IJQwerty: What Difference Does One Key Change Make? Gesture Typing Keyboard Optimization Bounded by One Key Position Change from Qwerty

Xiaojun Bi, Shumin Zhai

Google Inc.

Mountain View, CA, USA

{xiaojunbi, zhai}@acm.org

Figure 1. Keyboard Layouts Optimized for Gesture Typing and Learnability. (a) IJQwerty: one key difference from Qwerty (*I* and *J* are swapped). (b) QwertyH1: a key is at most *horizontally* one key away from its Qwerty position. Compared to Qwerty, 8 pairs of keys get swapped: {E, R}, {Y, U}, {P, O}, {A, S}, {D, F}, {G, H}, {K, L}, and {B, N}. (c) Qwerty1: a key is at most one key away from its Qwerty position in both *horizontal* and/or *vertical* directions. Changed keys are classified into 6 sets: {Q, W}, {A, S, Z}, {R, F, E, V, C, D, G}, {N, H, B}, {I, O, K, U, J}, and {M, L}. Within each set, a key is moved one of other keys' Qwerty position. Note that colored rectangles are for illustration only. They do not appear on the actual keyboards.

ABSTRACT

Despite of a significant body of research in optimizing the virtual keyboard layout, none of them has gained large adoption, primarily due to the steep learning curve. To address this learning problem, we introduced three types of Qwerty constraints, *Qwerty1*, *QwertyH1*, and *One-Swap* bounds in layout optimization, and investigated their effects on layout learnability and performance. This bounded optimization process leads to IJQwerty, which has only one pair of keys different from Qwerty. Our theoretical analysis and user study show that IJQwerty improves the accuracy and input speed of gesture typing over Qwerty once a user reaches the expert mode. IJQwerty is also extremely easy to learn. The initial upon-use text entry speed is the same with Qwerty. Given the high performance and learnability, such a layout will more likely gain large adoption than any of previously obtained layouts. Our research also shows the

disparity from Qwerty substantially affects layout learning. To minimize the learning effort, a new layout needs to hold a strong resemblance to Qwerty.

Author Keywords

Text Entry; Gesture Typing; Layout Optimization; Virtual Keyboard.

ACM Classification Keywords

H.5.2. [Information interfaces and presentation]: User Interfaces-Input devices and strategies.

INTRODUCTION

Qwerty layout, which was originally invented in the 19th century for mechanical typewriters, has long been known as inferior for one finger or stylus touchscreen text entry. It unnecessarily elongates the travelling distance of the input finger/stylus, and introduces high ambiguity to the keyboard decoder [9, 21].

The problems are exacerbated in the modern mobile computing era where mobile text entry proliferates and many users enter text by gesture typing (e.g., Google Keyboard, ShapeWriter, Swype, SwiftKey, SlidIt, and TouchPal). The accuracy of gesture typing is severely affected by the ambiguity of Qwerty [21]. Because common vowels such as *u*, *i* and *o*, cluster together, many pairs of words have identical gestures (e.g., *tip* vs. *top*), and many others have very similar gestures (e.g., *bit* vs. *but*, *not* vs.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

Copyright is held by the owner/author(s).

CHI'16, May 07-12, 2016, San Jose, CA, USA

ACM 978-1-4503-3362-7/16/05.

<http://dx.doi.org/10.1145/2858036.2858421>

nit, *if* vs. *of*, *in* vs. *on*, and *fix* vs. *fox*). Smith, Bi and Zhai's [21] analysis of a 40000-word lexicon showed that 6.4% of words have another word with an identical gesture on Qwerty.

To improve text entry experience, a significant body of research has been conducted to search for a replacement of Qwerty. However, though a number of optimized layouts have been proposed, none of them has gained wide adoption.

The biggest challenge of newly optimized keyboard layouts is learning. To achieve superior performance over Qwerty, users have to spend a considerable amount of time practicing. Only a few users are willing to make such an effort. Additionally, users suffer from performance degradation when they initially switch to a new layout, because of the steep learning curve [9]

