A Rapid Screening and Testing Protocol for Keyboard Layout Speed Comparison

Joonseok Lee, Hanggjun Cho, *Student Member, IEEE*, and R. I. (Bob) McKay, *Senior Member, IEEE*

***Abstract*—We propose an efficient low-overhead methodology for screening key layouts for ultimate typing speed. It is fast and complementary to existing protocols in other ways. For equal overhead, it allows testing over a wider range of users. It is subject to potential biases, but they can be quantified and adjusted. It assumes an existing standard layout which is used as a comparator. It eliminates bias from differing familiarity and finger memory by appropriately mapping both layouts. We illustrate it with two mobile phone keypad layouts: Samsung's variant of the common ABC layout, and a new user-specific layout (PM). We repeat a comparison previously undertaken by a training-based method, which used ten participants, and estimated a 54% speedup for PM (SD $= 35\%$). The new method used 116 participants and estimated a 19.5% speedup (SD $= 7.5\%$). Differences in speedup estimates can be explained by the wide confidence interval of the training-based methods, differences in test corpuses, and the inherent conservatism of the new method. Effects of user characteristics could be meaningfully tested due to the larger test group. Gender had no detectable effect on any of the measures, but age and keypad experience were significantly related to ABC and PM performance. However, the relative speedup was unaffected by age or keypad experience: the method can remove comparison biases arising from differing experience levels.**

***Index Terms*—Comparison, keypad, keyboard, layout.**

I. Introduction

MANY devices use keyed input for character data. The mapping of the input space (keys) to the output space (text)—the layout—heavily affects efficiency. Often there are a number of available layouts: the "de facto standard" [1] QWERTY and the Dvorak [2] layouts for English; the North and South Korean layouts for Korean (Fig. 1 [3]).[1] [2] Thus, we often need to compare two layouts—one well-known to users, the other unfamiliar. Existing methods for estimating ultimate speed, using extended training, take substantial effort and considerable elapsed time; with today's short product life cycles, faster methods for preliminary evaluation are needed.

One challenge to testing new layouts is the time it can take for participants to reach expert performance; but people bring experience with prior layouts to such experiments. We suggest a protocol using this experience to support rapid, direct comparison, by eliminating the effects resulting from differences in familiarity. We first describe the general idea, then show an analysis based on this protocol, comparing a personalised mobile phone keypad layout with a well-known "ABC" layout. We compare the theoretical improvement in efficiency, based on a model of typing speed, with the experimental results obtained from our protocol. We conclude with a discussion of the assumptions and limitations of the approach, of the ways it may be effectively combined with other techniques, and some possible future extensions.

II. Background

A. Earlier Methodologies

There are obstacles to fair rapid layout comparison. Familiarity with the incumbent biases direct comparisons. There are two obvious solutions: using complete novices, or providing sufficient time for complete familiarization with the new layout. Both have serious limitations. It may be difficult to find complete novices, where they can be found, they are atypical—children or new language users. Conversely, no rapid-testing environment can approach the speed of long-term users. Yet the typical one week's training is too time intensive for early prototyping, and severely limits the range of test subjects.

In order to evaluate the rapid testing protocols, we set several conditions their comparisons should satisfy:

1) Familiarity difference effects should be minimized.
2) Comparisons should be at a level that is as close to expert as possible.
3) The elapsed time should be as short as possible.
4) The methodology should support recruiting a large number of participants from a variety of backgrounds.
5) In particular, the testing regime should not require a large investment of time on the part of participants.

Previous strategies for dealing with the difference in familiarity can be roughly categorized into four groups:

1) Ignoring the familiarity issue.
2) Providing short familiarization sessions.
3) Restricting testing to novices.
4) Observing learning progress.

1) Ignoring the Familiarity Issue: the simplest approach, creating participant groups and analyzing the results independent of familiarity. It has been used by a number of researchers whose main concern was initial acceptance of a new design.

