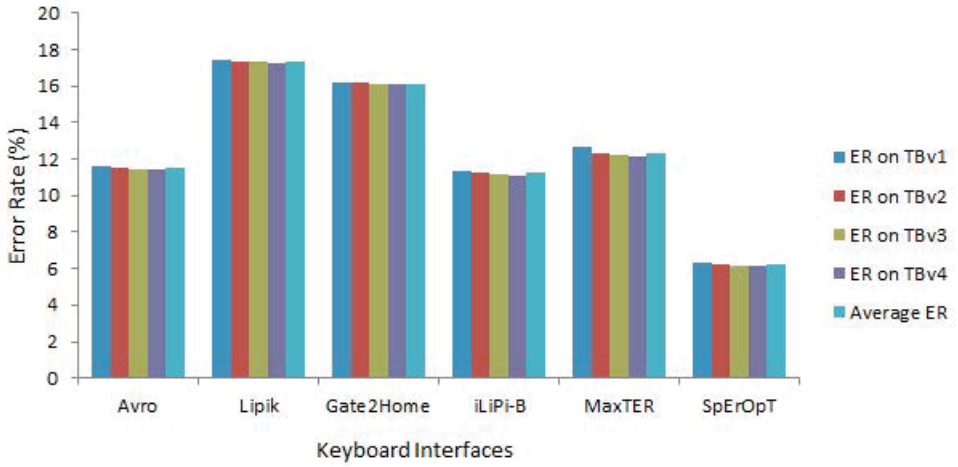
Table 5.3: Text Entry Rate for correctly typed characters on different virtual keyboards

Text Under Test	Text Entry Rate for correctly typed characters (WPM)					
	Avro	Lipik	Gate2Home	iLiPi-B	MaxTER	SpErOpT
TB_{v1}	4.62	4.00	4.03	5.11	6.23	6.44
TB_{v2}	4.77	4.07	4.16	5.31	6.41	6.71
TB_{v3}	5.06	4.24	4.25	5.49	6.61	6.83
TB_{v4}	5.17	4.29	4.37	5.68	6.76	6.94

Text entry rate for correctly typed characters on each virtual keyboard is also graphically illustrated in Fig. 5.9, where keyboard interfaces and text entry rate for correctly typed characters are represented through x- and y-axis, respectively.



(a) Text entry rate on different keyboard interfaces



(b) Typing error rate on different keyboard interfaces

Figure 5.8: User performance on different virtual keyboard interfaces

5.5 Summary

In this chapter, we propose the design of a multi-objective optimized Bengali virtual keyboard (*SpErOpT*) using NSGA-II, to maximize text entry rate and minimize typing error rate. To judge the efficacy of *SpErOpT* keyboard, we design another alternate virtual keyboard (*MaxTER*) using GA, maximizing text entry rate only. Then, an empirical study has been carried out with the existing virtual keyboards on the basis of text entry rate, error rate and text entry rate for correctly typed characters. The empirical result concludes that *SpErOpT* keyboard has

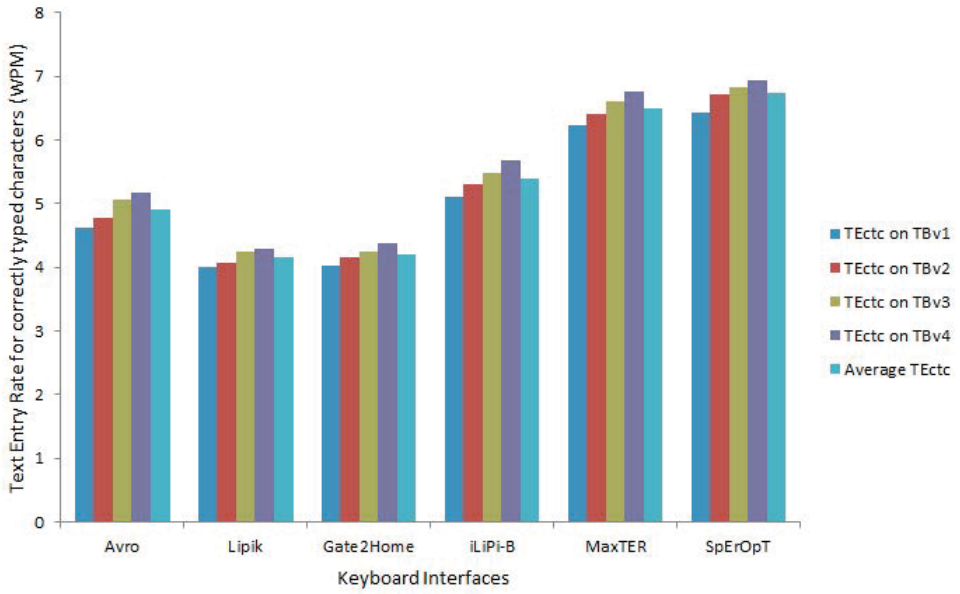


Figure 5.9: Text entry rate for correctly typed characters on different virtual keyboard interfaces