The observation that it is hard to move away from Qwerty against what people have already learned and memorized in users motor control skills has led to the nickname *Qwertynomics* for the more general theory of suboptimal path-dependent technosocial lock-in effects [8, 15]. In the HCI field previous research has already begun to take an *if you can't beat 'em, join 'em* approach by searching layouts using Qwerty similarity as one of the multiple optimization objectives. For example Smith, Bi and Zhai [21] used the average Manhattan distance of their new layout to Qwerty as one of their three objective functions. However, the learning advantage of these optimized layouts tended to be still too weak. Even though the average distance of the new key positions on an optimized layout (i.e., GK-T in [21]) to Qwerty might be small (on average 1.6 keys), each particular key to Qwerty could be either in the same location, or up to 4 keys away [21]. It means the Qwerty gaze pattern may not consistently guide the initial visual search behavior to the right location or even the right neighborhood. Bi et al [5] introduces the Quasi-Qwerty constraint in layout optimization, where each key is allowed to move at most one key away from its original Qwerty position. Such a method reduces the initial text entry time by 50% over a freely optimized layout. However, compared with Qwerty, its initial text entry time per word is still over 1000 ms longer. The initial performance degradation still prevents it from being widely adopted.

So the question facing the field is: is it possible to design a layout that has superior performance over Qwerty, and requires little learning effort? Does such a layout ever exist?

We explore this research question by taking a more radical approach. Instead of using the average distance to Qwerty as an objective function, we use a tight bound to Qwerty as a hard optimization constraint. The first type of bound is that no key position can be more than **one key position** away from Qwerty in any direction. It means a new key position is either in the same position as in Qwerty or in the

immediate neighborhood of its Qwerty position. We dub the result of such bounded optimization as *QwertyI* (Figure 1c). Note that this constraint is the same with the Quasi-Qwerty constraint introduced in Bi, Smith, and Zhai's work [5].

The second type of bound is to further limit the **one key position** change to the **horizontal** dimension only. This means the user would only have to search for the new location of a key between the left or right immediate neighbor if it were not in the same location as in Qwerty. We dub the optimized layout with the second type of bound as *QwertyH1* (Figure 1b).

The third type of one key position change is bounded by swapping only **one pair of keys**, and no key position can be more than **one key position** away from Qwerty in any direction. This is to minimize the memory load of the user when typing (initially) on the new layout. As we will see, optimization search result within this third type of bound happens to be *I* and *J*. We dub the optimized layout with the third type of bound as *IJQwerty* (Figure 1a). Though *IJQwerty* was discovered through rigorous optimization processes, it happens to possess three potential and specific advantages: 1) *I* and *J* are next to each other diagonally on Qwerty. 2) *I* and *J* are consecutive letters in the Roman alphabet. 3) *I* and *J* have similar visual appearance. All of these three factors contribute to the strong pairing effect to *I* and *J*, making swapping *I* and *J* easy to remember and easy to adapt to. Of course, one can also argue a subtle difference may more easily cause confusion than large changes and only empirical investigation would tell.

IJQwerty clearly stands out in our investigation. Our user study shows it reduces word error rate and improves input speed over Qwerty once a user reaches the expert mode. Thanks to its extreme similarity with Qwerty, *IJQwerty* is also very easy to learn. The initial text entry speed on *IJQwerty* is similar to that on Qwerty. Surprisingly, *IJQwerty* has only minor performance degradation compared to a layout optimized with far more freedom, such as the *QwertyH1* constraint.

Our research also reveals that the disparity from Qwerty strongly influences the learnability of a new keyboard layout. A layout optimized with the *QwertyH1* constraint still introduces 500ms longer initial visual search time per word than Qwerty. Our investigation indicates that to achieve high learnability, the new layout needs to hold a very strong similarity to Qwerty.

RELATED WORK

QWERTY, designed by Christopher L. Sholes and colleagues in 1867 [22], is the de facto standard of keyboard layout for both physical and virtual keyboards. As has been widely published [6, 9, 14, 16, 18, 21, 23], Qwerty was designed to minimize jamming in mechanical typewriters by placing common digraphs (consecutive letter pairs) on opposite sides of the keyboard. Though such a

design works well for two-hand typing on a physical keyboard, it is suboptimal as a virtual keyboard layout for one finger or stylus input. Starting from at least as early as Getschow and colleagues' optimization work for increasing efficiency for the motor impaired [11, 14], researchers have tried to find various ways to design more efficient alternatives, first using simple algorithms [11, 14] or heuristics [16], eventually to more rigorous mathematical optimization [5, 6, 23].

Optimizing Layouts for Smart Keyboard

As mobile text entry proliferates, a sizeable amount of research has been conducted to optimize the keyboard layout for gesture typing and touch typing (i.e., tapping).

Regarding gesture typing, Rick [20] proposed the OSK layout which had faster input speed over Qwerty, according to the model derived from Fitts' law. Smith, Bi and Zhai [21] optimized keyboard layout for gesture clarity, gesture speed, and similarity to Qwerty. Zhai et al. [24] designed the ATOMIK layout, which was available as ShapeWriter gesture keyboard in iPhone App Store in 2007. It shortened the average travel distance per character from 3.31 to 1.94 key width.

