OPTIMIZATION OF SWITCH VIRTUAL KEYBOARD BY USING COMPUTATIONAL MODELLING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Xiao (Cosmo) Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2014

Purdue University

West Lafayette, Indiana

Thanks to my advisor, Dr. Gregory Francis. Without his support along the way this would not be possible. I am grateful to him.

ACKNOWLEDGMENTS

I would like to thank Gregory Francis, Richard Schweickert and Sebastien Helie for their valuable discussions and suggestions.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

WPM	Word Per Minute
MIP	Mixed Integer Programming
HCI	Human Computer Interaction
GA	Genetic Algorithm
AKP	Assignment of Keys Problem
MLE	Maximum likelihood Estimation
PM	Position Mark

ABSTRACT

Zhang, Xiao M.S., Purdue University, December 2014. Optimization of Switch Virtual Keyboard by Using Computational Modelling. Major Professor: Gregory Francis.

In this thesis, I first reviewed some keyboard technologies used by people with motor difficulties, and described design elements that influence efficiency. I cast the design of a switch keyboard as an optimization problem, and arrangement of keys on such a keyboard as a Mixed Integer Programming problem. One significant variable in the MIP problem, the error rate, is related to several other variables. I treated modeling of the error rate as a parameter estimation problem, and used a data mining method. I designed HCI experiments to gather data for parameter estimation, using Bayesian logistic regression model. The empirical data and error rate modeling allowed for construction of several different types of keyboards. These different keyboards were compared and evaluated with regard to their use by people with motor difficulties.

1. INTRODUCTION

We have entered the **digital era**. Human being's life in this age is heavily relying on digital devices, such as desktops, laptops, pads and mobile phone. They are connecting individual's life with each other's, and the entire society.

This thesis focuses on the interaction between a particular group of people and Switch typing devices, via a special method: Switch devices with a virtual keyboard. My research explores the design of such a switch system, in order to improve the efficiency and efficacy of communication.

1.1 Thesis Overview & Main Ideas

The introductory part includes an introduction of the thesis (Chapter 1), literature review (Chapter 2), and a description of the special keyboards: the Switch Virtual Keyboards that I studied in this thesis (Chapter 3).

The first part of this thesis consists of the investigation of the design of this special keyboard, by using computational modeling. In Chapter 4, this thesis regards the arrangement of keys of the keyboard as a problem of global optimization, taking into consideration a standard criterion in industry: the trade-off between speed and accuracy. In Chapter 5 this thesis continues discussion on the research by trying to look for a set of unknown parameters that lie in the global optimization. However, my approach to these unknown parameters is heuristic, and more specifically machine-learning and data-mining oriented. It constructs models based on the behavioral experimental data. Overall, I treat this part as an attempt to apply machine-learning and data-mining in HCI, which connects historical behaviorism in psychology (by

looking into behavioral data), and computational methods in computer science (by constructing computational models).

The second part is the application and extension of the modeling results from the previous part, and evaluation of these applications and extensions. Chapter 6 considers different paths of the cursor movement of the virtual keyboard, and different layouts as well. Chapter 7 discusses the realization of this virtual keyboard through two different types of input devices. Chapter 8 studies performance of the results of modeling in different categories of corpora.

The last part concludes the thesis, and points out future research direction of related studies.

1.2 Thesis Statement

This thesis is aiming at improving performance of existing switch device related virtual keyboard, by investigating assignment of keys problem. It argues key arrangement on a keyboard as a design problem can be formulated as a mathematical optimization problem, and can be solved in the form of Mixed Integer Programming (MIP). It is also advocated that machine learning and data mining approaches on behavioral experimental data can help build computational models for HCI problems. In this way, this thesis **bridges** the computational modeling method and Human-Computer Interaction (HCI) to motivate new methods and systems in design of interactive devices and softwares, which may advance the communication between human agents and machines.

1.3 People with Motor Control Difficulties

In some occasions people have difficulties using their fingers to interact with a real keyboard or panel, to input information into a machinery agent. A particular example is when a pilot is performing a flight mission (Francis and Rash, 2005; **?**).

During this kind task, the individual can hardly free their hands to conduct another typing task, since both hands are constrained onto the wheel.

Another example is the “locked-in” situation. The term “Locked-in” here refers to almost fully paralyzed patients usually with a spinal cord and/or brain injury who in most cases cannot speak. Some of them can only move one part of their hands, maybe a thumb, but are unable to lift their arms. In some severe conditions, their remaining motor control abilities are limited to the head, for instance, they only can twitch a facial muscle, blink an eye, move up and down the eyeball, suck or puff air, or just sway the head slightly.

In the above cases, individuals maybe cognitively normal. Therefore, a common way for them to input information into a machinery agent, is to trigger binary signals that will be interpreted by the machinery agent, through a switch device and an embedded virtual keyboard. An embedded virtual keyboard is shown on a screen that the user can see, while a binary signal tells the machine whether a particular character or function on the keyboard is typed or not.

2. LITERATURE REVIEW

2.1 Human-Computer Interaction as a Research Topic

Designing and manufacturing switch devices and computer programs to facilitate the functionality of users with motor control difficulties, rely on the studies of Human-Computer Interaction (HCI). (Morland, 1983,)'s discussed several topics in HCI, including the optimization of screen layouts of interactive data entry, and handling of errors, as well as practical techniques for improving man-machine interactions, based on the studies of how people interacted with a video terminal. His discussion also involved the structural and algorithmic perspectives of interactions, and related elements from cognitive psychology.

2.2 Application of Machine Learning in HCI

Experimental data in HCI research are usually behavioral experimental data. Studies focused on the importance of use behavioral data prevail in machine learning and data-mining area. (Cao, 2008,) imported the concept "Behavior Informatics" by introducing human behavior in business activities. They defined it as: "Behavior Informatics and Analytics is a scientific field, which aims to develop methodologies, techniques and typical tools for representing, Modeling, analyzing, understanding and/or utilizing symbolic and/or mapped behavior, behavioral interaction and network, behavioral patterns, behavioral impacts, the formation of behavior–oriented groups and collective intelligence, and behavioral intelligence emergence." Behavior Informatics project, from University of Birmingham defined it more generally as "the use of informatics methods applied to behavioral data and models to facilitate multi-scale, cross-disciplinary integration of knowledge about behavior" (Koene, 2014).

Application of machine learning and data mining also widely exists in HCI studies. For example, a Hidden Markov model (HHM) was applied to find the behavior pattern of users working with a head-mouse driven writing tool (Hevizi et al., 2004), following the previous Dasher (Ward et al., 2000) research that employed similar method. In their research, a special mouse system controlled by the head movements was developed to enter letters. The *candidates of letters* were carefully put on the right side of the screen, and participants had to move the mouse to the designated areas to choose desired letters. They trained the HHM by tuning the parameters until it could generate the observed variables with the highest probability.

Another renowned application of machine learning in HCI is the well-known Office assistant in the Microsoft Office product, which is the Lumiere project of the Microsoft Research Institute (Horvitz et al., 1998). They applied a Bayesian network to determine when the users need help. Though in real use, this design suffered a lot criticism, but in academia, researcher still consider it a good try.

Neural networks also have its usages in HCI area. (Fu and Ho, 2009,) designed a fast Text-based communication system for handicapped aphasics by training a three layers neural network (with one hidden layer), to recognize finger language components. After training, several particular features were forwarded into the neural network, to yield a output that classified the meaning of different signs.

2.3 Keyboard Design

The most commonly used modern keyboard "QWERTY", is initiated in the 1870s, based on a layout created for the Sholes and Glidden typewriter and sold to Remington. In this design, the keys with the most common letters were arranged in hard to reach spots, to slow down typists and try to prevent the bars from colliding with each other and jamming, as they were swung into a tape coated with ink. However researches have shown "QWERTY" is the suboptimal design of a keyboard in modern use (Gopher and Raij, 1988; **?**). An alternative is the "Dvorak" keyboard, which is

aiming at shorten the distance between finger movement on a keyboard (Mackenzie et al., 1999).

Studies also investigated virtual key board design from different aspects. (Zhai et al., 2002,) stated an excellent discussion of applying optimization methods for virtual keyboard design. They advised identified designs found by researchers are close to optimal ones. Both (Hughes et al., 2002,) and (Zhai et al., 2002,) used optimization algorithm to identify the optimal positions of characters.

Character frequencies are considered by some researchers as well. (Mayzner and Tresselt, 1965,) designed keyboard based on a large word corpus that identified words frequency.

2.4 Computational Modeling on Switch Keyboards

(Francis and Johnson, 2011,) proposed that character placement on a switch virtual keyboard could be treated as a cost minimization problem that trades off speed and accuracy. Calculating these terms requires a corpus of the kind of text that the user would enter (e.g., poetry v.s. HTML code), a model of errors, and specification of acceptable average error rate. With such information, they designed a hill-climbing algorithm that could identify the cursor duration and key arrangement that minimized entry time.

If this approach is adapted, in practical optimization, the error rate is a term that needs to be determined explicitly. Previous research (Francis and Johnson, 2011) showed that the probability of making mistakes is high when the key is placed at the beginning of each row and/or column–the smaller the column and row numbers, the more likely a mistake will be made. Also, the shorter the cursor duration, the higher the likelihood that a mistake will be made. (Francis and Johnson, 2011,) modeled their experimental data by taking advantage of a software named NUANCE, which employs a Genetic algorithm (GA) to yield a mathematical arithmetic formula that represents the relationship of variables (Hollis et al., 2006).

This result, though it partially disclosed the relationship between those concerned variables, also had some disadvantages. First, the formula was generated by combining basic arithmetic operations, including addition, subtraction, multiplication, division and powers. The formula generated in this way, unavoidably, has a lot of redundant features, therefore it does not reveal the relationship among the variables in an informative way. Also, the output of this formula, yielded by putting values of different variables, will sometimes produce an error rate that exceeds 1, which is against the basic principle of probability theory–all probabilities must be within the range 0 and 1. Though some functions (like piecewise functions) can be used to reinterpret the output and put it into the correct range, it still needs a more detailed clarification of the formula.

3. SWITCH DEVICES & VIRTUAL KEYBOARD

3.1 Introduction of Switch Devices & Virtual Keyboards

Users interact with Switch devices to input on a virtual keyboard for their special needs. A commercialized virtual keyboard is shown below in Figure 3.1, called SwitchXS, which was developed by the AssistiveWare company. It provides various modes of keyboards, including simple mouse input and complicated combined input. It functions via the received signals from a user triggering the switch device, to enable the cursor to scan the keyboard, and make key selections. This product is used for both daily text input and special commands.

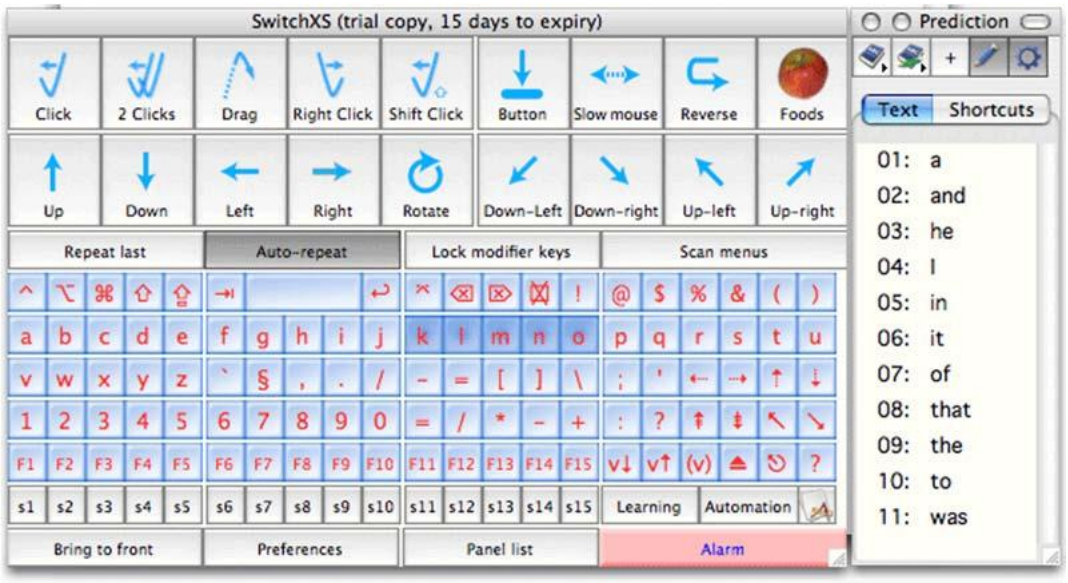


Fig. 3.1.: SwitchX keyboard

This keyboard can be user-customized in several different ways. In one design, the scanning cursor first jumps among different regions. The user has to trigger a signal when the cursor is in the designated region, and a sub-area will be chosen. Then the cursor scans key by key within the selected region as a sub-area of the entire layout, until the user triggers a signal to select the desired key. In this method, the user can first choose the region where the intended key lies, and then choose the exact key. In another design, the scanning cursor goes through all the rows first, while the user needs to trigger a signal to select a row. Then the cursor scans within this row, and the user triggers another signal when the cursor reaches and stays on a desired key, thus this final selection generates a typed input.