Manuscript received August 26, 2013; revised May 27, 2014, October 15, 2014, and December 4, 2014; accepted December 5, 2014. Date of publication January 14, 2015; date of current version May 13, 2015. This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2012-013-NRF-2012S1A2A1A01031076). This paper was recommended by Associate Editor Prof. Jenna Marquard. (*Corresponding author: R. I. (Bob) Mckay*, e-mail: urilabob@gmail.com).

J. Lee is with the Statistical Machine Learning and Visualization Lab, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: joonseok2010@gmail.com).

H. Cho and R. I. (Bob) McKay are with the Structural Complexity Lab, School of Computer Science and Engineering, Seoul National University, Seoul 151744, Korea (e-mail: hanggjun@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2014.2380641

[1]Both put consonants on the left and vowels on the right, but details differ.

[2]This figure is adapted from wikipedia/wikimedia, under a Creative Commons Attribution—Share Alike 3.0 Unported license.

2168-2291 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

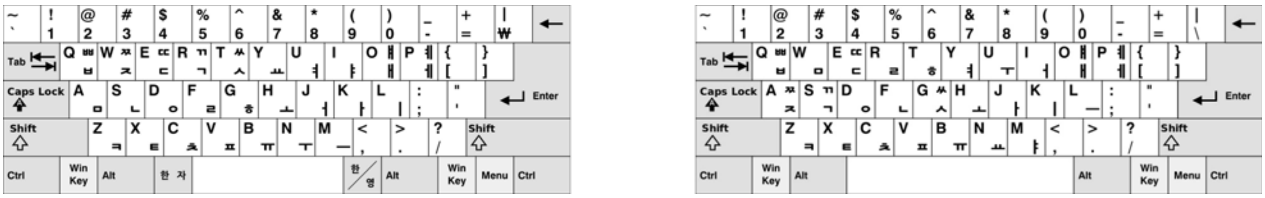


Fig. 1. Korean keyboard layout (Left: South Korea Dubeolsik Keyboard, Right: North Korea Dubeolsik Keyboard).

MacKenzie *et al.* [4] compared walk-up acceptance of three text input methods on a pen-based computer, testing 15 participants with no training. Lewis *et al.* [5] used a similar strategy for stylus input, with 12 QWERTY-familiar testers, noting the unfairness of the comparison, while Green *et al.* [6] used ten participants to evaluate stick keyboards against a QWERTY keyboard, and Mittal and Sengupta [7] tested their improvised mobile phone keypad layout with six participants.

This strategy satisfies criteria 3 to 5, at the cost of 1 and 2. It can only evaluate design acceptability, not ultimate speed.

2) Providing Short Familiarisation Sessions: to mitigate the worst effects of layout unfamiliarity and reduced bias. Because sessions are short, it partly satisfies criteria 3 to 5.

Butts and Cockburn [8] evaluated multitap mobile phone keypads against the two-key method. They allowed enough practice time for participants to feel comfortable—usually less than 1 min. Gong and Tarasewich [9] employed a very short training time in evaluating their alphabetically constrained mobile phone layout against an unconstrained optimized one and ABC-layout. It consisted of two sample sentences, which may have been enough to reduce the gross errors expected from the first use of a layout.

The familiarization approach trades off improved performance on criterion 1 against some loss on 3 to 5, but does not address criterion 2.

3) Restricting Testing to Novices: has an apparent simplicity and fairness that has appealed to some researchers.

Hirsch [10] used 55 nontypists to compare QWERTY with Griffith's Minimotion [11] layout. After an initial trial, 15 were eliminated (their QWERTY speed was too high to have come from novices). Hirsch anticipated that novices would be faster with Minimotion's recognizable alphabetical order. In fact, they were good at QWERTY. He concluded the participants may have had QWERTY experience, acknowledging failure to find true QWERTY novices. Harnett [12] used a similar evaluation method, finding that novice users can reach 50 WPM (words per minute) after 5 h, while a 50 WPM QWERTY typist reached 35-40 WPM with Dvorak after 4 h.