Regarding touch typing, Dunlop and Levine [9] introduced the Pareto front optimization in the layout optimization process. Their work led to layouts optimized for three metrics: finger travel distance, tapping ambiguity, and familiarity to Qwerty. Oulasvirta et al. [18] proposed the KALQ layout, which was designed to improve two thumb typing on a split keyboard. Bi, Smith, and Zhai [6] explored the optimality issues pertaining to multilingual touchscreen keyboard design, discovering that including multiple languages in the optimization introduces only minor performance degradation to each language.

Improving Layout Learning

Though previous research has proposed a number of optimized layouts, few of them have gained wide spread adoption, primarily due to the learnability issue. Researchers have explored different approaches to reduce the learning burden, with limited success.

On ATOMIK [24], the alphabetical ordering of letters were considered. It reduces the visual search time for letters like "a" and "z", the first or last letters alphabetically, but offers limited help to other letters.

Both Smith et al.'s [21] and Bi et al.'s [5] work consider introducing Qwerty similarity to improve learning. As aforementioned, although Bi et al.'s [5] work shows that introducing the Quasi-Qwerty constraint leads to promising results, it achieves only limited success.

OPTIMIZING KEYBOARD LAYOUT WITH QWERTY CONSTRAINTS

Our ultimate goal is to find a layout that reduces error rate, improves speed over the status-quo, and is also easy to learn. We designed three types of Qwerty constraints for the layout optimization procedure, and evaluated their effects

on keyboard learnability, input accuracy and speed. Note that the current research focuses on gesture typing.

Three Qwerty Constraints

We define a key's position on Qwerty as its *Qwerty position*, and the positions at most one key away from its Qwerty position as its *Qwerty neighbors*. E.g., key A has four Qwerty neighbors: Q, W, S, and Z, and key H has 7 Qwerty neighbors: T, Y, U, G, J, B, and N. Based on these concepts, we designed the following 3 types of Qwerty constraints:

1. Qwerty1 Constraint.

A key is only allowed to move to its Qwerty neighbors, which is at most one key away from its Qwerty position. Such a constraint is the same with the Quasi-Qwerty constraint introduced by Bi, Smith and Zhai [5].

2. QwertyH1 Constraint.

A key is only allowed to move to its horizontal Qwerty neighbor. Such a constraint is stronger than the Qwerty1 constraint, and is expected to increase learnability. Note also that in the smartphone portrait mode form factor keys are packed more tightly in the horizontal dimension than in the vertical dimension, which could further reduce the learning burden of QwertyH1.

3. One-Swap Constraint.

Only one swap is allowed, and the swap must happen between a key and one of its Qwerty neighbors.

These three constraints specify different levels of similarity to Qwerty. One-Swap constraint is the strongest. The strictness of constraints decreases from One-Swap, to QwertyH1, and to Quasi-Qwerty. As the constraint becomes less strict, the optimized layout shares less similarity with Qwerty. On the other hand, the optimization space becomes larger, which might lead to layouts with better performance. Varying the level of Qwerty constraint essentially trades the size of the optimization space with the resemblance to Qwerty.

Optimization Metrics

The two metrics in the optimization procedure are *Gesture Clarity* and *Gesture Speed*. Gesture clarity measures how unique the word gestures on a keyboard layout are. It quantifies the effect of a layout on gesture typing accuracy. Complementary to gesture clarity, gesture speed estimates how quickly users can gesture type on a keyboard layout. The definitions of these two metrics are identical to those in Smith, Bi and Zhai's work [21]. These two metrics are calculated based on a 20,000-word lexicon in this work.

Optimization Procedure

Designing a layout optimized for gesture typing is a multi-objective optimization problem, where the two objectives are (1) improving gesture clarity and (2) improving gesture speed. More specifically, the objective function in the optimization is:

$$O(l) = aC(l) + (1 - a)S(l) \quad (0 \leq a \leq 1) \quad (1)$$

where $C(l)$ is the gesture clarity for a given layout l , and $S(l)$ is the estimated gesture typing speed. $C(l)$ is calculated from Equation (1) and $S(l)$ is from Equation (7) in [21]. α is a weighting factor.

To solve this multi-objective optimization problem, we adopted the commonly used Pareto optimization method, [4, 9, 21]. Since the optimization procedure in this work is mostly identical to that in Smith et al.'s work [21], we focus on explaining the procedures unique to the current problem.