29.04%, 42.29%, 42.58%, 17.59% more average text entry rate compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B* keyboard, respectively. However, it has 3.26% lesser text entry rate compared to *MaxTER*. Moreover, *SpErOpT* keyboard has 46.07%, 64.28%, 61.62%, 44.76%, 49.74% lesser typing error rate and 37.25%, 62.08%, 60.07%, 24.69%, 3.50% more average text entry rate for correctly typed characters compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B*, *MaxTER* keyboard, respectively. The above discussed empirical result reveals that our proposed *SpErOpT* keyboard perform much better compared to all other keyboards except *MaxTER* keyboard in the context of text entry rate. However, *SpErOpT* keyboard outperforms all other keyboards for error rate and text entry rate for correctly typed characters. It establishes the efficacy of our proposed *SpErOpT* keyboard over other keyboards.

Chapter 6

Conclusion and Future Work

In this thesis, designing of an efficient virtual keyboard (*SpErOpT* keyboard) for maximizing text entry rate and minimizing typing error rate in Bengali has been proposed. Text entry rate is computed by Fitts-digraph model [12]. However, no model exist to predict the typing error on different virtual keyboard design parameters. To overcome this limitation, there is a need to identify the design parameters which put significant influence on typing error rate. To accomplish this, we have listed typing error influencing design parameters. Then, we have performed several experiments on those identified parameters with users having different level of expertise. Moreover, statistical analysis on those experimental results has been carried out which establishes that there are four virtual keyboard design parameters which play a significant role on user typing errors during text entry. The parameters are *key size*, *space between keys*, *distance between GSC pairs* and *key grouping*.

Next, a predictive error model based on those identified parameters using ANN-GA hybrid soft computing approach has been developed. This model is able to predict the typing error rate for a virtual keyboard instance with different combination of design parameter values. To judge the efficiency of our proposed model, similar model using ANN approach is also developed. The models are validated with both in-domain and out-domain data on the basis of various statistical performance metrics like *MAPE*, *MSE*, *SDE* and *R*. The experimental result concludes that for both in-domain and out-domain data, proposed model performs better than ANN model for each performance metric.

Then, we design a multi-objective optimized Bengali virtual keyboard (*SpErOpT*) using NSGA-II to maximize text entry rate and minimize typing error rate. Text entry rate and typing error rate are measured using Fitts-digraph model [12, 24] and our proposed error model, respectively. To judge the efficacy of the *SpErOpT* keyboard, we design another alternate virtual keyboard (*MaxTER*) by employing GA to maximize text entry rate only. Then, empirical studies have been carried out with the existing virtual keyboards to measure text entry rate, error rate and text entry rate for correctly typed text. The empirical result substantiates that *SpErOpT* keyboard has 29.04%, 42.29%, 42.58%, 17.59% more average text entry rate compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B* keyboards, respectively. However, it has 3.26% lesser text entry rate than *MaxTER*. Moreover, *SpErOpT* keyboard has 46.07%, 64.28%, 61.62%, 44.76%, 49.74% lesser typing error rate and 37.25%, 62.08%, 60.07%, 24.69%, 3.50% more average text entry rate for correctly typed characters compared to *Avro*, *Lipik*, *Gate2Home*, *iLiPi-B*, *MaxTER* keyboard, respectively. The above discussed empirical results reveal that our proposed *SpErOpT* keyboard perform much better than all other keyboards except *MaxTER* keyboard in context of text entry rate only. However, *SpErOpT* keyboard outperforms all other keyboards for error rate and text entry rate for correctly typed characters. The result establishes the efficacy of our proposed *SpErOpT* keyboard over other keyboards. This chapter summarizes the major contributions of our work and future scope for extending the research.