The keys on the keyboard code different characters, functions, and symbols. For example, in Figure 3.1, the first and second rows indicate mouse movements and actions, the third row and the last row show special functions of this keyboard, and row 4 to row 7 are of characters, while row 8 consists of function keys.

Switch virtual keyboards are used by a variety of people with motor difficulties, who are not necessarily "locked-in". Another similar system, the "Logo" keyboard, was developed for people with motor disabilities (Norte and Lobo, 2007). This system follows the following principles: two major scanning groups (numeric or alphabetic) are shown. Users trigger the switch first to select either the numeric or the alphabetic part when cursor is flashing between two groups, while the second scanning is row by row for the user to trigger to select an item of that group. The selection can be made by a switch device as well. Experimental results showed that disabled users have a better user experience with this virtual keyboard, compared with a physical keyboard, in terms of speed, usability and precision.

3.2 Design Elements of Switch Virtual Keyboards

Switch virtual keyboards have several constraints that influence their design. An important constraint is that patients interact with a computer by generating a binary

switch signal, “0” and “1” in computer language (Laureys et al., 2005), instead of a real physical movement. This property nullifies the Fitts' Law (Fitts, 1954) in this type of virtual keyboard. Therefore, other elements in designing this kind of virtual keyboards have to be taken into consideration. This section describes key aspects of switch keyboards.

3.2.1 Character Arrangement

Because the virtual keyboard is usually scanned character by character, or row by row, the character arrangement (layout), for example which key to put in a location with fewer steps, will determine the efficiency of the keyboard. In other words, the more frequently a character appears, the sooner this character should be reached by the cursor. Such an approach saves scanning time. Figure 3.2 demonstrates an alphabetic ordered keyboard and an optimized one (so that frequent characters are placed early in the cursor path), assuming the cursor will first go down row by row, and once a row is selected, the cursor will move across columns.

In the first keyboard of Figure 3.2, every character is placed in alphabetic order. Therefore, this arrangement has nothing to do with the frequency of characters appearing in a text. The next keyboard is optimized. In this arrangement, the characters of high frequencies are arranged in a way that they can be reached with fewer steps. They are arranged in early rows and/or columns.

a	b	c	d	e	f	g	h
i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x
y	z		,	.	?	(	)
”	/	&	{	*	_	’	%
}	<	@	:	∧	[	]	>
$	!	;	-	+	=	0	1
2	3	4	5	6	7	8	9

(a) An alphabetic layout of characters

	a	s	u	c	g	h	z
e	n	t	m	f	x	:	(
i	r	d	q	b	=	∧	w
l	p	.	<	-	)	/	+
o	v	j	[		;	*	_
$	y	?	”	{	!	’	}
>	k	%	]	@	&	0	1
2	3	4	5	6	7	8	9

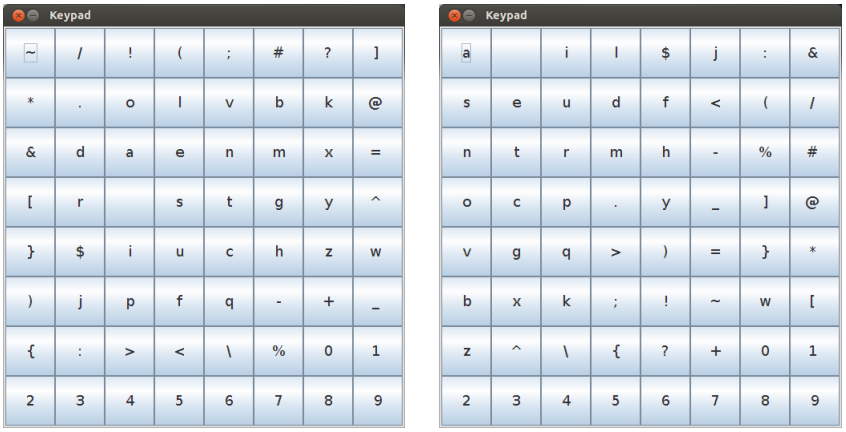
(b) An optimized layout of characters

Fig. 3.2.: Two examples of switch virtual keyboard

3.2.2 Scanning Speed & Entry Errors

The scanning speed of the cursor movement is also an important element when designing a switch virtual keyboard: If the speed is too fast, users do not have enough time to respond correctly; if it is too slow, users have to wait for the cursor to a desired key. Hence, both fast and slow speeds tend to lead users to respond with higher error rates. The keyboard design has to consider the average error rate of entries to ensure it is within an acceptable threshold, in order to satisfy users' needs.

This thesis is using ϵ to denote the average acceptable error rate, which is the average probability that a user can tolerate to make errors when selecting a key. ϵ is an individual threshold defined by a user. Figure 3.3 shows two keyboards with different ϵs. The keyboard on the left has a smaller ϵ, which means the maximum average scanning time to reach each character is larger, while the second keyboard is of bigger ϵ, which means the average scanning time to reach each character is smaller. The average scanning time determined by the cursor duration is the interval the cursor stays on each key on the keyboard, which implies the larger the average scanning time, the longer the cursor duration, and vice versa. In general, the keyboard design involves a trade-off between speed and accuracy.



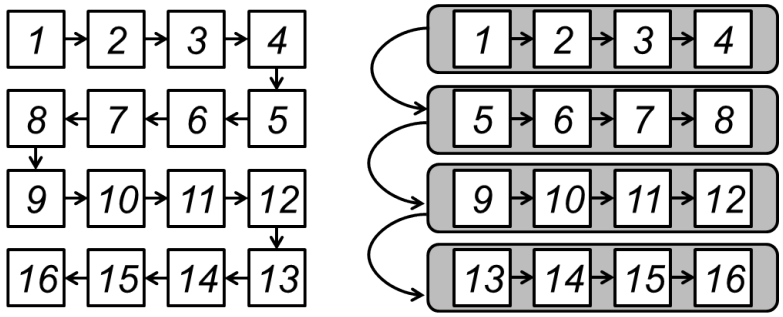
(a) Character layout when $\epsilon = 0.30$ (b) Character layout when $\epsilon = 0.50$

Fig. 3.3.: Two optimized keyboards

3.2.3 Cursor Paths

The method of scanning, or in other words, the cursor path, is also an influential factor. Figure 3.4 shows two different scanning methods for the same physical keyboard. In the first path demonstrated by Figure 3.4(a), the cursor starts at the first key (top left) and then each letter is scanned one by one. The order of scanning is shown by the arrows. At the end of each row, the cursor turns to a new row. The user has to make a selection when the cursor is on the key with a desired character. In the second path that is illustrated by Figure 3.4(a), the cursor first scans across the rows, along the path indicated by the bold arrows. When the user selects a row, the cursor begins to scan across the columns in the selected row, and the user has to select the target key by triggering the switch device again.

There also exist other search paradigms, like binary search, in which the scanning starts between two regions of the keyboard first, then it starts between the subregions of the selected region, and goes deeper until a target key is reached.



(a) Scanning Path "Zig-Zag" (b) Scanning Path "Row to Column"

Fig. 3.4.: Example of two scanning paths

Also, different combinatorially paths can be used in practice. For example, in one kind of path design, the user has to choose a region containing the target key and make a selection, which then leads to the next level. In the sub-level of scanning, the scanning method can vary as described above. In this fashion, a combined scanning method also can be created.

3.2.4 Switch Devices

Switch devices for people of motor difficulties who have special needs, require particular properties. Examples of these devices are demonstrated in Figure 3.5. From left to right, they are sip/puff breeze with headset, button-style adaptive switch, and Emotiv EEG reader that can identify eye blinking and other facial actions.

(a) Sip and Puff Switch

(b) Button-style Adaptive Switch

(c) Emotive EEG reader

Fig. 3.5.: Different switch devices

Another issue in the design is how to integrate virtual keyboards with switch devices, since different switch devices may have different characteristics.

The main body of the sip/puff breeze switch device is a headset to be put on the head, and the end of the tube on the main body is put into the user's mouth. Users trigger the switch signal by sipping or puffing the plastic tube in their mouth. In this way, the pressure sensor inside the tube produces a signal to the computer, and the signal will lead the cursor to take action.

The button-style adaptive switch device works in this way: one end of cable is connected to a computer through a signal transforming interface, and the other end is connected to the main body of the switch device. The main part is a button big enough to be clicked by body parts like finer, elbow, chin, and so on. Users trigger a switch signal by clicking the button, and then the cursor on the virtual keyboard begins to scan, or enter next scanning area and scan. Due to different structures, different devices will lead to different error rates when guiding the cursor. As such, the corresponding optimization will also depend on which device is being used.

Part I

Computational Modeling in Keyboard Design

A fundamental problem in HCI research is how to effectively apply quantitative approaches to investigate in this topic. In traditional qualitative approaches, some biases are usually unavoidable. For example, how to insure that the samples drawn are representative of the population, or that the measurements of a human being's preference are valid, or that the elements that are paid attention to are vital ones in a design problem.

Behavior Informatics provides a way to answer these questions, since the main idea of applying Behavioral Informatics into HCI problems involves two aspects: First, measuring efficacy and efficiency of the systemic design in a definitive and quantitative way; Second, mining and utilizing information from the behavioral data to assist the design. Based on the analysis of these two aspects, computational models can be established to help find solutions to the questions, and the results can be applied into the design. The following two chapters describe how I analyzed these two aspects in the design of the switch virtual keyboard.

Chapter 4 describes how the measurement of efficacy and efficiency was analyzed in the design of switch virtual keyboard, and based on the analysis, a computational model was developed to compute the optimal solution.

Chapter 5 depicts how this thesis mined the behavioral experimental data of human subjects on simulated scenarios of using switch virtual keyboard, and how this information was utilized to construct the model for prediction in unobserved conditions.

4. KEYBOARD DESIGN AS AN OPTIMIZATION PROBLEM

4.1 Introduction

In considering the design of a keyboard, the speed of typing, naturally will be treated as an important characteristic. In addition, the error rate people make in typing, also has a significant impact on usage of a keyboard: people are not willing to make too many mistakes in a typing task. Also, traditionally some keys are grouped together, like numbers and brackets. Violations of this rule, together with the term of speed and error rate, can form a standard cost function. All of these terms are measurable: speed can be measured as the average time to reach each key for a given whole text to type, across all entries; error rate is the average probability of making an incorrect typing across all entries; and violations is the Manhattan distance of each pairs of keys, or each groups of keys, which are supposed to be stay next to each other (Francis and Johnson, 2011).

In a pilot study, it was discovered that the violation term actually plays a very insignificant role in the keyboard design in practice, compared with the speed and error rate terms, given that the number keys are fixed. Therefore, the model was constructed without considering this term.

The cost terms is denoted by C_t and C_e, where C_t and C_e are cost of time and cost of error rate respectively. Then I can give a quantitative definition of the design problem that is faced. I can treat the design problem as an optimization problem: try to minimize C_t as an objective function across all possible designs (with C_e expressed in the constraints). Here all possible designs can be reinterpreted in a quantitative way as well, with the definition of problems, making up the domain of feasible solutions.

4.2 Related Work

Previous work (Francis and Johnson, 2011) solved the optimization problem with a hill climbing algorithm. Although effective, this algorithm was computationally intensive, and was not guaranteed to produce a global optimal solution. A major advance in current work of this thesis is to cast the optimization as a mixed integer programming problem. Standard software packages to solve such problems are fast and guaranteed to produce a global optimal solution, if it exists.

4.3 Mixed Integer Programming

4.3.1 Introduction

Mixed Integer Programming (MIP, or MILP) is such a problem: If only some, but not all of the unknown variables in a linear optimization problem are required to be integers, then the problem is called a mixed integer programming problem. A formal definition is as follows:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n \times \mathbb{R}^p \end{aligned} \tag{4.1}$$

A standard way of solving MIP problems is by using the Gurobi Optimizer. It can be integrated with multiple programming languages, including Python, Java, C, and MATLAB. It is widely respected in both commercial and academic areas, since it won in public benchmark tests versus other solvers. Also, in the pilot research, it performed quite well.