1. *Metric Normalization*. Similar to that in [21], the purpose of this step is to estimate the minimum and maximum possible values for each metric. We then linearly normalize each of the metric's scores so that the minimum and maximum values are mapped to 0.0 and 1.0 respectively. The metric score before the normalization is called *raw value*, while it is referred as *normalized value* after the normalization.

None of the Qwerty constraints was introduced in this step: a key could be swapped with any other key on the keyboard. The normalization step showed that the range of gesture clarity metric was [0.35, 0.74] key width, and the range for gesture typing speed was [51.6, 78.6] WPM. Figure 2 shows layouts with maximum gesture clarity and speed.

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

(a) Layout optimized for gesture clarity without any Qwerty constraint.

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

(b) Layout optimized for gesture speed without any Qwerty constraint.

Figure 2. Layouts optimized for gesture clarity only (a) and speed only (b).

2. *Pareto Front Generation*. We generated Pareto fronts with Qwerty1, QwertyH1, and one-pair-swapping constraints separately, following the same procedure in [21]. Note that the optimization space has only 62 layouts when one-swap constraint is implemented. We simply conducted an exhaustive search in this condition. This step generates three Pareto fronts (Figure 3). Figure 4 shows some layouts on Pareto Fronts.

3. *Select the optimized layout*. We select the layout closest to the 45° line on each Pareto front as the most balanced layout under a specific Qwerty constraint. Such a layout has the a balanced gesture clarity and speed (Figure 1 and Figure 4).

Optimization Results

Figure 4 and Table 1 show the performance of generated optimized layouts. As expected, the performance of optimized layouts decreases as the Qwerty constrain is becoming stronger. The highest normalized gesture clarity under Qwerty1 constraint is 0.751, while it drops to 0.497 under QwertyH1, and 0.471 under One-Swap (Figure 4, left column). The same trend holds for the speed metric (Figure 4, middle column). This finding is understandable. The more strict a Qwerty constraint is, the smaller the search space becomes. With a smaller search space, the optimized layout is less likely to be superior.

Figure 3 Pareto Fronts for Qwerty1 (left), QwertyH1 (middle) and One-Swap (right)

Layout	Gesture Clarity Normalized	Raw (Key Width)	Gesture Speed Normalized	Raw (WPM)	# of Keys Different From Qwerty
Qwerty	0.345	0.48	0.454	63.8	0
Qwerty1	0.585	0.58	0.557	66.6	22
QwertyH1	0.470	0.53	0.501	65.1	16
IJQwerty	0.469	0.53	0.495	64.9	2
Clarity Only	1.0	0.74	0.37	61.6	25
Speed Only	0.28	0.46	1.0	78.6	26

Table 1. Performance of optimized keyboard layouts. Qwerty1, QwertyH1, and IJQwerty are shown in Figure 1, while layouts optimized for Clarity or Speed without any constraints are shown in Figure 2.

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

Qwerty1 with Highest Clarity
Clarity: 0.705, Speed: 0.473

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

Qwerty1 with Fastest Speed
Clarity: 0.415, Speed: 0.677

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

Most Balanced Qwerty1
Clarity: 0.585, Speed: 0.557

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

QwertyH1 with Highest Clarity
Clarity: 0.497, Speed: 0.456

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

QwertyH1 with Fastest Speed
Clarity: 0.408, Speed: 0.561

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

Most Balanced QwertyH1
Clarity: 0.470, Speed: 0.501

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

One-Swap with Highest Clarity
Clarity: 0.471, Speed: 0.426

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

One-Swap with Fastest Speed
Clarity: 0.328, Speed: 0.499

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

Most Balanced One-Swap: IJQwerty
Clarity: 0.469, Speed: 0.495

Figure 4. Layouts on the Pareto Fronts. Both Clarity and Speed are normalized values.

As we are particularly interested in the layouts with the most balanced gesture clarity and speed, we closely examined the layouts closest to the 45° line on Pareto Fronts (Figure 4.)

Under the One-Swap constraint, the layout closest to the 45° line is IJQwerty (Figure 1b), on which I and J keys are swapped (Figure 1a).

QwertyH1, the most balanced layout under the QwertyH1 constraint, is the layout on which 16 keys are moved to one

of its horizontal Qwerty neighbors. These 16 keys can be grouped into 8 pairs: (E, R), (Y, U), (O, P), (A, S), (D, F), (G, H), (K, L), and (B, N). Keys in each of the pairs get swapped. The remaining 10 keys remain unchanged.