6.1 Contribution of Our Work

Several contributions have been made in order to design multi-objective optimized virtual keyboard (*SpErOpT* keyboard). The contributions are listed below.

- **Virtual keyboard design parameter identification:** Several typing error influencing virtual keyboard design parameters are listed. Then, several empirical studies and *ANOVA* tests have been carried out on those parameters, which establish that there are four parameters which have significant influence on typing error rate. The parameters are *key size*, *space between keys*, *distance between GSC pairs* and *key grouping*.

- **Modeling of typing error:** A predictive error model based on those identified parameters using ANN-GA hybrid soft computing approach has been developed. The model is capable of predicting the user typing error rate for a virtual keyboard instance with different combinations of design parameter values.
- **Character panels and multi-zonal architecture:** Different panels and multi-zonal architecture of Bengali virtual keyboard is proposed to properly position the large alphabet set into single tap virtual keyboard. The keyboard layout is divided into three panels namely character, inflexion and punctuation panel. Moreover, the character panel is divided into three zones namely, Zone 1, Zone 2 and Zone 3. Characters are grouped according to their frequency, i.e. higher, medium, lower, and placed in Zone 1, 2, 3, respectively.
- **Keyboard Design to Maximize Text Entry Rate:** A Bengali virtual keyboard design (*MaxTER* keyboard) is proposed to maximize text entry rate by employing GA in each Zone separately and final solution comes out by merging the solution of each Zone together.
- **Keyboard design to maximize text entry rate and minimize error rate:** We design a virtual keyboard (*SpErOpT* keyboard) based on multi-objective optimization using NSGA-II algorithm based on two optimality metrics, maximizing text entry rate and minimizing user error rate influenced by design parameters. Here, text entry rate is computed by Fitts-digraph model [12] and typing error rate is computed by our proposed error model.

6.2 Future Work

We have designed a multi-objective optimized virtual keyboard *SpErOpT* using NSGA-II algorithm to maximize text entry rate and minimize error rate. Apart from Genetic Algorithm approach (NSGA-II), multi-objective optimization problem can be solved through many other ways like Simulated Annealing [105], Ant Colony Optimization algorithm [62, 106] and Particle Swarm Optimization algorithm [107], Multilevel Programming [108] etc.

Investigating the applicability of solving current problem by other multi-objective optimization approaches may be the avenue of future work. We proposed an error model using ANN-GA hybrid approach to predict the typing error rate for a virtual keyboard. The error model is created for single-tap Bengali virtual keyboard. In future this model can be extended for multi-tap virtual keyboard for mobile devices. Further, our approach to design multi-objective optimized virtual keyboard can easily be extended to many other Indian languages as well as other languages.

Publications out of this work

Journal

- **Soumalya Ghosh** and Debasis Samanta, "Modeling of User Error Behaviour with Virtual Keyboard Interface", *Behaviour & Information Technology, Taylor & Francis* (Communicated).

- **Soumalya Ghosh** and Debasis Samanta, "Analysis on User Errors in Virtual Keyboards", *International Journal of Computers and Applications, ACTA Press* (Minor Revision).

- Debasis Samanta, Sayan Sarcar and **Soumalya Ghosh**, "An Approach to Design Virtual Keyboards for Text Composition in Indian Languages", *International Journal of Human-Computer Interaction, Taylor & Francis*, 2012 (Accepted and in press).

Conference

- **Soumalya Ghosh**, Debasis Samanta and Monalisa Sharma, "Cost of Error Correction Quantification with Bengali Text Transcription", *In Proceedings of the* 4^{th} *International Conference on Intelligent Human Computer Interaction (IHCI)*, IEEE, 2012, pp. $1-6$, $27-29$ December, 2012, Kharagpur, India.

- Debasis Samanta, **Soumalya Ghosh**, Somnath Dey, Sayan Sarcar, Manoj Kumar Sharma, Pradipta Kumar Saha and Santa Maiti, "Development of Multimodal User Interfaces to Internet for Common People", *In Proceedings of the 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, IEEE, 2012, pp. $1-6$, $27-29$ December, 2012, Kharagpur, India.

- **Soumalya Ghosh**, Sayan Sarcar and Debasis Samanta, "Designing an Efficient Virtual Keyboard for Text Composition in Bengali", *In Proceedings*

of the 3rd International Conference on Human Computer interaction, ACM, pp: 84 – 87, 7 – 11 April, 2011, Bangalore, India.