4.3.2 Mathematical Representation of Switch Virtual Keyboard Design

In the previous research, (Francis and Johnson, 2011) proposed three major approaches to improve switch keyboard usability: one is to modify the arrangement of

keys on the keyboard so that commonly used items are located at early points in the cursor movement cycle; second is to adjust the cursor scanning speed to satisfy the speed-accuracy trade-off; the third one is to vary the path of the cursor. I refer to the key arrangement problem here as the assignment of keys problem (AKP). The optimization problem, as described above, is to minimize the time cost as an objective function under constraints, when given frequency of characters of a typing text. An instance of the AKP consists of an integer set $\mathcal{I} = \{1, 2, 3, ..., 64\}$ of characters, an integer set of $\mathcal{J} = \{1, 2, 3, ..., 8\}$ of rows, and an integer set $\mathcal{K} = \{1, 2, 3, ..., 8\}$ of columns. Since numbers are traditionally grouped together in keyboard layout designs, and their frequencies in the chosen text are almost equal to zero, I put them in the last row of the keyboard, and the last few columns of the second to last row. These positions can be formulated as a set

$$\begin{aligned}\mathcal{S} = \{&(55, 7, 7), (56, 7, 8), (57, 8, 1), (58, 8, 2), (59, 8, 3),\\ &(60, 8, 4), (61, 8, 5), (62, 8, 6), (63, 8, 7), (64, 8, 8)\}.\end{aligned}$$

And this constraint can be easily modified for other applications.

4.3.3 MIP Formulation

I define the following decision variables:

- $C_t = \frac{D}{\sum\limits_{i\in\mathcal{I}} f_i} \sum\limits_{i\in\mathcal{I}} \sum\limits_{j\in\mathcal{J}} \sum\limits_{k\in\mathcal{K}} f_i x_{ijk} t_{jk}$ is the average cost of the input time;
- $C_e = \frac{1}{\sum\limits_{i\in\mathcal{I}} f_i} \sum\limits_{i\in\mathcal{I}} \sum\limits_{j\in\mathcal{J}} \sum\limits_{\in\mathcal{K}} f_i p_{jk}(D) x_{ijk}$ is the average cost of error rate;
- x_{ijk} is equal to 1 if the key i is located on the keyboard on row j and column k, and 0 otherwise (It is also possible to extend x_{ijk} to x_{iljk}, if beyond rows and columns, blocks are also used, as well as other situations);

Definition of C_t and C_e ensures the time cost and error cost are equal to the actual time span and error probability average of the entire keyboard respectively. In the

definition, f_i is the frequency of the ith character in a given text corpus, t_{jk} that is variable in different cursor scanning path is the scanning steps required to reach the character on the keyboard with position indicator j, k, and $p_{jk}(D)$ is the probability of making an error in the j, k position, given a cursor duration D.

The problem can be modeled as follows:

$$\text{minimize} \quad C_t \tag{4.2a}$$

subject to

$$\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk} = 1 \qquad \text{for } i \in \mathcal{I} \tag{4.2b}$$

$$\sum_{i\in\mathcal{I}} x_{ijk} = 1 \qquad \text{for } j \in \mathcal{J}; k \in \mathcal{K} \tag{4.2c}$$

$$x_{ijk} = 1 \qquad \text{for } (i, j, k) \in \mathcal{S}; \tag{4.2d}$$

$$C_e \leq \epsilon; \tag{4.2e}$$

$$x_{ijk} \in \{0, 1\} \qquad \text{for } i \in \mathcal{I}; j \in \mathcal{J}; k \in \mathcal{K} \tag{4.2f}$$

$$C_t \geq 0 \tag{4.2g}$$

$$C_e \geq 0 \tag{4.2h}$$

The objective function is (4.2a). It satisfies following constraints: Constraint (4.2b) ensures every key can only be placed on one particular position, and Constraint 4.2c ensures that every position of the keyboard layout can only contain one particular key, due to the fact that every position of the keyboard can only contain a single character, and every character can be only located in one position, justified by the cursor path in Figure 3.4(b). Constraint 4.2d pre-locates the keys of numbers on fixed position, insuring that numbers from 0 to 9, corresponding to the mappings of key 55 to key 64 are fixed on the positions from the row 7, column 7 to row 8, column 8 of the keyboard, with the set $\mathcal{S}$. Constraint 4.2e ensures the average error rate is lower than an given acceptable error rate ϵ, defined by the user. Constraints 4.2f restrain the possible values of x_{ijk}. Finally 4.2g and 4.2h are variable-type constraints.

Note that All linear programming formulation can be transformed into the standard form as in 4.1 as follows (Bertsimas and Tsitsiklis, 1997):

Let $m = |\mathcal{I}|, n = |\mathcal{J}| = |\mathcal{K}|, q = |\mathcal{S}|$ and suppose $\mathcal{S} = \{(i_1, j_1, k_1), \ldots, (i_q, j_q, k_q)\}$. Then let $\mathbf{x} = (x'_{ijk}s, C_e, u, C_t)$, which is a $(mn^2 + 3) \times 1$ vector, where u is a slack variable to satisfy 4.2e. Also let $\mathbf{c} = (0, 0, \ldots, 0, 1)$ be the corresponding coefficient vector of $\mathbf{x}$ and let A be a $(m + n^2 + q + 1) \times (mn^2 + 3)$ matrix, in which

$$A_{h\ell} = \begin{cases} 1 & \text{for } 1 \leq h \leq m, n^2(h-1) \leq \ell \leq n^2 h; \\ 1 & \text{for } m+1 \leq h \leq m+n^2, \ell = 1, n^2+1, 2n^2+1, \ldots, (m-1)n^2+1; \\ 1 & \text{for } (h, \ell) = (m+n^2+r, (i_r-1)n^2 + (j_r-1)n + k_r) \text{ where } (i_r, j_r, k_r) \in \mathcal{S}, r = 1, \ldots, q; \\ 1 & \text{for } h = m+n^2+q+1, \ell = mn^2+1, mn^2+2; \\ 0 & \text{Otherwise} \end{cases} \tag{4.3}$$

and let $\mathbf{b} = (1, \ldots, 1, \epsilon)$ be a $1 \times (m + n^2 + q + 1)$ vector, then formulation 4.2 can be written as in 4.1.

4.4 Conclusion

The AKP can solve based on the above well constructed MIP model by using a Gurobi Optimizer package in python with standard techniques that guarantee a globally optimal solution (if it exists).

A comparison test was run between (Francis and Johnson, 2011,)'s hill climbing algorithm, and the above proposed MIP formulation, on the same data set, with the same $p_{jk}(D)$ used in their research. Results of the generated keyboard layouts are shown below in Figure 4.1(a) and Figure 4.1(b). Comparing these two results, it can be noticed the generated layouts are quite similar between these two methods. Differences are mostly across the left-right diagonal, which does not matter, since the time to reach a key in the position j, k is the same as in the position k, j, taking into account how the cursor moves among rows and columns (like in 3.4(b)): Because scanning will always take j steps to scan the rows, and k steps to scan the columns,

to reach the target key. Therefore, the total amount of steps to reach the target key are always $t_{jk} = j + k$.

	e	i	r	p	.	y	?
a	s	t	d	g	<	+	w
n	u	o	’	j	z	]	\
l	c	v	h	-	∗	}	;
m	f	b	k	#	=	∼	@
q	x	:	)	%	&	!	{
>	[	(	∧	_	/	0	1
2	3	4	5	6	7	8	9

(a) Keyboard layout generated by hill climbing.

	a	i	r	$	.	-	]
e	s	t	c	g	>	)	(
n	u	o	m	h	:	_	#
l	d	v	b	y	\	&	!
p	f	j	<	@	/	{	w
q	x	z	?	}	∧	=	[
k	∗	∼	;	%	+	0	1
2	3	4	5	6	7	8	9

(b) Keyboard layout generated by MIP.

Fig. 4.1.: Example of two optimized keyboards

Nevertheless, the running time of the current MIP formulation with Gurobi solver, is much faster than the hill climbing algorithm. The hill climbing algorithm takes approximately three hours to generate the results on a desktop, while the MIP formulation with Gurobi solver takes less than three seconds on a standard laptop. Also, the MIP formulation provides an exact solution (if there exists one), while hill climbing algorithm only yields a local feasible solution that may not be global optimal.

However, although the MIP model is constructed with most necessary parameters fully identified, one important parameter remains unknown: the terms $p_{jk}(D)$, which is the probability of making a mistake when typing in each position, given a duration of the scanning cursor. The next chapter discusses this problem.

5. ERROR RATE AS A DATA MINING PROBLEM

5.1 Introduction

To solve the AKP as an optimization problem, the probability of making an typing error $p_{jk}(D)$ on each position of the keyboard, is needed. This section tries to integrate known information and construct a model to predict those probabilities, hence all necessary variables in the MIP model can be fully identified.

Hitting a key in the original 8×8 path design keyboard layout, consists of two consecutive processes: First the user needs to select a target row when the cursor is scanning row by row; and then the user has to select the target column when the cursor is scanning all the keys in the selected row. Hereby, it can be assumed that correctly hitting a key is two independent events of these two processes. Hence, the basic model of the hit response is:

$$\pi_{jk}(D) = \pi_j(D) \times \pi_k(D), \tag{5.1}$$

where $\pi_j(D)$ is the probability of hitting the row correctly and $\pi_k(D)$ is the probability of hitting the column correctly. Now finding out the pattern of $\pi_j(D)$ and $\pi_k(D)$ based on this information is the current problem. Assuming the scanning direction does not affect the hit response, then j and k can be merely regarded as two position marks, horizontally and vertically, in two consecutive scans. Hence, the error rate $p_{jk}(D)$ can be formulated as

$$p_{jk}(D) = 1 - \pi_{jk}(D). \tag{5.2}$$

5.2 HCI Experiments

In order to obtain data containing information of making an error response in different positions, while the cursor is scanning the keyboard, HCI experiments were designed to gather data from real users in simulated scenarios.

5.2.1 Typing Program

A typing program was written in the JAVA language. The text window of the typing program is shown in Figure 5.1 below. The upper part of the text window is used to indicate how much time has elapsed. The white area with text inside it is to illustrate text that participants in the experiments are supposed to type with a virtual keyboard. The text shown in this area is always a famous quotation, with the author's name at the end. If the participant's response is correct, the character will be highlighted in green; if it is a early response, or a miss, the corresponding character will be highlighted in red. In Figure 5.1, the red colors indicate many errors among a few (green) correct typings. 311 famous quotations were included in the text corpus.

Another important component of the JAVA typing program is the virtual keyboard, as shown in Figure 5.2. The keyboard is affiliated to the text window. Key arrangement of the keyboard was designed by using the MIP formulation with the error rate model and the text corpus in (Francis and Johnson, 2011,)'s study, by setting the acceptable error rate threshold ϵ as 0.6. The cursor moves first along the row and then column path, as described in Figure 3.4(b). These details should not affect the probability measurements, but were an effort to have participants use a reasonable keyboard.

The program recorded the behavioral data during the typing experiments, including information about which row the participant first selected (including a miss) and then which character the participant selected (including a miss as well). The typing program tracked the user's typing history for every character, from which the early response rate, the hit response rate, and the miss response rate of a given scanning

speed, for every row selection and every column selection given a row, can be calculated. That information would then drive the development of a model of error rate (or entry accuracy).

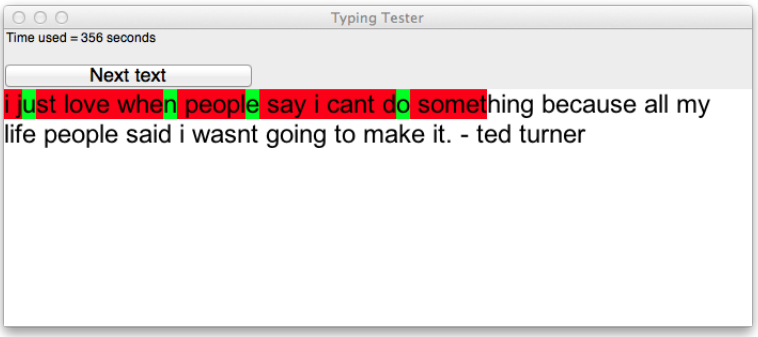


Fig. 5.1.: Text window of the typing program

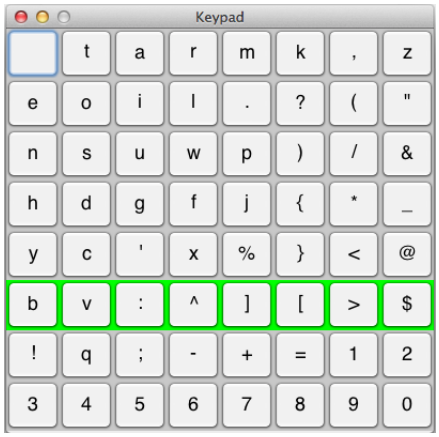


Fig. 5.2.: Keyboard in use

5.2.2 Apparatus

Stimuli were presented on an iMac LCD (Liquid Cristal Display) with 1920 x 1200 pixels, running the Mac OS X v10.5 Leopard operating system. Two switch devices, including the button-style adaptive switch and the sip/puff breeze switch with headset (Figure 3.5(b) and Figure 3.5(a)), were used. I was trying to simulate the conditions when people with motor control difficulties restrained only to their fingers by using button-style adaptive switch and restrained only to the head by using the sip/puff breeze switch.