On the most balanced layout generated under the Qwerty1 constraint (Qwerty1, Figure 1c), 22 keys are moved to one of its Qwerty neighbors. These 22 keys can be grouped into 6 sets: {Q, W}, {A, S, Z}, {E, R, F, V, G, D, C}, {B, N, H}, {I, O, K, U, J}, {M, L}. In each of sets, keys are moved to one of the other keys' Qwerty position..

One surprising result is that IJQwerty, the layout obtained with the one-swap constraint, shows superior performance over Qwerty, especially for gesture clarity. The gesture clarity IJQwerty is 0.53 key width, which is 10.4% higher than that of the Qwerty layout (0.48 key width). This result concurs with the finding from previous research that Qwerty layout is notoriously error-prone for gesture typing. This result suggests that the gesture typing accuracy could be improved simply by making a swap. Our daily experience of gesture typing shows that Qwerty keyboard often has problems recognizing *bit* vs. *but*, *not* vs. *nit*, *if* vs. *of*, *in* vs. *on*, and *fix* vs. *fox*, because the common vowel letters *u*, *i*, *o* are clustered together. With a simple swap, IJQwerty could alleviate these problems. In addition to IJQwerty, we found that IKQwerty (I and K are swapped), and OKQwerty (O and K are swapped) also improve the gesture clarity over Qwerty. The gesture clarity scores are 0.526 key width for IKQwerty and 0.522 key width for OKQwerty. Both of them are higher than that of Qwerty (0.48 key width), which shows that separating I, O, and K keys apart will boost the accuracy.

Another interesting finding is that IJQwerty has only slight performance drops compared to the layout optimized with the QwertyH1 constraint. The gesture clarity and speed of IJQwerty are only 1.2% and 0.3% lower than those of QwertyH1.

USER STUDY

The optimization processes led to Qwerty1, QwertyH1 and IJQwerty which improve the gesture clarity and speed over Qwerty, and possess different levels of similarity to Qwerty. To evaluate whether these layouts lead to superior performance in real usage, and how the gains in gesture clarity and speed metrics translate to the actual accuracy and speed improvements, we conducted a formal user study. In particular, we focus on investigating how the Qwerty constraints affect layout learnability.

Experimental Setup

Tasks

The study consisted of two sessions: a phrase input session, and a word repetition session.

The purpose of the phrase input session is to evaluate users' initial upon use text entry speed of a particular layout. It reflects the learnability of the keyboard. In this session, each participant gesture typed the same set of 20 phrases randomly chosen from MacKenzie and Soukoreff's test set [17]. Participants were instructed to gesture typed words as naturally as possible, and they were allowed to freely correct the words, and use the suggestions provided by the keyboard.

Complementary to phrase input, the 2nd session of the study includes a word repetition task in which each participant gesture typed a word seven times consecutively for 20 words. Participants were not allowed to use the backspace

or suggestion bar. The purpose of this task was to evaluate the expert performance on a given layout. As a user repeatedly enters the same word multiple times, she will reach the expert performance in the last few repetitions. Examining the performance over the last few repetitions reveals the potential expert performance users can reach. The 20 words in the study are “*the and you that is in of know not they get have were are bit quick fox jumps lazy on*” The first 19 words are from the list proposed by Zhai and Kristensson [25], which cover all letters of the English alphabet and approximate both letter frequencies and digraph frequencies in English. The word “on”, which also has a high frequency and easily gets confused in gesture typing was added to the 19-word list, in order ensure that both phrase input and word repetition tasks include 2 blocks and each block has 10 stimuli.

Design

The study was a within-subject design. The independent variable was the keyboard layout with 4 levels: Qwerty, Qwerty1, QwertyH1, and IJQwerty. Each participant firstly performed 20 phrase input trials, followed by 20 word repetition trials, on all the four layouts. Since every participant was familiar with Qwerty and no learning effort was required for this layout, each participant firstly entered text on Qwerty, and then on the other three optimized layouts. The orders of the three optimized layouts were counterbalanced across participants by a Latin square. Prior to the formal study, each participant performed a warm-up session of 5 phrases. The dependent variables were word error rate, and input speed, which were the commonly used metrics measuring text entry performance [2, 21]. shows the application used in the experiment.

Figure 5. A screenshot of the study application.