- Manoj Kumar Sharma, Pradipta Kumar Saha, Sayan Sarcar, **Soumalya Ghosh** and Debasis Samanta, "Accessing Dynamic Webpages in Users Languages", *In Proceedings of the Students' Technology Symposium (TechSym)*, IEEE, pp. 35 – 38, January, 2011, Kharagpur, India.

- **Soumalya Ghosh**, Sayan Sarcar, Manoj Kumar Sharma and Debasis Samanta, "Effective Virtual Keyboard Design with Size and Space Adaptation", *In Proceedings of the Students' Technology Symposium (TechSym)*, IEEE, pp. 262 – 267, 3 – 4 April, 2010, Kharagpur, India

- Sayan Sarcar, **Soumalya Ghosh**, Pradipta Kumar Saha and Debasis Samanta, "Virtual Keyboard Design: State of the Arts and Research Issues", *In Proceedings of the Students' Technology Symposium (TechSym)*, IEEE, pp. 289 – 299, 3 – 4 April, 2010, Kharagpur, India.

References

[1] G. Sholes, C. Glidden, and S. Soule, "Improvement in type-writing machines," U.S Patent 79,868, 1868.

[2] Fitaly, "The One-Finger Keyboard," [Online] Available: http://www.fitaly.com/fitaly/fitaly.htm, accessed on July 2010.

[3] Addictivetips, "G-Board Lite For Android Is A Fast Gesture-Based Keyboard (IME)," [Online] Available: http://www.addictivetips.com/mobile/g-board-lite-for-android-is-a-fast-gesture-based-keyboard-ime, accessed on July 2012.

[4] K. Vertanen and P. O. Kristensson, "Parakeet : A Continuous Speech Recognition System for Mobile Touch-screen Devices," in *Proceedings of the* 14^{th} *International Conference on Intelligent User Interfaces*, Florida, USA, 2009, pp. 237 – 246.

[5] P. Majaranta, "Text Entry by Eye Gaze," Ph.D. dissertation, University of Tampere, Tampere, Finland, 2009.

[6] A. Dvorak, N. L. Merrick, W. L. Dealey, and G. C. Ford, *Typewriting Behaviour : Psychology applied to Teaching and Learning Typewriting*. New York, USA: American Book Company, 1936.

[7] R. A. Chubon and M. R. Hester, "An Enhanced Standard Computer Keyboard System for Single-finger and Typing-stick Typing," *Journal of Rehabilitation Research and Development*, vol. 25, no. 4, pp. 17 – 24, 1988.

[8] J. Mankoff and G. D. Abowd, "Cirrin: A Word-Level Unistroke Keyboard for Pen Input," in *Proceedings of the* 11^{th} *annual ACM symposium on User Interface Software and Technology*, California, USA, 1998, pp. 213 – 214.

[9] I. S. MacKenzie and S. X. Zhang, "The Design and Evaluation of a High-performance Soft Keyboard," in *Proceedings of Human Factors in Computing Systems*, Pittsburgh, USA, 1999, pp. 25 – 31.

[10] J. R. Lewis, P. J. Kennedy, and M. J. LaLomia, "Development of a Digram-Based Typing Key Layout for Single-Finger / Stylus Input," in *Proceedings of the Human Factors and Ergonomics Society* 43^{rd} *Annual Meeting*, Texas, USA, 1999, pp. 415 – 419.

[11] S. Zhai, M. Hunter, and B. A. Smith, "The Metropolis keyboard - An Exploration of Quantitative Techniques for Virtual Keyboard Design," in *Proceedings of the* 13^{th} *Annual ACM Symposium on User Interface Software and Technology*, California, USA, 2000, pp. 119 – 128.

[12] S. Zhai, M. Hunter, and B. A. Smith, "Performance Optimization of Virtual Keyboards," *Human-Computer Interaction*, vol. 17, no. 2 – 3, pp. 89 – 129, 2002.

[13] M. Raynal and N. Vigouroux, "Genetic Algorithm to Generate Optimized Soft Keyboard," in *Proceedings of Human Factors in Computing Systems - Extended Abstracts*, Portland, USA, 2005, pp. 1729 – 1732.

[14] Wikipedia, "Keyboard Layout," [Online] Available: http://en.wikipedia.org/wiki/Keyboard_layout, accessed on Jun 2011.