5.2.3 Subjects

I recruited nine subjects by posting fliers. Two of the them are female. Four of them are undergraduate students and the others are graduate students. Their ages range between 18 and 30 years old. Seven of the subjects participated in the experiments using button-style adaptive switch and four of them participated in the experiments using sip/puff breeze switch with headset (two of them participated both experiments). The subjects have normal or corrected to normal vision and all of them are right-handed. The subjects were paid a base of $5 for each a one-hour session and had an opportunity to earn an additional amount (approximately $35) based on their performance in a session.

5.2.4 Method

The subjects were required to type in their account information in a login panel to start or continue their experiments (They signed up for their accounts in the first session). After login, they clicked the "Next text" button to initiate a new session. At the beginning of each trial, the subject started the cursor scanning by sending a signal to the program, by triggering the switch device. The cursor first went through all rows one by one, and then columns. The participant needed to trigger the switch

(by clicking the button, or puffing/sipping the Breeze), when the cursor was on the row containing the target key, and then again trigger the switch when the cursor was on the target key. If the switch was triggered before the cursor reached the target key, it was a early response error; if the switch was triggered after the cursor left the target key, or no response was made, it was a miss error; if the switch was triggered during the duration that the cursor was on the target key, it was a correct response (hit). For every character typed in, there were two consecutive responses, and the system recorded the row and column that were selected (including the miss response, in which the row or column is "null").

Across blocked sessions, there were five different cursor durations, 0.200 second, 0.175 second, 0.150 second, 0.125 second and 0.100 second Every subject participated in all five different conditions, while one of these 5 conditions were assigned to each sessions. The duration decreased gradually for sessions, such that the scanning speed increased. A session roughly lasted for one hour and the full set of experiments sessions took approximately 25 hours. In each session, there were roughly an average of 5 to 6 quotes. Usually, a subject attended the experiments one hour per day. Before formal experiments, subjects were required to practice for one hour, to get themselves familiar with the keyboard layout and the scanning speed. Subjects could take a rest at any time during each session whenever they wanted, typically for around 5 minutes.

5.2.5 Analysis

The data gathered from the experiments contains several attributes and responses. Since it was assumed the directions of cursor movement have no effect on the response (whether the cursor is moving from left to right, or from top to bottom), the attributes (independent variables) that influenced the response are position marks (PM, row number or column number), cursor duration and their cross effect. Because participants were to find the target before starting the cursor, visual search was not included in the task; and Fitt's law is not applicable for such situation, in

that no real movement of body part was involved, it is believed that the cross effect of PM and cursor duration, which can be interpreted as elapsed time to reach a target key, had a dominant influence on the responses.

This thesis used the data gathered from the above experiments, to construct a model to predict the error rate, given the row number, column number and the cursor duration, which are combined to form the time to reach the target. For example, the time to reach a key in row i, column j, given duration D, is $(i+j) \times D$. The modeled error rate can then be introduced back to the MIP optimization model, to generate an optimized character layout. The following sections describing the error rate model.

5.3 Bayesian Logistic Regression Model

Logistic regression is widely used to model the outcomes of a categorical dependent variables that can only take two different discrete values. It is popular in economics to predict the discrete nominal output of an individual's behavior. This model is introduced here because the output of every trial in the experiments is binary. In logistic regression, the linear component is the logit transform–the natural logarithm of the odds that some event will happen.

5.3.1 Binomial Logistic Regression

Suppose a random variable Z can take one of two possible values. Given a sample size M, where each observation is independent, a vector of M binomial random variables that are the results (label, response) of each trial can be formed, demoted as $\mathbf{Z}$. Value 1 is used to indicate a success trial, and 0 to suggest a failure trial. A design matrix is also constructed by aggregating the data such that each row represents one distinct combination of the independent variables. Suppose there are N such different rows, which are referred as "populations" (Czepiel, 2014), represent the total number of populations, and $\mathbf{n}$ be a column vector with element n_i representing

the total number of observations in the population i ($i \in N, \sum_{i=1}^{N} n_i = M$). Another column vector $\mathbf{Y}$ of length N is created, in which each Y_i is a random variable representing the number of successes of $\mathbf{Z}$ for population i. Let the realization $\mathbf{y}$ contain elements y_i representing the observed counts of successes for each population. Let $\boldsymbol{\pi}$ be a column vector of length N with elements $\pi_i = P(Z_i = 1/i)$, indicating the probability of success for any given observation in the ith population.

Hence, a design matrix $\mathbf{X}$ can be formed, with N rows and $K+1$ columns, where K is the number of independent variables specified in the model. The first element of each row is always 1, together with the first element in the coefficient vector to represent the intercept of the linear component. The coefficient vector that is of length $K+1$, contains the parameters of the model, denoted by $\boldsymbol{\beta}$.

The logistic regression model is equal to the *logit* transformation that is the log-odds of the probability of a success, to the linear component. The transformation can be written as

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} = \sum_{k=0}^{K} x_{ik}\beta_k, \quad i = 1, 2, \ldots, N \tag{5.3}$$

5.3.2 Parameter Estimation

The objective of constructing the logistic regression model is to estimate the $K+1$ unknown parameters $\beta_0, \ldots, \beta_k$ in $\boldsymbol{\beta}$. The next paragraphs describe the procedures of estimating $\boldsymbol{\beta}$ in the Bayesian framework.

Likelihood

First, the likelihood $L(\boldsymbol{\beta}|\mathbf{y}) = f(\mathbf{y}|\boldsymbol{\beta})$ can be derived as the probability distribution of the dependent variable, $\mathbf{Y}$. In that each y_i represents a binomial distribution of the dependent variable, the joint probability density function of $\mathbf{Y}$ is:

$$f(\mathbf{y}|\boldsymbol{\beta}) = \prod_{i=1}^{N} \frac{n_i!}{y_i(n_i - y_i)} \pi_i^{y_i}(1-\pi_i)^{n_i - y_i}, \tag{5.4}$$

where the probability of y_i successes is $\pi_i^{y_i}$, and the probability of $n_i - y_i$ failures is $(1-\pi_i)^{n_i-y_i}$, with $\binom{n_i}{y_i}$ different ways of arrangements, for each population i. Therefore,

$$L(\boldsymbol{\beta}|\mathbf{y}) = f(\mathbf{y}|\boldsymbol{\beta}) = C\prod_{i=1}^{N}\pi_i^{y_i}(1-\pi_i)^{n_i-y_i}, \tag{5.5}$$

where C is the constant $\dfrac{n_i!}{y_i(n_i-y_i)}$. Since from Equation 5.3, it is easily derived $\pi_i = \dfrac{\exp(\mathbf{x}^{\mathrm{T}}\boldsymbol{\beta})}{1+\exp(\mathbf{x}^{\mathrm{T}}\boldsymbol{\beta})}$, therefore

$$L(\boldsymbol{\beta}|\mathbf{y}) \propto \prod_{i=1}^{N}\left\{\exp(\sum_{k=0}^{K}x_{ik}\beta_k)\right\}^{y_i}\left\{1+\exp(\sum_{k=0}^{K}x_{ik}\beta_k)\right\}^{-n_i}. \tag{5.6}$$

Consequently, the log-likelihood should be

$$l(\boldsymbol{\beta}|y) \propto \sum_{i=1}^{N}y_i\left(\sum_{k=0}^{K}x_{ik\beta_k}\right) - \sum_{i=1}^{N}n_i\log\left(1+\exp(\sum_{k=0}^{K}x_{ik}\beta_k)\right). \tag{5.7}$$

However, a closed form of solution to $\boldsymbol{\beta}$ cannot be obtained by using regular maximum likelihood estimation (MLE) method. Thus, Maximum A Posterior estimation in the Bayesian framework is applied.

Bayesian Analysis

The Bayesian inference for the binomial logit model proceeds with the posterior density for the model parameters proportional to their prior density times the likelihood. (Polson et al., 2013,) developed a simple Gibbs sampler for the Bayesian logistic regression model by carefully constructing the Polya-Gamma family distribution. By adapting data-augmentation strategies, they introduced a Polya-Gamma random variable as a latent variable. A random variable $\Omega \sim PG(b,c)$ has a Polya-Gamma distribution if

$$\Omega \overset{D}{=} \frac{1}{2\pi^2}\sum_{k=1}^{\infty}\frac{g_k(b)}{(k-1/2)^2+c^2/(4\pi^2)}, \tag{5.8}$$

where $g_k(b) \sim Ga(b, 1)$ are independent gamma random variables. Subsequently, the likelihood of each population can be written as

$$L_i(\boldsymbol{\beta}|y_i) = \frac{\left(\exp\left\{\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right\}\right)^{y_i}}{\left(1+\exp\left\{\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right\}\right)^{n_i}} \propto \exp\left(k_i\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)\int_0^{\infty}\exp\left\{-\omega_i\left(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)^2/2\right\}p(\omega_i|n_i,0), \tag{5.9}$$

where $k_i = y_i - n_i/2$, $p(\omega_i|n_i, 0)$ is the density of a Polya-Gamma random variable with parameters $(n_i, 0)$. As a result, the posterior of the coefficient vector $\boldsymbol{\beta}$ is fully identified, given $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)$:

$$p(\boldsymbol{\beta}|\boldsymbol{\omega}, \mathbf{y}) \propto \pi(\boldsymbol{\beta})\prod_{i=1}^{N} L_i(\boldsymbol{\beta}|w_i) \propto \pi(\boldsymbol{\beta})\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}\Omega(\mathbf{z}-\mathbf{X}\boldsymbol{\beta})\right\}, \tag{5.10}$$

where $\mathbf{z} = (k_1/\omega_1, \ldots, k_N/\omega_N)$, and $\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_N)$. Hence, a Gibbs sampler becomes possible.

5.3.3 Modeling of Data

For the HCI experimental data, the responses were first categorized into two types: correct and incorrect responses (Because in the modeling, I found that two class classification outperformed three class classification). The correct responses are those that were made during the duration, when the moving cursor was on the targeted PM. In this Bayesian Logistic Regression model, I combined the previously defined two categories: early response and miss, together as the incorrect response class. Based on the above analysis of the attributes, I constructed the design matrix X with 40 rows and 2 columns. Values in the first column are all "1"s, denoted by x_0, matching to the intercept coefficient in the coefficient vector $\boldsymbol{\beta}$. After combining the data, I had $5 \times 8 = 40$ different combinations across PM and cursor duration, since there were 5 different cursor durations and 8 different PMs. Initially I assumed both of these two features, PM and cursor duration should be used in the logistic regression. But later it was found, by combining these two features into the crossproduct of them, as a dimension reduction, can yield a better performance in the model. Also, the

crossproduct has a practical interpretation: they are the "time" attributes to reach targets. Hence, besides the intercept, I just used one feature in the model, the time needed to reach a target. Therefore, the second column consists of these 40 different values of the "time" attribute, denoted by x_1. For these 40 populations, there is n_i as the total number of trials in each, and y_i as number of successes, to indicate the number of correct responses in population i. In the experimental data, $K = 1$ and $N = 40$. Therefore, the design matrix in the Bayesian Logistic Regression model is established. This model was first applied to data from experiments using the adaptive button switch device in this section, and later applied to the data from experiments using the sip-puff switch device, because it is believed that different devices have different parameters in the model.

In order to estimate the coefficient vector $\boldsymbol{\beta} = (\beta_0, \beta_1)$, this thesis used the R package "BayesLogit" developed by (Polson et al., 2013,) with his proposed data-augmentation method that constructed the Gibbs Sampler. I set up the simulation to learn the coefficient vector $\boldsymbol{\beta}$ as follows: the simulation ran a single Monte Carlo Markov Chain to sample from the posterior of $\boldsymbol{\beta}$. The simulation was set to run for 51000 iterations, and the first 1000 iterations was discarded as the "burn-in" period. The prior of *bbeta* is a improper flat prior $\boldsymbol{\beta} \sim N(\mathbf{0}, Precision = \mathbf{0})$.

5.3.4 Results

I first applied the model on the adaptive button device data. For the data, I obtained the following results :

$$\tilde{\boldsymbol{\beta}} = \begin{pmatrix} 1.092375 \\ 2.327665 \end{pmatrix}$$

as the parameters of the Logistic regression model.

The plotting of the chains is in Figure 5.3. The top two figures show the distributions of sampled posteriors of β_0, β_1, in which the means are the point estimate $\tilde{\boldsymbol{\beta}}$ of $\boldsymbol{\beta} = (\beta_0, \beta_1)$. The bottom two figures are samples from the posteriors plotted against iteration numbers (the "Burn-in" period was not included), for the purpose of diagno-

sis of the chains. As the figures show, the chains are in the form of thick "caterpillars", which indicates convergence. This ensures the reliability of the estimates according to this method.