15 subjects participated in the study. 9 of them were female, and all of them were right-handed. Their ages ranged from 20 to 45. All the participants were familiar with the Qwerty layout, and entered text with gesture typing on a daily basis. 12 participants entered text with the index finger, while other used the thumb. The study was conducted on Nexus 5 devices running Android 5.0. The keyboards in test were Google keyboards with different layouts. As default, the top suggestion of the gesture typing was shown directly in the text view, while the 2nd and the 4th suggestions were displayed on the suggestion bar. Participants could choose the suggestion and freely correct errors in the phrase input tasks, but not in the word repetition tasks. In summary, the study included:

15 subjects $\times$ 4 layouts $\times$ 20 phrases = 1200 phrases

15 subjects $\times$ 4 layouts $\times$ 20 words = 1200 words

Metrics

The metrics are input speed and word error rate. The input speed was calculated as:

$$WPM = \frac{|S|}{T} \times 60 \times \frac{1}{5} \quad (2)$$

where S was the final transcribed string and T as the elapsed time in seconds from the moment the phrase appeared on the screen to the finish of the last word in the phrase. Note that the numerator was $|S|$, instead of $|S - 1|$ because the time to enter the first character was also included.

Since most of the current smart keyboards perform correction at word level, we adopted the word error rate to measure accuracy. The word error rate is based on Minimum Word Distance (MWD) [2], which is the smallest number of word deletions, insertions, or replacements needed to transform the transcribed string into the expected string. The word error rate is defined as:

$$\text{Word Error Rate} = \frac{MWD(S,P)}{\# \text{ of words in } P} \times 100\% \quad (3)$$

Results

Speed and WER for Phrase Input

Figure 6 shows the means (Std. Error) of input speed across all the participants. The means of input speed in WPM were 42.8 for Qwerty, 17.52 for Qwerty1, 28.02 for QwertyH1, and 40.39 for IJQwerty. ANOVA showed the layout had a significant main effect on the input speed ($F_{3,42} = 3.412$, $p < 0.05$). The pairwise comparisons showed the differences between every two layouts ($p < 0.05$) were significant except for Qwerty vs. IJQwerty.

Figure 7 shows the means (SEM) of word error rate across all the participants. Since participants were allowed to freely use backspaces and correct errors, this metric measures the final error rate after corrections have been performed. The means are 5.0% for Qwerty, 5.8% for Qwerty1, 5.2% for QwertyH1, and 5.9% for IJQwerty. ANOVA did not show a significant main effect of layout on Word Error Rate ($F_{3,42} = 2.01$, $p = 0.127$).

Figure 6. Mean (SEM) of Input Speed of Phrase Input.

Figure 7. Mean (SEM) of Word Error Rate of Phrase Input.

In summary, input speed differed across layouts, except between Qwerty and IJQwerty, while word error rates remained similar for all the four layouts.

Speed and WER of Word Repetition

Figure 8 shows the input speed per the repetition position. We firstly examined the input speed at the first repetition, which includes the initial visual search time when users use an unfamiliar new layout. The mean input times (ms) of the word at the first repetition position are 2554.98 for Qwerty, 4011.23 for Qwerty1, and 3041.02 for QwertyH1, and 2645.91 for IJQwerty. ANOVA showed a main effect of layout ($F_{3,42} = 4.02$, $p < 0.05$) and pairwise comparisons showed the differences were significant between every two layouts ($p < 0.05$) except for Qwerty vs. IJQwerty.

Figure 8. Mean Input Time per Word by Repetition Position

As shown in Figure 8, the input time remains stable since the third repetition, indicating that participants reached the expert level from #3 to #7 (Figure 9). Their input speeds were largely governed by their motor limitation in these repetitions. Considering the input speed from #3 to #7 repetitions only, the average input times (ms) per word are 1530.3 for Qwerty, 1276 for Qwerty1, 1299.27 for QwertyH1 and 1331.23 for IJQwerty. ANOVA showed a main effect of keyboard ($F_{3, 42} = 3.432$, $p < 0.05$), and pairwise mean comparisons showed the differences are significant between Qwerty and any of other three layouts ($p < 0.05$). However, no significant difference was found between other three layouts.

Figure 9. Mean (Std. Error) Input Time per Word from Repetitions #3 to #7.

Figure 10. shows the word error rate for word repetition tasks. The mean error rates were 13.54% for Qwerty, 9.6% for Qwerty1, 10.4% for QwertyH1, and 10.3% for IJQwerty. ANOVA showed a significant main effect of layout ($F_{3, 42} = 2.98$, $p < 0.05$) and pairwise comparisons showed the differences were significant between every two layouts except for QwertyH1 vs. IJQwerty.

Figure 10. Mean (Std. Error) of Word Error Rate of Word Repetition Input.

Discussion

IJQwerty has the highest learnability.