[15] C. D. GIST, "A Document for Enhanced InScript Keyboard Layout 5.2," [Online] Available: http://malayalam.kerala.gov.in/images/8/80/Qwerty_enhancedinscriptkeyboardlayout.pdf, accessed on January 2010.

[16] Gate2Home, "Gate2home," [Online] Available: http://gate2home.com/Bengali-Keyboard#IN Bengali Inscript, accessed on August 2010.

[17] Lipik, "Lipik: A Predictive Text Input System," [Online] Available: http://www.lipik.in, accessed on August 2010.

[18] Lookeys, "Indian Languages Virtual Keyboard," [Online] Available: http://www.lookeys.com/?page_id=01503, accessed on August 2010.

[19] O. Lab, "Avro Bengali keyboard," [Online] Available: http://www.omicron-lab.com/avro-keyboard.html, accessed on August 2010.

[20] Guruji, "Indian Internet Search Engine," [Online] Available: http://www.guruji.com/hi/index.html, accessed on February 2010.

[21] D. Samanta, S. Sarcar, and S. Ghosh, "An Approach to Design Virtual Keyboards for Text Composition in Indian Languages," International Journal of Human-Computer Interaction, Available online, http://dx.doi.org/10.1080/10447318.2012.728483, 2012.

[22] X. Bi, B. A. Smith, and S. Zhai, "Quasi-qwerty Soft Keyboard Optimization," in *Proceedings of Human Factors in Computing Systems*, Atlanta, USA, 2010, pp. 283 – 286.

[23] M. D. Dunlop and J. Levine, "Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed, Familiarity and Improved Spell Checking," in *Proceedings of Human Factors in Computing Systems*, Austin, USA, 2012, pp. 2669 – 2678.

[24] R. W. Soukoreff and I. S. MacKenzie, "Theoretical Upper and Lower Bounds on Typing Speed using a Stylus and Soft Keyboard," *Behaviour & Information Technology*, vol. 14, no. 6, pp. 370 – 379, 1995.

[25] J. L. Arnott, "Text entry in Augmentative and Alternative Communication," in *Proceedings of Efficient Text Entry*, 2005.

[26] C. Aliprandi, N. Carmignani, and P. Mancarella., "An Inflected-sensitive Letter and Word Prediction System," *International Journal of Computing & Information Sciences*, vol. 5, no. 2, pp. 79 – 85, 2007.

[27] J. O. Wobbrock and B. A. Myers, "From Letters to Words: Efficient Stroke-based Word Completion for Trackball Text Entry," in *SIGACCESS Conference on Computers and Accessibility*, Oregon, USA, 2006, pp. 2 – 9.

[28] S. Ghosh, S. Sarcar, M. K. Sharma, and D. Samanta, "Effective Virtual Keyboard Design with Size and Space Adaptation," in *Proceedings of Students' Technology Symposium*, Kharagpur, India, 2010, pp. 262 – 267.

[29] S. Jain and S. Bhattacharya, "Predictive Error Behavior Model of On-screen Keyboard Users," in *Proceedings of Human Factors in Computing Systems - Extended Abstracts*, Vancouver, Canada, 2011, pp. 1435 – 1440.

[30] R. M. K. Sinha, "Man-Machine Integration in Translation Processes: an Indian Scenario," [Online] Available: http://mt-archive.info/NLPSC-2011-Sinha.pdf, accessed on January 2012.

[31] I. S. MacKenzie and R. W. Soukoreff, "Text Entry for Mobile Computing: Models and Methods, Theory and Practice," *Human–Computer Interaction*, vol. 17, no. 2 – 3, pp. 147–198, 2002.

[32] T. E. Hutchinson, K. P. White(Jr), W. N. Martin, K. C. Reichert, and L. A. FREY, "Human-Computer Interaction using Eye-Gaze Input," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1527–1534, 1989.

[33] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz, "Sensing Techniques for Mobile Interaction," in *Symposium on User Interface Software and Technology*, San Diego, USA, 2000, pp. 91 – 100.

[34] X. Bi, B. A. Smith, and S. Zhai, "Multilingual Touchscreen Keyboard Design and Optimization," *Human Computer Interaction*, vol. 27, no. 4, pp. 352 – 382, 2012.

[35] S. Bhattacharya, A. Basu, and D. Samanta, "Performance Models for Virtual Scanning Keyboards: Reducing User Involvement in the Design," in *Proceedings of the International Conference on Information and Communication Technologies for Development*, Bangalore, India, 2007, pp. 1 – 8.