Restricting testing to novices clearly satisfies criteria 1, 3, and 5. Criterion 2 is ignored, while criterion 4 is problematic because novices are rare and atypical.

4) Detailed Observation of the Learning Progress: trains participants until the learning rate tails off, and then estimates the asymptotic limit. Learning progress can be compared even when participants have differing experience. It has the important advantage of generating learnability data in addition to speed data, but it imposes substantial overheads.

Michaels compared an alphabetic layout with QWERTY [13], using 30 participants over 25 sessions, and finding no advantage for the alphabetic layout. Thomas *et al.* [14] compared three kinds of wearable computer, using an hour's training spread over six sessions over three weeks. MacKenzie and Zhang [15] compared a new mobile keypad against QWERTY, with five participants undertaking 20 experimental sessions of 45 min, concluding that the new design was easier to learn, and would be faster than the old after sufficient practice. Ingmarsson *et al.* [16] applied detailed observation of interface learning to television appliances, with five participants over ten sessions. Hwang and Lee compared a QWERTY-like phone keypad layout [17] with a common ABC layout. They concluded from five sessions over five days that it was both easy to learn, and ultimately faster. We used detailed observation of learning [18] to evaluate a personalised multigram (PM) layout against an ABC 12-key layout. Ten participants trained with PM until they reached their initial ABC speed, extrapolation of the progress leading us to conclude that PM is more efficient.

Although these examples appear successful, there are important limitations. The experimental commitment—around a week of participants' time—means criteria 3 and 5 are not satisfied. For these reasons, the preceding experiments used only very narrow pools of participants – often laboratory members. This can cause serious problems in statistical reliability. In [15], the number of participants was limited to 5. All were CS (Computer Science) majors, and only one was female: criterion 4 was not satisfied. On the other hand, this method offers the highest likelihood of satisfying 1 and 2. Thus, it is a kind of "gold standard," providing relatively high-reliability statistics. Our method is complementary to detailed observation of learning, providing rapid prescreening: the overhead of the training approach need only be incurred on layouts already tested as promising.

These conclusions are summarized in Table I. No strategy satisfies all criteria. For effective and reliable screening, we need the new protocol proposed in Section III.[3]

B. Theoretical Models

Our experimental validation compares two layouts for two-thumb use of a multistroke keypad. There have been a number of previous attempts to model the factors determining keying speed on different keyboards. We mention some of the most relevant below.

1) Two-Thumb Keyboards: MacKenzie and Soukoreff [20] studied two-thumb text entry on miniature QWERTY

[3]A brief introduction to the method was previously presented in [19].

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

TABLE I
EVALUATION OF TRADITIONAL KEYBOARD COMPARISON METHODOLOGIES

Method	Fairness	Expert-level	Scalability	Breadth	Low Participant Commitment
Ignoring the familiarity issue	×	×	✓	✓	✓
Restricting testing to novices	✓	×	×	✓	✓
Providing training session	×	×	✓	○	○
Observing learning progress	✓	✓	×	×	×

×: fails; ✓: satisfies; ○: partially satisfies

keyboards, developing a theoretical model based on two fundamental components, the time to move a thumb between keys (using Fitts' law [21]), and a minimum time between keystrokes on opposite thumbs (set at 88 ms). It relied on a specific assignment of keys to each thumb. It was subsequently amended and validated by Clarkson *et al.* [22], who noted that practiced users will often use the opposite thumb to this model, where it would result in faster typing—depending on context, "g"' might be typed with the right hand or "h" with the left. While the QWERTY keyboards physically differ from mobile phone keypads, their model is readily adapted to such keypads.

2) One-Thumb Keypads: Moradi and Nickabadi [23] analyzed one-thumb keypads as part of an exercise to optimize the layout using a genetic algorithm. Their model, while not specifically mentioning Fitts' Law, is compatible with it, and in effect can be seen as a one-thumb restriction of the MacKenzie and Soukoreff model to keypads.