As expected, IJQwerty, the closest layout to Qwerty, is the easiest to learn. In the phrase input session, which is the first 10 minutes of using a given layout, the input speed on IJQwerty is similar to that on Qwerty and is significantly faster than those of Qwerty1 and QwertyH1. The same result is observed when examining the first word input speed in word repetition session. These results indicate that IJQwerty introduces little learning burden to novice, and has higher learnability than Qwerty1 and QwertyH1.

In addition to the minimum difference to Qwerty, IJQwerty happens to possess other advantages, which might contribute to the high learnability: 1) *I* and *J* are next to each other diagonally on Qwerty. 2) *I* and *J* are consecutive letters in the Roman alphabet. 3) *I* and *J* have similar visual appearance. All of these three factors make strong pairing effect to *I* and *J*, contributing to the ease of learning. In contrast, though Qwerty1 and QwertyH1 are also close to Qwerty, they still introduce non-negligible learning burden to users. It indicates that to eliminate performance degradation when users initially use a new layout, the layout has to hold a very strong resemblance to Qwerty.

Optimized Layouts show superior expert performance over Qwerty.

Users were more accurate and faster in the last 5 repetitions in word repetition tasks on all the three optimized layouts than on Qwerty. This finding concurs with the predictions from gesture clarity and speed metrics.

As predicted by the metrics, study results show Qwerty1 has the lowest error rate and fastest speed for expert users, followed by QwertyH1 and IJQwerty. However, the differences between QwertyH1 and IJQwerty are minor. No significant differences were observed between these two layouts. Both QwertyH1 and IJQwerty were 11.7% relatively faster than Qwerty, and the WER were 24% lower than Qwerty.

IJQwerty shows performance improvements over Qwerty. It is probably because swapping *I* and *J* pushes away the clustered vowels *u*, *i*, and *o*, reducing the ambiguity of many common words such as *tip* vs. *top*, *bit* vs. *but*, *not* vs. *nit*, *if* vs. *of*, *in* vs. *on*, and *fix* vs. *fox*.

In summary, the study results concur with the findings in the optimization process. IJQwerty, stands out as an easy-to-learn, and superior layout over Qwerty. When users initially use the IJQwerty, the performance is similar to Qwerty, indicating that little learning cost is imposed. When users reach expert level, their WER are reduced and input speed are improved over Qwerty.

LIMITATIONS AND FUTURE WORK

This work investigated the layout learnability in laboratory experiments, where participants were exposed to the new layouts for a short period of time, and only a limited

number of phrases were covered. Though the empirical results concur with the theoretical predictions, future work should include a longitudinal study, such as Reyat et al.'s experiment [19], to verify the findings. In the longitudinal study, layouts will be deployed to users for daily usage and the metrics will be logged. Such a study will be a great complement to the current laboratory experiment.

As our primary goal is to keep the change as minimum as possible, we did not evaluate the effects of two, three or more key swaps on learnability. It is worth investigating how the number of swapped keys will affect the layout learnability.

Though the focus of this research is gesture typing, it will be interesting to see how a layout optimized for gesture typing affects touch typing (i.e., tapping keys). Intuitively, the performances of gesture typing and touch typing are positively correlated. Optimizing a layout for gesture typing speed means shortening the average length of word gesture traces, which also shortens the average finger travel distance of entering a word in touch typing, especially for one-finger input. Optimizing a layout for gesture accuracy tends to push apart keys that would otherwise easily get confused with neighbor keys, in order to provide distinct shapes for common words. This process also benefits touch typing. For example, on IJQwerty, swapping *i* and *j* pushes apart key *i* from key *u*, which provides distinct shapes for words like *bit* and *but*. It will also likely increase the touch typing accuracy for *bit* and *but*, because users are less likely to land touch point on the key *u* when they intend to type *i*.

CONCLUSIONS

Though previous research has proposed a number of optimized keyboard layouts, none of them has gained large adoption, due to their steep learning curve. Learnability clearly is key for successful layout adoption. In this research, we addressed this problem by introducing three types of Qwerty constraints, namely Qwerty1, QwertyH1, and One-Swap constraints to the layout optimization process. We performed layout optimization for gesture typing with these constraints separately, and investigated their effects on layout learnability and performance. Our investigation leads to an optimized layout (i.e., IJQwerty) with high learnability and superior performance over Qwerty.