[36] S. Sarcar, S. Ghosh, P. K. Saha, and D. Samanta, "Virtual Keyboard Design: State of the Arts and Research Issues," in *Proceedings of Students' Technology Symposium*, Kharagpur, India, 2010, pp. 289 – 299.

[37] Wikipedia, "Languages of india," [Online] Available: http://en.wikipedia.org/wiki/Languages_of_India, accessed on August 2011.

[38] Wikipedia, "List of Languages by Number of Native Speakers," [Online] Available: http://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers, accessed on April 2011.

[39] W. T. White, "Typing for Accuracy." H.M. Rowe Company, 1945.

[40] P. F. MacNeilage, "Typing Errors as Clues to Serial Ordering Mechanisms in Language Behaviour," *Language and Speech*, vol. 7, no. 3, pp. 1144 – 1159, 1964.

[41] D. R. Gentner, J. T. Grudin, S. Larochelle, D. A. Norman, and D. E. Rumelhart, *Cognitive Aspects of Skilled Typing*. New York, USA: Springer Verlag, 1983, ch. A Glossary of Terms Including a Classification of Typing Errors, pp. 39 – 43.

[42] M. Bhagat, "Spelling Error Pattern Analysis of Punjabi Typed Text," Master's thesis, Thapar University, Patiala, India, 2007.

[43] J. O. Wobbrock and B. A. Myers, "Analyzing the Input Stream for Character- Level Errors in Unconstrained Text Entry Evaluations," *ACM Transactions on Computer-Human Interaction*, vol. 13, no. 4, pp. 458 – 489, 2006.

[44] A. Kano, J. C. Read, A.Dix, and I. S. MacKenzie, "Expect: An Expanded Error Categorisation Method for Text Input," in *Proceedings of the* 21^{st} *British CHI Group Annual Conference on Human Computer Interaction*, Lancaster, UK, 2007, pp. 147 – 156.

[45] T. Chen, Y. Yesilada, and S. Harper, "What Input Errors Do You Experience? Typing and Pointing Errors of Mobile Web Users," *International Journal of Human-Computer Studies*, vol. 68, no. 3, pp. 138 – 157, 2010.

[46] S. Trewin, "Automating Accessibility : the Dynamic Keyboard," in *Proceedings of the* 6^{th} *International ACM SIGACCESS Conference on Computers and Accessibility*, Atlanta, USA, 2004, pp. 71 – 78.

[47] S. Kane, J. Wobbrock, M. Harniss, and K. Johnson, "TrueKeys: Identifying and Correcting Typing Errors for People with Motor Impairments," in *Proceedings of the* 13^{th} *International Conference on Intelligent User Interfaces*, Canary Islands, Spain, 2008, pp. 349 – 352.

[48] K. . Z. Gajos, J. O. Wobbrock, and D. S. Weld, "Improving the Performance of Motor - Impaired Users with Automatically-Generated, Ability-Based Interfaces," in *Proceeding of the* 26^{th} *Annual SIGCHI Conference on Human Factors in Computing Systems*, New York, USA, 2008, pp. 1257 – 1266.

[49] S. Bhattacharya, A. Basu, and D. Samanta, "Computational Modeling of User Errors for the Design of Virtual Scanning Keyboards," *Transactions on Neural Sytems and Rehabilitation Engineering*, vol. 16, no. 4, pp. 400 – 409, 2008.

[50] S. Bhattacharya, D. Samanta, and A. Basu, "User Errors on Scanning Keyboards: Empirical Study, Model and Design Principles," *Interacting with Computers*, vol. 20, no. 3, pp. 406 – 418, 2008.

[51] Wikipedia, "Curve fitting," [Online] Available: http://en.wikipedia.org/wiki/Curve_fitting, accessed on July 2012.

[52] I. S. MacKenzie and K. Tanaka-Ishii, *Text Entry Systems: Mobility, Accessibility, Universality*, 1^{st} ed., S. Card, J. Grudin, and J. Nielsen, Eds. Morgan Kaufmann Inc, 2007.

[53] K. Shieh and C. Lin, "A Quantitative Model for Designing Keyboard Layout," *Perceptual and Motor Skills*, vol. 88, no. 1, pp. 113 –125, 2009.

[54] P. M. Fitts, "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381 – 391, 1954.