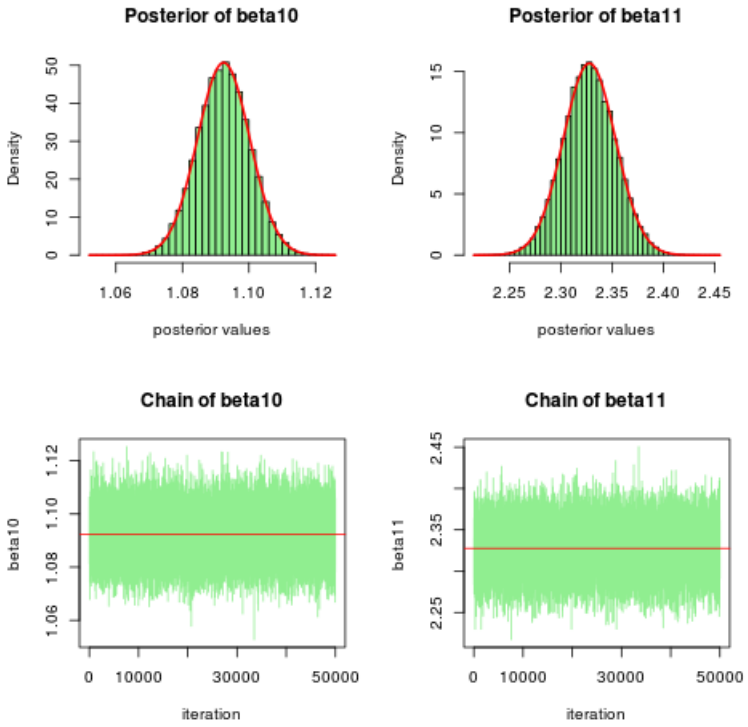


Fig. 5.3.: Bayesian simulation results of $\boldsymbol{\beta}$

The dots plotting of predicted hit response probability against real data hit response probability are shown in Figure 5.4. The larger a plotted point is, the more data are aggregated at that value. Roughly, some of the plots are in the diagonal, or close to the diagonal, which means the model generated data points partially match the observed data, since the model is a generative probabilistic model.

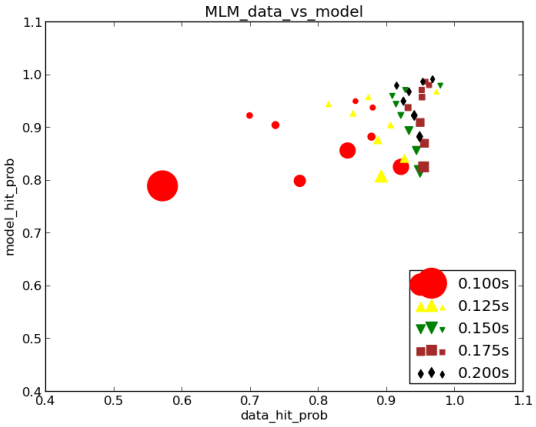


Fig. 5.4.: Model prediction vs. real data correlation

5.4 Refinement of Model

Since the reaction time for a Olympic Sport Player is 0.109s to 0.210s, as a fastest extrema (Lipps et al., 2011), I speculate the time to reach a target, if less than 0.100s, would not allow users with motor difficulties to react correctly. To reflect this reaction constraint in the modeling, this thesis propose a piecewise function:

$$\pi_k = \begin{cases} \dfrac{\exp(\beta_0 + \beta_1 x_1)}{1 + \exp(\beta_0 + \beta_1 x_1)} & \text{if } x_1 \geq 0.1; \\ 0 & \text{Otherwise} \end{cases} \tag{5.11}$$

That is, if the time to reach a target is less than 0.100s, I force the probability of making a correct response to 0 (probability of making ab error response is 1), for the purpose of adjusting human reaction. This adjustment ensures the model capture the information in real scenario. I plug this into 5.1 and 5.2 to obtain $p_{jk}(D)$.

After this refinement, I plotted the relationship between the model generate data and the empirical data again. Figure 5.5 shows the prediction from the Bayesian Logistic Regression model and the real data. This fit captures the main information of the data, but it is not robust to all the data points.

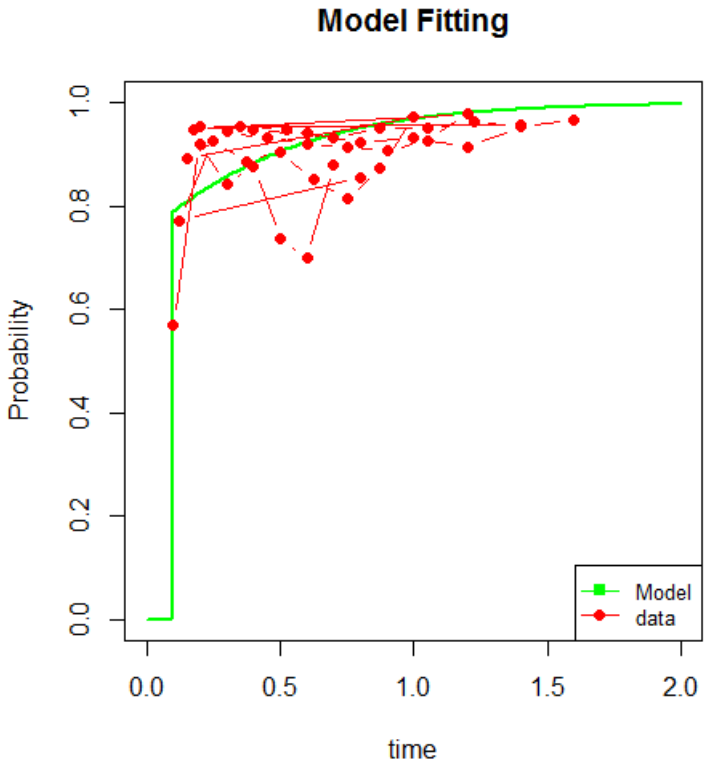


Fig. 5.5.: Model prediction vs. real data on a scale

Part II

Performance of Keyboards

This part of the thesis investigates and discusses how different design elements will influence the performance of the keyboard, and possible alternatives that people may choose.

Chapter 6 mainly discusses the generated optimized keyboards pre-constrained by the cursor movement paths, including $4 \times 4 \times 4$ path, 8×8 path, binary path and "Zig-Zag" path, and makes a comparison based on their performance.

Chapter 7 investigates the optimized keyboards between different switch devices, the adaptive switch button device and the "Sip-Puff" device.

Chapter 8 taps the optimized keyboards for different text, or say, corpus environment, including literature creation and computer language programming (Python codes as an example).

6. DIFFERENT CURSOR PATHS

In this chapter, the performance of virtual keyboards generated by the optimization method among different cursor movement paths is presented, and their characteristics are discussed.

6.1 "Zig-Zag" Design

The "Zig-Zag" design can be regarded as a prototype of other variations, it follows the simplest rule: the cursor just goes through all the keys on the keyboard one by one in a "Zig-Zag" fashion. The cursor first moves along all the keys on the first row horizontally, and then moves to the last key on the second row, and consequently moves along all the keys on the second row in the opposite direction, until it reaches to the first key, and so on. The path of the cursor movement looks like a "Zig-Zag" moving snake, therefore I named this design after that. Figure 6.1 shows the cursor movement in the "Zig-Zag" design path.

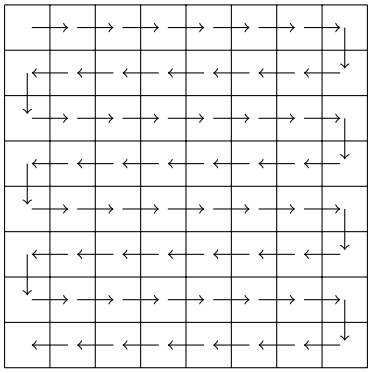


Fig. 6.1.: Cursor path of "Zig-Zag" design

A simple analysis can show that given the same frequency of all characters, the expected steps $E[S]$ of the moving cursor to reach a key is

$$\frac{(1+N)}{2}. \tag{6.1}$$

Hence, step consumption of using this keyboard is high, and the same for the time consumption if duration is fixed. But on the contrast, since on average it takes longer to reach a key (even given a short cursor duration), the error rate is lower.

6.2 8 by 8 Design

The 8×8 path design is a traditional approach to design a switch virtual keyboard, and is commonly adapted by commercial products. Figure 6.2 shows how the scanning cursor moves on a 8×8 keyboard. When the user triggers the switch, the cursor starts to move along the rows vertically. In the short stay (duration) that the cursor is on the target row, the user triggers the switch again, and the cursor starts to move along the target row horizontally. when the cursor is on the target key, the user triggers the switch a third time, and the desired key is selected.

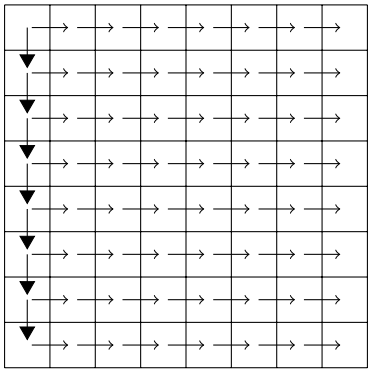


Fig. 6.2.: Cursor path of 8 by 8 keyboard

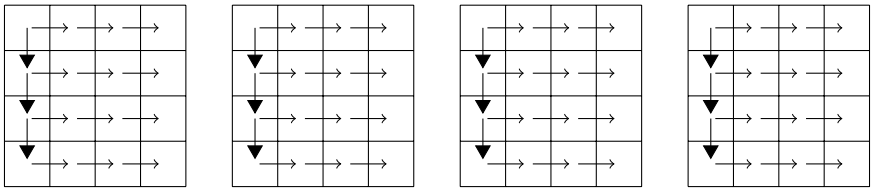
Intuitively, this scanning method is a moderate trade-off between time and errors, because users have to make two selections, and each selection consumes several steps to reach the target. In each selection, the maximum numbers of steps to reach a key, is the twice the square root of the total number of keys, which is $2\sqrt{N}$, given N keys. Assuming all the keys are of the same frequencies, it is trivial to prove that the expected number of steps $E[S]$ of selecting a target key is

$$\sqrt{N} + 1. \tag{6.2}$$

Without considering the rate of errors, this reduces the complexity of the original problem a lot. But since the user needs to select the target twice, the error rate generally increases.

6.3 4 by 4 by 4 Design

The $4 \times 4 \times 4$ path design is a variation of the above 8×8 design. The difference is that the $4 \times 4 \times 4$ keyboard is pre-divided into 4 blocks (smaller keyboards). When the user triggers the switch, the cursor first moves across four blocks; when the cursor is on the stay (duration) of the block that contains target key, the user triggers the switch again to select that block; consequently, within this selected block (smaller keyboard), the user triggers the switch two more times to select the target key, in the same way as the 8×8 design. The separation of the keyboard can be either physical or just movement-wise. The following graphs 6.3 and 6.4 show the $4 \times 4 \times 4$ design, while Figure 6.3 adapts the visually separated design and Figure 6.4 illustrates the visually combined one, but movement-wisely separated. Figure 6.4 uses both the depth of red colour, and the bigger arrows to indicated the order of the cursor movement among blocks (smaller keyboards). In terms of the optimization algorithm, these are identical keyboards. The term 4.2f in the MIP formulation shall be adjusted to x_{iljk} with $l \in \mathcal{L}$ to present the blocks, so as x_s and $\mathcal{S}$.



(a) block 1 (b) block 2 (c) block 3 (d) block 4

Fig. 6.3.: Separated $4 \times 4 \times 4$ design

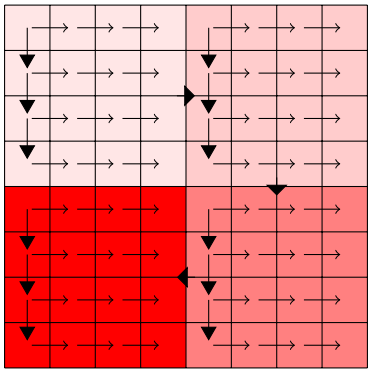


Fig. 6.4.: Non-separated $4 \times 4 \times 4$ design

In this design, users need to make three selection. This $4 \times 4 \times 4 Design$ further decreased the number of steps to reach each key and reduces the complexity of the original problem: Given a keyboard of N keys, and each key of the same frequencies, the expected number of steps $E[S]$ of selecting a target key is

$$\frac{3(\sqrt[3]{N}+1)}{2}.$$

Meanwhile, the time taken to reach a key, has a linear relationship with the steps to reach it, the expected time to reach a key is also decreased, which may cause users more likely to make errors while triggering the switch device multiple times.

6.4 Binary Design

In yet another design, the keys in a keyboard are under a deeper hierarchical structure frame than $4 \times 4 \times 4$ design and the 8×8 design. All the keys are divided into two groups, and each group is divided into two subgroups again, and this division continues. The smallest element of a subgroup is a single key. The structure can be represented by a binary tree. Figure 6.5 shows a binary search tree, which can be used to elucidate the structure of this design.