Our work, for the first time, proves that making one pair of keys change on Qwerty: swapping *I* and *J* (i.e., IJQwerty), improves input accuracy and speed of gesture typing over Qwerty, once a user reaches the expert mode. Our theoretic analysis shows that the IJQwerty increases *gesture clarity*, a metric that describes the extent to which word gestures are distinct from each other, by 10.4% over Qwerty, and also improves the predicted gesture typing speed. User studies in which participants repeatedly entered words showed that the IJQwerty reduced the error rate by 24%, and improved input speed by 11.7% over Qwerty, once users reached the expert mode.

A feature that distinguishes the IJQwerty from previously optimized layouts is that it is very easy to learn. The IJQwerty imposes little learning burden to users. User studies did not show any degradation in either speed or accuracy over Qwerty when users initially used IJQwerty. Although IJQwerty is generated through a rigorous optimization procedure, it happens to possess strong paring effects as previously outlined, which contribute to the ease of learning too.

Our research also reveals that to eliminate performance degradation when users initially use a new layout, the layout has to hold a very strong resemblance to Qwerty.

ACKNOWLEDGEMENTS

We would like to thank Elnaz Sarbar and Nilgoon Askari for running user studies, Kurt Partridge and CHI reviewers for their valuable insights and comments to this work.

REFERENCES

1. Shiri Azenkot and Shumin Zhai. 2012. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services* (MobileHCI '12), 251-260.
2. Xiaojun Bi, Shiri Azenkot, Kurt Partridge, and Shumin Zhai. 2013. Octopus: evaluating touchscreen keyboard correction and recognition algorithms via. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 543-552.
3. Xiaojun Bi, Ciprian Chelba, Tom Ouyang, Kurt Partridge, and Shumin Zhai. 2012. Bimanual gesture keyboard. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12), 137-146.
4. Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2014. Both complete and correct?: multi-objective optimization of touchscreen keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14), 2297-2306.
5. Xiaojun Bi, Barton A. Smith, and Shumin Zhai. 2010. Quasi-qwerty soft keyboard optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), 283-286.
6. Xiaojun Bi, Barton A. Smith, Shumin Zhai. (2012) "Multilingual Touchscreen Keyboard Design and Optimization", *Human-Computer Interaction*, Volume 27, Issue 4, 352-382.
7. Xiang Cao and Shumin Zhai. 2007. Modeling human performance of pen stroke gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07), 1495-1504.
8. Paul A. David. 1995. Clio and the Economics of QWERTY. *The American Economic Review*. Vol. 75, No.2 332 - 337

9. Mark Dunlop and John Levine. 2012. Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12), 2669-2678.
10. Paul M. Fitts. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 381-391.
11. Getschow, C.O., Rosen, M.J., and Goodenough-Trepagnier, C. A systematic approach to design a minimum distance alphabetical keyboard. In *Proc. RESNA 1986*, RESNA (1986), 396-398.
12. Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. In *Biometrika* 57, 1 (1970), 97-109.
13. Per-Ola Kristensson and Shumin Zhai. 2004. SHARK²: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST '04), 43-52.
14. James R. Lewis, Peter J. Kennedy, Mary J. LaLomia (1999). Development of a Digram-Based Typing Key Layout for Single-Finger/Stylus Input. *Proceedings of The Human Factors and Ergonomics Society 43rd Annual Meeting*.
15. Liebowitz, Stan J., and Stephen E. Margolis. "The fable of the keys." *Journal of Law and Economics* (1990): 1-25
16. I. Scott MacKenzie and Shawn X. Zhang. 1999. The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*(CHI '99), 25-31.
17. I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '03). ACM, New York, NY, USA, 754-755.
18. Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 2765-2774.
19. Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. Performance and User Experience of Touchscreen and Gesture Keyboards in a Lab Setting and in the Wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 679-688.
20. Jochen Rick. 2010. Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology* (UIST '10), 77-86.
21. Brian A. Smith, Xiaojun Bi, and Shumin Zhai. 2015. Optimizing Touchscreen Keyboards for Gesture Typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3365-3374.
22. Hisao Yamada. A historical study of typewriters and typing methods: from the position of planning Japanese parallels. *Information Processing* 2, 4 (1980), 175--202.
23. Shumin Zhai, Michael Hunter, and Barton A. Smith. 2000. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (UIST '00), 119-128.
24. Shumin Zhai, Michael Hunter, and Barton A. Smith (2002). Performance optimization of virtual keyboards. *Human-Computer Interaction*, 17(2,3), 89-129.
25. Shumin Zhai and Per Ola Kristensson. 2008. Interlaced QWERTY: accommodating ease of visual search and input flexibility in shape writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 593-596.
26. Shumin Zhai and Per Ola Kristensson. 2012. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM* 55, 9 (September 2012), 91-101.