[55] D. A. Norman and D. Fisher, "Why Alphabetic Keyboards Are Not Easy to Use: Keyboard Layout Doesn't Much Matter," *The Journal of the Human Factors and Ergonomics Society*, vol. 24, no. 5, pp. 509 – 519, 1982.

[56] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics: an Introduction*, W. T. Rhodes and H. E. Stanley, Eds. Springer, 2010, vol. 80.

[57] D. E. Goldberg and J. H. Holland, "Genetic Algorithms and Machine Learning," *Machine Learning*, vol. 3, no. 2, pp. 95 – 99, 2005.

[58] TDIL, "iLeap - Multilingual Word Processor," [Online] Available: http://www.cdac.in/html/gist/products/ileap.asp, accessed on Jun 2010.

[59] TDIL, "Inscript Keyboard," [Online] Available: http://tdil.mit.gov.in/key-overlay.htm, accessed on October 2010.

[60] Microsoft, "Onscreen Keyboard," [Online] Avalible: http://windows.microsoft.com/en/windows7/Type-without-using-the-keyboard-On-Screen-Keyboard, accessed on February 2010.

[61] J. Eggers, D. Feillet, S. Kehl, M. O. Wagner, and B. Yannou, "Optimization of the Keyboard Arrangement Problem using an Ant Colony Algorithm," *European Journal of Operational Research*, vol. 148, no. 3, pp. 672 – 686, 2003.

[62] M. Dorigo and L. M. Gambardella, "Ant System : Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems Man Cybernetics, Part B*, vol. 26, no. 2, pp. 29 – 41, 1996.

[63] S. Deshwal and K. Deb, "Design of an Optimal Hindi Keyboard for Convenient and Efficient Use," Technical Report KanGAL, Indian Institute of Technology, Kanpur, 2003.

[64] P. Y. Yin and E. P. Su, "Cyber Swarm Optimization for General Keyboard Arrangement Problem," *International Journal of Industrial Ergonomics*, vol. 41, no. 1, pp. 43 – 52, 2011.

[65] P. Y. Yin, F. Glover, M. Laguna, and J. X. Zhu, "Cyber Swarm Algorithms – Improving Particle Swarm Optimization using Adaptive Memory Strategies," *European Journal of Operational Research*, vol. 201, no. 2, pp. 377 – 389, 2010.

[66] T. A. Salthouse, "Perceptual, Cognitive, and Motoric Aspects of Transcription Typing," *Psychological Bulletin*, vol. 99, no. 3, pp. 309 – 319, 1986.

[67] Y. L. Lin, M. C. Chen, Y. P. Wu, Y. M. Yeh, and H. P. Wang, "A Flexible On-screen Keyboard: Dynamically Adapting for Individuals' Needs," in *Proceedings of HCI International*, Beijing, China, 2007, pp. 371 – 379.

[68] W. McCulloch and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115 –133, 1943.

[69] R. Rosenblatt, *Principles of Neurodynamics*. Spartan Books, 1962.

[70] M. Minsky and S. Papert, *Perceptrons : An lntroduction to Computational Geometry*. MIT Press, 1969.

[71] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," in *Proceedings of National Academy of Sciences*, vol. 79, no. 8, USA, 1982, pp. 2554–2558.

[72] P. Werbos, "Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation, Harvard University, 1974.

[73] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Rocessing : Exploration in the Microstructure of Cognition*. MIT Press, 1986.

[74] M. Hajmeer, I. Basheer, J. Marsden, and D. Fung, "New Approach for Modeling Generalized Microbial Growth Curves using Artificial Neural Networks," *Journal of Rapid Methods and Automation in Microbiology*, vol. 8, no. 4, pp. 265 – 284, 2000.

[75] C. C. Hsu and C. Y. Chen, "Regional Load Forecasting in Taiwan-Applications of Artificial Neural Networks," *Energy Conversion and Management*, vol. 44, no. 12, pp. 1941 – 1949, 2003.

[76] R. P. Lippmann, "Pattern Classification using Neural Networks," *IEEE Communication Magazine*, vol. 27, no. 11, pp. 47 – 64, 1989.

[77] D. G. Roussinov and H. Chen, "Document Clustering for Electronic Meetings: an Experimental Comparison of Two Techniques," *Decision Support Systems*, vol. 27, no. 1, pp. 67 – 79, 1999.

[78] S. Ferrari and R. Stengel, "Smooth Function Approximation using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 24 – 38, 2005.