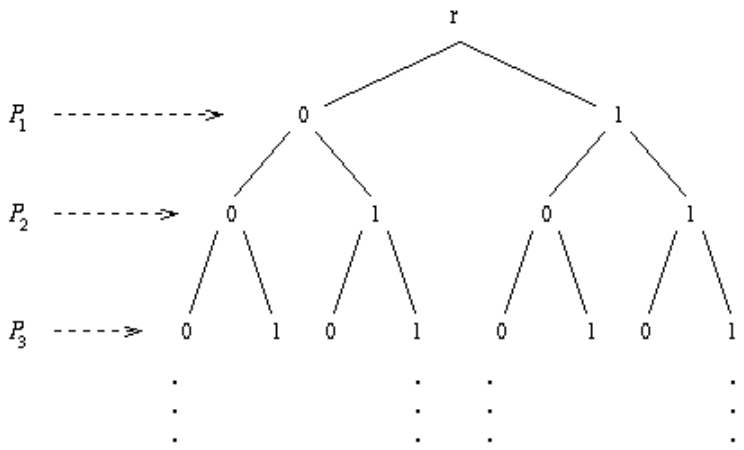


Fig. 6.5.: A binary search tree

To represent the keyboard case, the root "r" is the entire keyboard. Letters "P" with a sub-number represent levels of a keyboard. The sub-number of "P" indicates the order of selection. Half of the keys of one node are always grouped to the left child, and half to the right child. When the cursor starts moving, users need to make a selection when the cursor is going through all the nodes in the same level, namely, choose a subgroup of keys. The leaves of this binary tree represent single keys of the keyboard. Without losing generality, even if N is not of the form 2^{ν}, where $\nu \in \mathcal{Z}^{+}$, it is trivial to prove that a keyboard with N keys needs the user to make $\lceil \log_2 N \rceil$ selections to reach each target key, which is also the depth of the binary tree. On each selection, the step is either "1" or "2", it is easy to verify, the expected number of steps $E[S]$ of selecting a target key is

$$\frac{3\lceil \log_2 N \rceil}{2}.$$

Figure 6.6 shows the graphical structure of the binary design. The cursor moves gradually from the entire keyboard area to the target key. The arrows, together with different intensities of shades of the green color, indicate the order of the cursor movement (from light green to deep green).

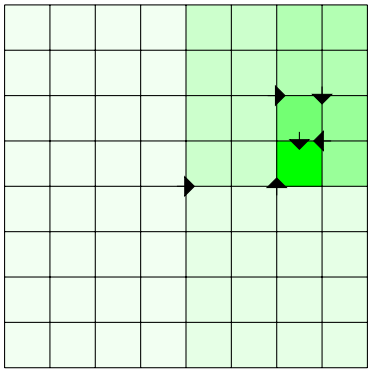


Fig. 6.6.: Example of a binary path

This design reflects a weighted bias on the speed of the speed-error trade off, since the expected steps to reach each key is a function of logarithm, the time consumption drastically decreases; But at the same time, users are more likely to make errors when they trigger the switch to make selections.

6.5 Simulations & Results

This thesis used the quotation text from the experiments as the corpus for simulations, to emulate the literature input environment. I calculated the frequencies of all characters and made a frequency table for simulation purpose. The simulations were set in this way: I first set 9 different acceptable error thresholds,

$$\epsilon \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}.$$

I plugged back the correct response probability model for adaptive button switch device. In order to obtain the smallest cursor duration that satisfies the acceptable entry error rate ϵ for a given text, this thesis employed an exhaustive search for the smallest duration in the range $(0, 1]$, with which the MIP model can yield a feasible solution, by employing a binary search algorithm, for each ϵ, for each path respectively. The algorithm is shown in Algorithm 1. The stopping criterion this algorithm set for the search was that, for feasible durations MIP solver found, $|Duration^{(t)} - Duration^{(t-1)}| < 0.001$. Thus, a search for a given ϵ took at most $\lceil \log_2 \frac{1s}{0.001s} \rceil = 10$ times.

This thesis shows the results of the simulations with three different measures of the optimization, the smallest cursor duration, the average entry time for each input, and the average error rate for each input. They are shown in Figure 6.7(a), 6.7(b), and 6.7(c) respectively.

Algorithm 1 Finding Optimal Duration

$D_{max} \leftarrow 1.0$ {The initial feasible solution of the Duration}
$D_{min} \leftarrow 0.0$ {The initial infeasible solution of the Duration}
while 1 **do**
  $Dur \leftarrow (D_{max} + D_{min})/2$ {Current Duration used in the MIP}
  MIP Procedure {here we run our Mixtured Integer Programing}
  if the model is infeasible **then**
    $D_{min} \leftarrow Dur$
    if $|D_{min} - 1.0| < 0.001$ **then**
      BREAK {No feasible solution exists in this range}
    end if
  else
    $D_{max} \leftarrow Dur$
    if $D_{max} - D_{min} < 0.001$ **then**
      BREAK {Feasible solution found}
    end if
  end if
end while

6.6 Discussion

There does not exist a feasible solution for the binary type keyboard, with an acceptable error threshold $\epsilon = 0.1$. Generally, the patterns on the figures show that the smallest feasible duration drops while ϵ increases. Similarly, the average entry time also drops when ϵ increases. Meanwhile, the average error rate raises while ϵ increases. However, the changing rate of these measures are different along with the change of ϵs. For the binary path, all three measures decreases or increases drastically with ϵs, as implied by the steep step slopes in the graph. The changing rates of all measures of the $4 \times 4 \times 4$ path and the 8×8 path are relatively moderate, while that of the "Zig-Zag" path is very small,in which the line slope is almost flat.

Intuitively, the patterns on the graph are also quite understandable, because the Binary design has the fewest expected steps and the most selections to reach a target key, it is more likely for uses to make errors when triggering switch devices. To satisfy the smaller ϵs, the duration (stay of the cursor on a target) has to be prolonged to ensure a smaller error rate. Furthermore, it is easy to prove that average entry time actually has a linear relationship with the smallest duration, when the path of keyboard is specified, given the fixed corpus character frequency table.

For the "Zig-Zag" keyboard, the MIP solver put those characters that appear infrequently in locations where errors are more likely to occur: the beginning part and the latter part of the keyboard, to shorten the average entry time. It also put characters that occur frequently in the middle of the keyboard to ensure the time consumption shortened and error rate lowered simultaneously. In this fashion, the "Zig-Zag" path optimized the trade-off of the time consumption and error rate.

As shown in the graphs, if when evaluated by average entry time as a measure to evaluate, the "Zig-Zag" keyboard always has the best performance; if average error rate is used as an evaluation measure 6.7(c), when the acceptable error rate ϵ is higher than approximately 0.7 (It means the users are seeking for a fast speed when typing, sacrificing their accuracy), the traditional 8×8 path keyboard outperforms other

keyboards with lower average error rates. All path designs have average error rates the same as the acceptable error rate ϵ, when ϵ is roughly smaller than 0.7.

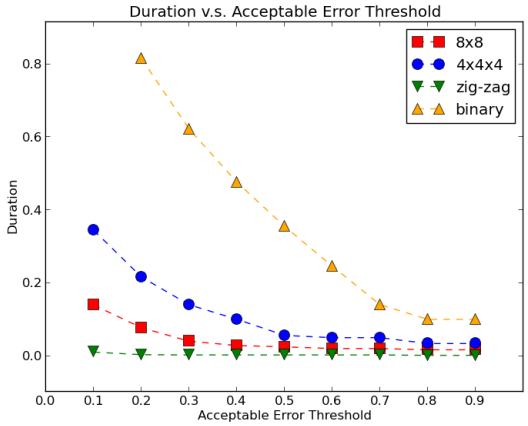
The reason that the $4\times4\times4$ path and the 8×8 path do not have good performance, is that the average steps to reach each key is short, thus error rates that are spread on their keys cannot be lowered effectively. To lower the error rate as possible, the MIP solver put characters of high frequencies on location of the keyboard with more steps. And this increases the time consumption.

In addition, this thesis shows two generated optimized keyboards to illustrate the characteristics of the results: Figure 6.8(a) shows the 8×8 keyboard generated with the smallest feasible duration when $\epsilon = 0.15$, and Figure 6.8(b) shows the "Zig-Zag" keyboard with the smallest feasible duration generated when $\epsilon = 0.15$. For the 8×8 keyboard in Figure 6.8(a), the average entry time is 1.227s, the average error rate is 0.150 and the smallest duration is 0.103s. For the "Zig-Zag" keyboard in Figure 6.8(b), the average entry time is 0.279s, the average error rate is 0.150 and the smallest duration is 0.007s.

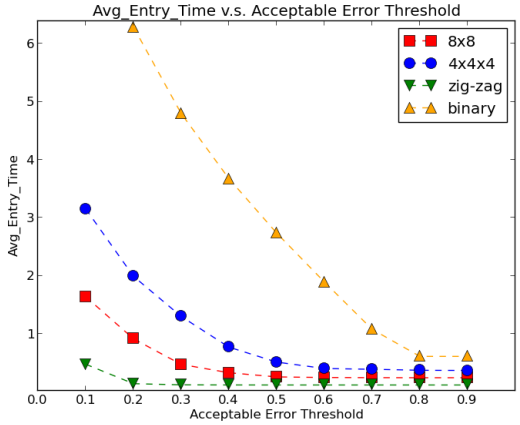
This thesis also shows two generated optimized keyboards of relaxed acceptable error rate ϵ: Figure 6.8(c) shows the 8×8 keyboard generated with the smallest feasible duration when $\epsilon = 0.85$, and Figure 6.8(d) shows the "Zig-Zag" keyboard with the smallest feasible duration generated when $\epsilon = 0.85$. For the 8×8 keyboard in Figure 6.8(c), the average entry time is 0.250s, the average error rate is 0.802 and the smallest duration is 0.017s. For the "Zig-Zag" keyboard in Figure 6.8(d), the average entry time is 0.125s, the average error rate is 0.787 and the smallest duration is 0.002s.

6.7 Conclusion

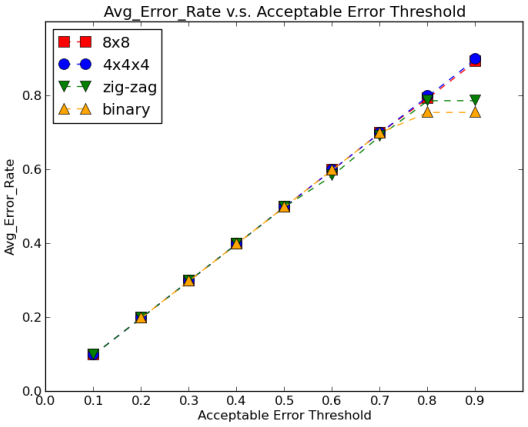
The simulations show that the algorithm can generate keyboards based on both the needs of the user (by specifying the acceptable error threshold ϵ), and the feasibility of the MIP formulation. Different path designs in the algorithm will also lead to different performance of the keyboards, as have been shown, the "Zig-Zag" design appears to be the best choice for the cases I considered.



(a) Smallest cursor duration

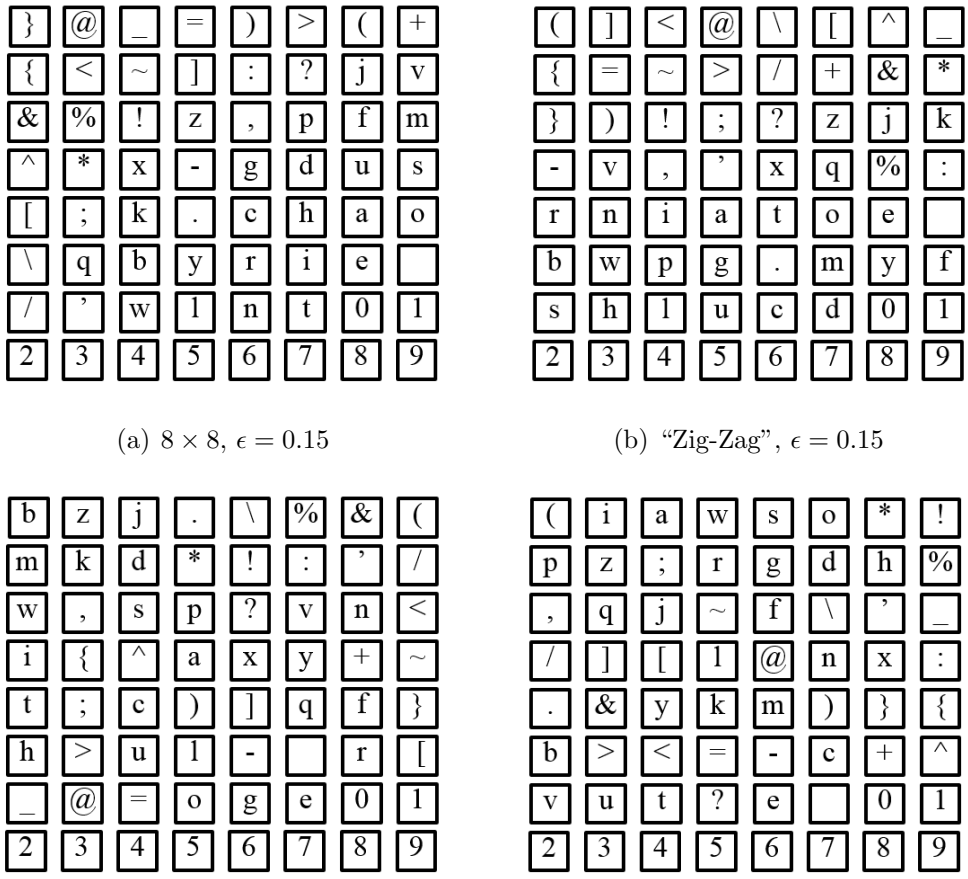


(b) Average entry time



(c) Average error rate

Fig. 6.7.: Simulation results for different paths



(a) 8×8, $\epsilon = 0.15$

(b) “Zig-Zag”, $\epsilon = 0.15$

(c) 8×8, $\epsilon = 0.85$

(d) “Zig-Zag”, $\epsilon = 0.85$

Fig. 6.8.: Examples of generated keyboards for different path designs

7. DIFFERENT DEVICES

In different situations, diverse devices may be required to adjust the needs. An extreme example are the severely damaged "locked-in" patients, who can barely move their bodies parts, except the head or neck. They are cognitively normal, and also have control on their facial muscles to perform simple actions like eye blink, chin movement, puff, sip and so on. This chapter discussed two different devices, one is the normal button switch device, and another is the "sip-puff" headset, which is specially designed for above mentioned severely constrained "locked-in" patients. They can use their mouth to sip or puff the tube to trigger the device.

This chapter illustrates the performance of virtual keyboards attached to different switch devices. Different devices only influence the error rate PM of characters on the keyboard, no matter which corpus is used. This thesis uses quotations in the HCI experiments in different devices to demonstrate the final constructed virtual keyboard.

7.1 Switch Button

The button adaptive switch is shown in Figure **??**.

Fig. 7.1.: Button-style adaptive switch

It is believed that button adaptive switch device is easier for users to use and trigger switch signals. The results of button adaptive switch are shown in the previous chapter (when investigating *different cursor paths* as a design factor, adaptive switch button device were used). In this chapter, I will mainly focus on the performance keyboard attached to "Sip-Puff" headset device.

7.2 "Sip-Puff" Headset

The "Sip-Puff" headset is another switch device, which is shown in Figure **??**.

Fig. 7.2.: Sip and puff switch

Intuitively, the task is more difficult than using the switch button, because the muscle agility in mouth is not as good as that on fingers.

7.3 Logistic Regression Model

The model is the same as the one this thesis constructed above. But I feed into it with the data that were collected in experiments using the "Sip/Puff" headset switch device. The coefficient vector obtained is

$$\tilde{\boldsymbol{\beta}} = \begin{pmatrix} 0.7787903 \\ 2.2024768 \end{pmatrix},$$

as the parameters of the Logistic regression model.

And the plotting of the MCMC chain is in Figure 7.3. The interpretation is similar to that before.

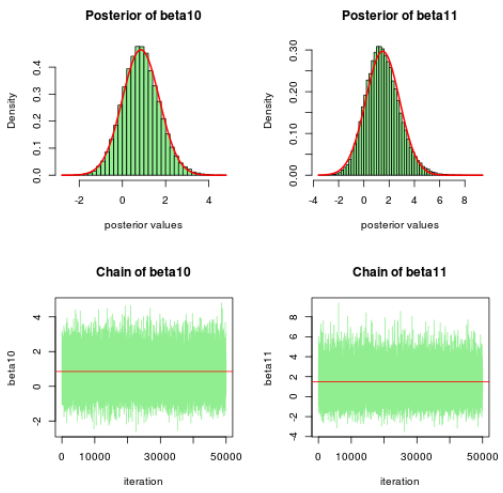


Fig. 7.3.: Bayesian simulation results of $\boldsymbol{\beta}$

The dots plotting of predicted hit response probability against real data are shown below in Figure 7.4. The model predicts dots of high hit probability and some of the dots of low hit probability relatively well, but for those data points with moderate probability, it does not do a good job. One possible reason is that the data contains high noise. The model does not do a good job here.

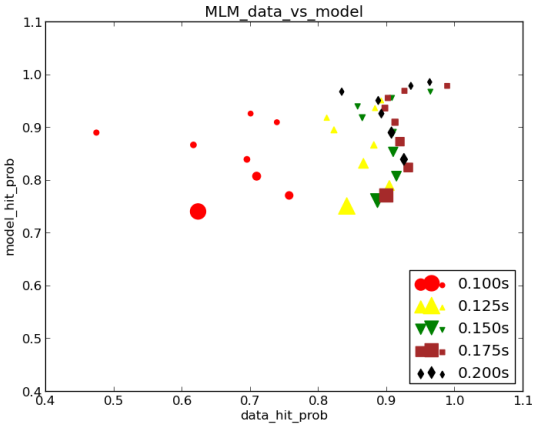


Fig. 7.4.: Model prediction vs. real data correlation

7.4 Simulation Results

I used the same setting as I ran the simulation of different paths when using the button adaptive device, except locally in the MIP formulation 4.2h, I plug back the probability that is calculated from the logistic regression model in "sip-puff" condition, to predict the error in each PM.

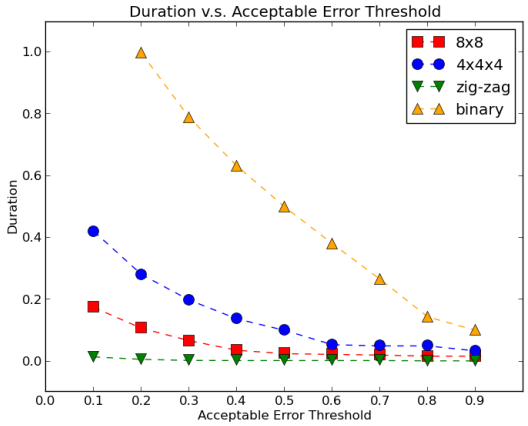
The results of the simulations are shown with three different measures, the smallest cursor duration, the average entry time for each input, and the average error rate for each input, same as previous. They are shown in Figure 7.5(a), 7.5(b), and 7.5(c) respectively. The results almost have the same pattern as that of the button adaptive device.

7.5 Discussion

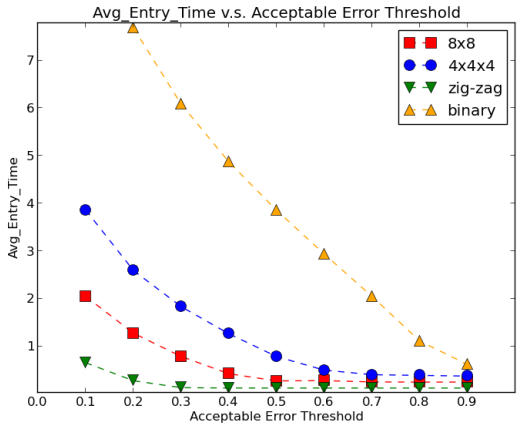
There does not exist a feasible solution for a binary design keyboard with an acceptable error threshold $\epsilon = 0.1$, as in adaptive button situation. Generally, the patterns of different designs of paths are similar to that in the button adaptive switch device. The curve of the "Zig-Zag" path is always below all the others, in smallest cursor duration, average entry time, and average error rate measures. This shows, the "Zig-Zag" path always has the best performance when choosing the "Puff-Sip" device as the switch device.

7.6 Conclusion

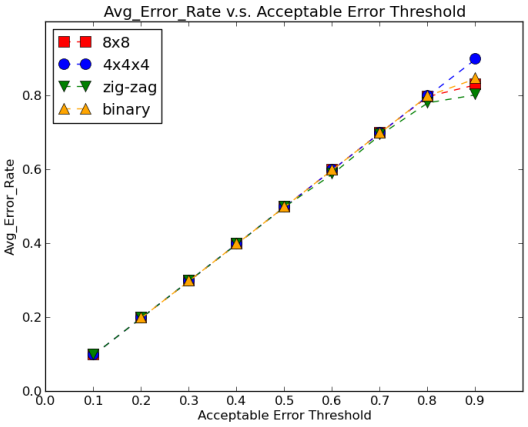
The simulation successfully gave a quantitative analysis of all the four different path designs when using the "Sip-Puff" device. With the results, this thesis concludes that the "Zig-Zag" keyboard is the best among the four path designs. But due to the fact that logistic regression model does not fit the data pretty well, if a user is able to use both the button adaptive switch device and the "sip-puff" one, it is suggested to choose the former, due to the predicted error rate on each PM is more reliable.



(a) Smallest cursor duration



(b) Average entry time



(c) Average Error Rate

Fig. 7.5.: Simulation results of different paths for "Sip-Puff" Headset

8. DIFFERENT TEXTS

In the real world, some groups of people with motor control difficulties are more focused on literature composition tasks. They are earning their living by writing novels, reviews, poems or other literary forms. Some others are working on technical input tasks, like website design and programming. Based on different tasks, the corpus text that people working on, consist of different frequencies of characters. When constructing customized keyboards, I shall take into consideration different texts. In this chapter, I present two examples of different corpus, one is the quotations I used in the HCI experiment as literature corpus; another is the python programming codes I used in this project.

8.1 Literature Corpus (Quotations)

This thesis took the quotations as representations of the literature corpus. When people are working on literature composition, some characters appear more frequently than others. For example, letters like “e” are frequently used, because of the intensive usage of “be” verbs, like “are”, “were” and “be”.

The following table 8.1 shows characters included in this literature text environment.

Table 8.1.: Character frequency of literature environment

Char	Freq
‘ ’	5346
‘a’	1764
‘b’	370
‘c’	800
‘d’	758
‘e’	2876
‘f’	521
‘g’	503
‘h’	1138
‘i’	1803
‘j’	61
‘k’	223
‘l’	1089
‘m’	590
‘n’	1680
‘o’	1995

‘p’	415
‘q’	19
‘r’	1385
‘s’	1655
‘t’	2076
‘u’	827
‘v’	306
‘w’	552
‘x’	23
‘y’	620
‘z’	29
‘!’	4
‘@’	0
‘,’	165
‘.’	434
‘%’	2

‘:’	9
‘∧’	0
‘&’	0
‘*’	0
‘(’	0
‘)’	0
‘;’	7
‘”’	108
‘?’	15
‘+’	0
‘=’	0
‘<’	0
‘>’	0
‘/’	0
‘\’	0
‘{’	0

‘}’	0
‘[’	0
‘]’	0
‘-’	325
‘_’	0
‘∼’	0
‘0’	5
‘1’	1
‘2’	2
‘3’	0
‘4’	0
‘5’	0
‘6’	0
‘7’	0
‘8’	1
‘9’	1

8.2 Programming Language (Python Codes)

When people are working on programming tasks, their text environment are different. In my study, I used the python codes to run simulations within this keyboard project as text source to represent the programming language corpus. Some characters are extensively used in programming languages, like “i”, “f”, and “:” particularly in python, because of the frequent occurrence of “if” condition, and “for” loop.

This corpus contains 64 characters that will be put onto different positions of the keyboard to form 64 keys, same as in the literature environment. Table 8.2 shows

characters included in this coding text environment. There are two major differences between this text environment, when compared with the literature one: first, characters “#” and “”” replaced characters “{” and “}” in the literature environment; second, the distribution of character frequencies is more uniform.

8.3 Simulations Results

I computed two different character frequency tables from the two corpus, with adaptive button switch as the input device. The results of using the literature corpus are the same as that in the Chapter "Different Cursor Paths". Here I focus on the results of using the Python codes as the corpus. I have the same simulation settings as in the Chapter "Different Cursor Paths", and the logistic regression model of button adaptive device is applied, except that the corpus is of programming language.

The results are shown in Figure 8.1(a), 8.1(b), and 8.1(c) respectively.