[79] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, "Learning Representations by Backpropagating Errors," *Nature*, vol. 323, no. 6188, pp. 533 – 536, 1986.

[80] J. H. Holland, *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[81] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Springer, 2008.

[82] M. Mitchell, *An Introduction to Genetic Algorithms*, 1^{st} ed. MIT Press, 1996.

[83] E. Patuwo, M. Y. Hu, and M. S. Hung, "Two – Group Classification Problem Using Neural Networks," *Decision Sciences*, vol. 24, no. 4, pp. 825 – 845, 1993.

[84] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with Artificial Neural Networks : The State of the Art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35 – 62, 1998.

[85] A. Sedki, D. Ouazar, and E. E. Mazoudi, "Evolving Neural Network using Real Coded Genetic Algorithm for Daily Rainfall – Runoff Forecasting," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4523 – 4527, 2009.

[86] R. Sivaraj and T. Ravichandran, "A Review of Selection Methods in Genetic Algorithm," *International Journal of Engineering Science and Technology*, vol. 3, no. 5, pp. 3792 – 3797, 2011.

[87] Google, "Google Transliteration," [Online] Available: http://www.google.com/transliterate, accessed on April 2011.

[88] S. Ghosh, D. Samanta, and M. Sarma, "Cost of Error Correction Quantification with Bengali Text Transcription," in *Proceedings of the* 4^{th} *International Conference on Intelligent Human Computer Interaction*, Kharagpur, India, 2012, pp. 1–6.

[89] W. E. Hick, "On the Rate of Gain of Information," *Quarterly Journal of Experimental Psychology*, vol. 4, no. 1, pp. 11 – 26, 1952.

[90] R. Hyman, "Stimulus Information as a Determinant of Reaction Time," *Journal of Experimental Psychology*, vol. 45, no. 3, pp. 188 – 196, 1953.

[91] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*, New Jersey, USA, 1985, pp. 93 – 100.

[92] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, California, USA, 1993, pp. 416 – 423.

[93] J. Horn, N. Nafploitis, and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proceedings of the* 1^{st} *IEEE Conference on Evolutionary Computation*, New Jersey, USA, 1994, pp. 82 – 87.

[94] P. Hajela and C. Y. Lin, "Genetic Search Strategies in Multicriterion Optimal Design," *Structural and Multidisciplinary Optimization*, vol. 4, no. 2, pp. 99 – 107, 1992.

[95] N. Srinivas and K. Deb, "Muiltiobjective Optimization using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221 – 248, 1994.

[96] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182 – 197, 2002.

[97] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257 – 271, 1999.

[98] J. D. Knowles and D. W. Corne, "Approximating the Nondominated Front using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149 – 72, 2000.

[99] V. Khare, X. Yao, and K. Deb, "Performance Scaling of Multiobjective Evolutionary Algorithms," in *Proceedings of the* 2^{nd} *International Conference on Evolutionary Multi-Criterion Optimization*, Faro, Portugal, 2003, pp. 376 – 390.

[100] T. P. Hong, H. S. Wang, W. Y. Lin, and W. Y. Lee, "Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process," *Applied Intelligence*, vol. 16, no. 1, pp. 7 – 17, 2001.

[101] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. Wiley-Interscience, 2004.

[102] M. A. Abido, "Multiobjective Evolutionary Algorithms for Electric Power Dispatch Problem," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 315 – 329, 2006.

[103] A. H. Wright, "Genetic Algorithms for Real Parameter Optimization," *Foundations of Genetic Algorithms*, vol. 1, pp. 205 – 218, 1991.

[104] A. Bharati, P. Rao, R. Sangal, and S. M. Bendre, "Basic Statistical Analaysis of Corpus and Cross Comparision," in *Proceedings of International Conference on Natural Language Processing*, Hyderabad, India, 2002.

[105] D. S. Johnson, C. R. Aragon, L. A. . McGeoch, and C. Schevon, "Optimization by Simulated Annealing: an Experimental Evaluation; Part

I, Graph Partitioning," *Operations Research*, vol. 37, no. 6, pp. 865 – 892, 1989.

[106] M. Dorigo and L. M. Gambardella, "Ant Colony System : a Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53 – 66, 1997.

[107] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, Perth, Australia, 1995, pp. 1942 – 1948.

[108] J. Bracken and J. T. McGill, "Mathematical Programs with Optimization Problems in the Constraints," *Operations Research*, vol. 21, no. 1, pp. 37 – 44, 1973.