Table 8.2.: Character frequency of Python programing environment

Char	Freq
' '	1980
'a'	211
'b'	54
'c'	132
'd'	171
'e'	297
'f'	76
'g'	48
'h'	49
'i'	204
'j'	17
'k'	56
'l'	109
'm'	167
'n'	213
'o'	245

'p'	142
'q'	7
'r'	428
's'	198
't'	271
'u'	117
'v'	48
'w'	42
'x'	62
'y'	58
'z'	2
'!'	3
'@'	1
','	101
'.'	152
'%'	15

':'	57
'∧'	0
'&'	1
'*'	649
'('	113
')'	113
';'	0
'"'	70
'?'	0
'+'	22
'='	81
'<'	8
'>'	3
'/'	21
'\'	2
'#'	244

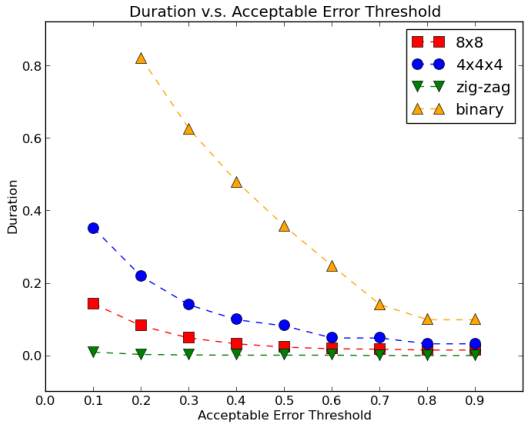
'"'	30
'['	65
']'	65
'-'	13
'_'	91
'∼'	0
'0'	100
'1'	80
'2'	29
'3'	7
'4'	13
'5'	24
'6'	19
'7'	12
'8'	4
'9'	6

Generally speaking, the pattern of results in this simulation is quite similar to that in the Chapter "Different Devices": there does not exist a feasible solution for the binary type keyboard with an acceptable error threshold $\epsilon = 0.1$. The "Zig-Zag" path always has the best performance when using the python codes as the corpus.

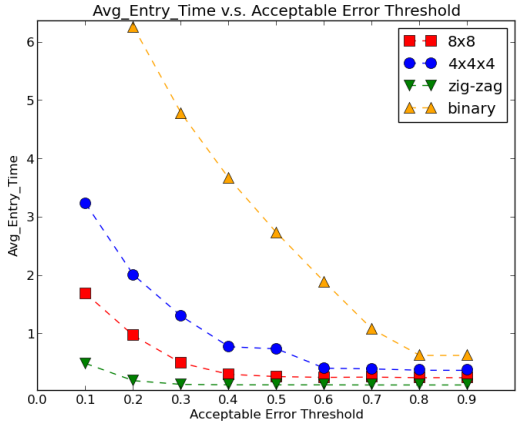
Since I concluded in previous sections that the "Zig-Zag" keyboard always has the best performance, in terms of average entry time, this thesis is using "Zig-Zag" keyboard to demonstrate the results of different text environments. Figure 8.2 shows two generated optimized keyboards to illustrate the characteristics of the results: Figure 8.2(a) shows the "Zig-Zag" keyboard with the smallest feasible duration generated when $\epsilon = 0.15$, in a literature text environment, of which the measures are same as that in 6.8(b). And Figure 8.2(b) illustrates the "Zig-Zag" keyboard also with the smallest feasible duration generated when $\epsilon = 0.15$, in a programing text environment, of which the average entry time is 0.378s, the average error rate is 0.150 and the smallest duration is 0.007s.

8.4 Conclusion

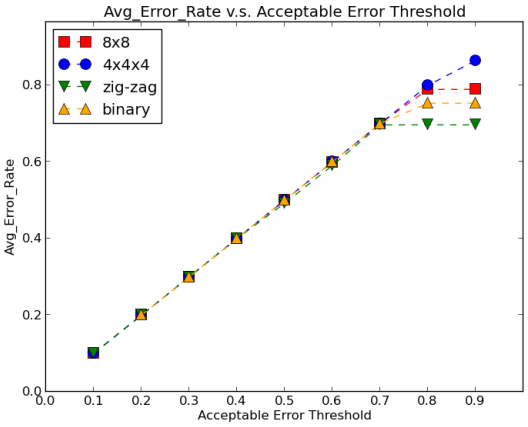
As is shown, the frequencies of characters, are playing a determining role in generating the final keyboard layout. The simulation successfully gave a quantitative analysis of all the four different path designs when using python codes as corpus. To summarize, the "Zig-Zag" keyboard is the best among all four path designs in this text environment.



(a) Smallest cursor duration

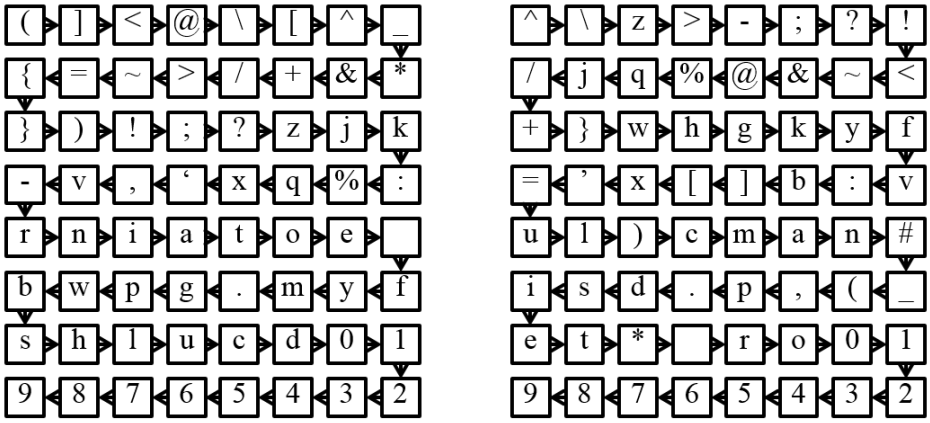


(b) Average entry time



(c) Average Error Rate

Fig. 8.1.: Simulation results of different paths for Python Codes



(a) Literature environment, $\epsilon = 0.15$ (b) Programing environment, $\epsilon = 0.15$

Fig. 8.2.: Examples of generated “Zig-Zag” keyboards for different text environments

Part III

Conclusions

9. CONCLUSION

In the era of big data, we also have numerous stored records of human behaviour data. This thesis is trying to bridge optimization and Data Mining and HCI by applying them towards HCI problems, by investigating a particular problem. We made this point by focusing on such a problem: the design of a switch virtual keyboard, for users with motor control difficulties.

This chapter relates the contributions, impact, and future directions of the study.

9.1 Contributions

This thesis answered some questions that are important in the HCI and Data Mining research communities, such as:

How to rephrase and abstract the design problem quantitatively

In Chapter 4, this thesis set up industrial standard measurements, the time cost and the error cost as the measurements of a design. This setting up naturally lead to a representation of the original problem into an optimization problem.

How to cast a industrial optimization problem into a model with global solution

In Chapter 4, this thesis re-framed the original problem into a Mixed Integer Programming (MIP) problem that can guarantee a global solution. And a well-known solver (Guroby) could provide solutions through very fast computation.

How to establish a model on Human Behaviour In Chapter 5, this thesis established a model for error rate prediction in users' behaviour, by using data mining and Bayesian statistical method. Also, the data were collected by HCI experiments by real users.

This thesis provided new insights to the interaction of data mining and HCI:

Methodologies:This thesis casts a light on the complicated system analysis: Globally, a problem of that kind can be treated with a determined approach that can provide a determined solution. Locally, when some sub-problems are too fuzzy to get a determined answer, we can try heuristic methods to find a solution to the sub-problem as close to the real one, and plug that solution back to the global problem, to yield a solution to the original global problem.

Algorithms: This thesis proposed the idea that some human-computer interactive patterns can be explained and modelled by using classical algorithms or their variations, like binary search tree.

Theories: This thesis established the theory that error rate of human behaviour can be modelled by using a variation of the traditional logistic regression. In traditional logistic regression, only the response class is used as an output of the algorithm, by comparing the probabilities of classes. Here I took one step back, and used the probability itself as the predicator. This transformed the logistic regression problem into a non-linear regression problem. But the model is entirely satisfactory and can be further improved.

Applications: The proposed methods were used to generate keyboards. This thesis compared the performances of optimized virtual keyboards, considering different design elements. Based on those research results in this thesis, real keyboards can be produced by using similar methods, for users who need this type of technology.

This thesis linked quantitative methods like Operational Research, Data Mining, Computational Modelling and so on, with HCI problems. I am looking forward towards a deeper interaction between above mentioned disciplines and HCI. Meanwhile this thesis advocates using the existing HCI data to advance research based on them.

9.2 Impact

This thesis gave a solution to the construction of more efficient switch based virtual keyboards for users with motor control difficulties, and was trying to extend the use

of this kind of keyboard to real scenarios for normal people in variety of situations. Also, it is of high probability that the research results can be commercialized.

9.3 Future Directions

So far This thesis had constructed quantitative models on the switch device related virtual keyboard, and evaluated the performance of those generated keyboards based on different design element. To look forward, this thesis also has some potential directions to expand the research.

9.3.1 Different Design Elements

This thesis has already considered a variety of different design elements. Nevertheless, they can still be regarded as prototypes, since the keyboard layouts in the study are generalized cases. It is possible to extend the approaches to different situations. For example, future researches can combine different cursor paths into one, in a hierarchical structure, and evaluate its performance. Also, future researches can consider other possible devices and texts. By looking into some other large corpus database, it is possible to establish more generalized or more specific text statistics tables as input.

9.3.2 Automation With Machine Learning

It is possible to extend the project with some application from machine learning, especially from natural language processing. Future researches can incorporate some classical models like collaborative filtering, hidden markov model, bayesian network and so on. These models can help predict the word that the user want to input, by using typed part of the word.

APPENDIX

REFERENCES

REFERENCES

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*.

Cao, L. (2008). Behavior Informatics and Analytics: Let Behavior Talk. In *2008 IEEE International Conference on Data Mining Workshops*, pages 87–96.

Czepiel, S. A. (2014). Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381–91.

Francis, G. and Johnson, E. (2011). Speed-accuracy tradeoffs in specialized keyboards. *International Journal of Human–Computer Studies*, 69(7-8):526–538.

Francis, G. and Rash, C. (2005). Analysis and Design of Keyboards for the AH-64D Helicopter. Technical report, US Army Aeromedical Research Laboratory, Fort Rucker, Alabama.

Fu, Y.-F. and Ho, C.-S. (2009). A Fast Text-Based Communication System for Handicapped Aphasiacs. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 583–594.

Gopher, D. and Raij, D. (1988). *Typing with a two-hand chord keyboard: will the QWERTY become obsolete?*, volume 18.

Hevizi, G., Biczo, M., Poczos, B., Szabo, Z., Takics, B., and Lorincz, A. (2004). Hidden Markov model finds behavioral patterns of users working with a headmouse driven writing tool. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, pages 669–674.

Hollis, G., Westbury, C. E., and Peterson, J. B. (2006). NUANCE 3.0: using genetic programming to model variable relationships. *Behavior research methods*, 38(2):218–28.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265. Citeseer, Morgan Kaufmann.

Hughes, D., Warren, J., and Buyukkokten, O. (2002). Empirical Bi-Action Tables: A Tool for the Evaluation and Optimization of Text-Input Systems. Application I: Stylus Keyboards. *HumanComputer Interaction*, 17(2-3):271–309.

Koene, A. (2014). Behavior Informatics project.

Laureys, S., Pellas, F., Van Eeckhout, P., Ghorbel, S., Schnakers, C., Perrin, F., Berré, J., Faymonville, M.-E., Pantke, K.-H., Damas, F., Lamy, M., Moonen, G., and Goldman, S. (2005). The locked-in syndrome: what is it like to be conscious but paralyzed and voiceless? *Progress in brain research*, 150:495–511.

Lipps, D. B., Galecki, A. T., and Ashton-Miller, J. A. (2011). On the Implications of a Sex Difference in the Reaction Times of Sprinters at the Beijing Olympics. *PLoS ONE*, 6(10):e26141.

Mackenzie, I. S., Zhang, S. X., and Soukoreff, R. W. (1999). Text entry using soft keyboards. *Behaviour & Information Technology*, 18(4):235–244.

Mayzner, M. S. and Tresselt, M. E. (1965). Tables of Single-Letter and Diagram Frequency Counts for Various Word-Length and Letter-Position Combinations. *Psychonomic Monograph Supplements*, 1(2):13–32.

Morland, D. V. (1983). Human factors guidelines for terminal interface design. *Communications of the ACM*, 26(7):484–494.

Norte, S. and Lobo, F. G. (2007). A virtual logo keyboard for people with motor disabilities. *ACM SIGCSE Bulletin*, 39(3):111.

Polson, N., Scott, J., and Windle, J. (2013). Bayesian Inference for Logistic Models Using PólyaGamma Latent Variables. *Journal of the American statistical Association*, 108:1339–1349.

Ward, D. J., Blackwell, A. F., and MacKay, D. J. C. (2000). Dasher—a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology - UIST '00*, pages 129–137, New York, New York, USA.

Zhai, S., Hunter, M., and Smith, B. A. (2002). Performance Optimization of Virtual Keyboards. *HumanComputer Interaction*, 17(2-3):229–269.