3) Two-Thumb Keypads: Combining the ideas from the above, we previously [24] generated a theoretical model of two-thumb keypad entry, validating it against training experiments.

III. METHODOLOGY

A. Familiarity Issues

There are two main familiarity issues in comparing layouts. Participants differ in their recollection of key locations. Experts immediately locate each key, while less experienced users need to recollect key locations, which may require around a second, and complete novices waste time looking for keys. This variation is typical for familiar layouts like QWERTY; for a new layout, on the other hand, all users are novices. For fair comparisons, we need to close the novice-expert speed gap. People who care about keyboard efficiency presumably aspire to high keyboard speed. Therefore, we need to standardize to that level.

The second category of familiarity is "finger memory." Frequently used phrases eventually lead to unconscious knowledge of relative movements on the keyboard. For example, experienced users do not separately locate the "t," "h," and "e" keys to type "the" in English. The more experienced the user, the more the words are remembered through finger memory of the sequence. We need to eliminate this source of bias.

To remove such biases, there are two alternatives: standardization to the level of an experienced user, or to the level of a novice. For location memory, we propose a compensation approach, mapping characters to transfer the knowledge of the familiar layout to the new one. For finger memory, we use a penalty strategy, eliminating any expertise benefit.

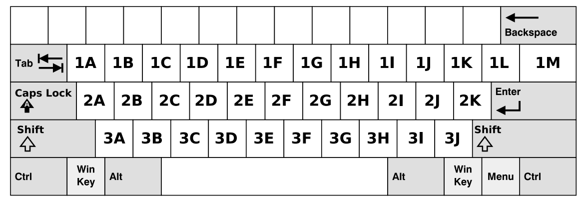


Fig. 2. Example of key coding.

B. Handling Location Memory Through Character Mapping

We need a way to transfer users' proficiency to the new layout. We outline a general description here, noting that it needs adaptation to specific cases. We assign a value to each key, the keystroke code. How it is coded depends on the device. For a computer keyboard, an integer may be assigned to each key. For a multitap mobile phone keypad, we code both the key and stroke count to fully specify each character assignment.

We formalize the function f_{layout} through

$$f_{\mathrm{layout}}(l) : L \to C. \tag{1}$$

Here, L denotes the target letters and C denotes keystroke codes. We require f_{layout} to be bijective: each symbol in the target language L corresponds to exactly one keystroke in C so that there is an inverse described in

$$f^{-1}_{\mathrm{layout}}(c) : C \to L. \tag{2}$$

Iterated application of f_{layout} to a string generates a sequence of keystrokes, and f^{-1}_{layout} inverts this.

Suppose we wish to compare an "old" layout and a "new" one—say QWERTY and Dvorak—with "computer" as test word. Users can type "computer" in QWERTY, but they cannot immediately type it in Dvorak. So instead, we ask them to type $f^{-1}_{\mathrm{qwerty}}(f_{\mathrm{dvorak}}$("computer")) in QWERTY. Using the coding in Fig. 2, and the layouts in Fig. 3[2] f_{dvorak}("computer") is the sequence ⟨1H, 2B, 3G, 1D, 2D, 2H, 2C, 1I⟩. Applying f^{-1}_{qwerty} gives "ismrfkdo"; measuring its typing time in QWERTY tells us the time to type "computer" in Dvorak.

More abstractly, users know f_{old}, but not $f_{\mathrm{new}}(s)$. So we ask them to type $f^{-1}_{\mathrm{old}}(f_{\mathrm{new}}(s))$ using the old keyboard. They apply f_{old} (which they know) to $f^{-1}_{\mathrm{old}}(f_{\mathrm{new}}(s))$. What they produce is the string $f_{\mathrm{old}}(f^{-1}_{\mathrm{old}}(f_{\mathrm{new}}(s)) = f_{\mathrm{new}}(s)$—what we desired—using their knowledge of the "old" layout.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

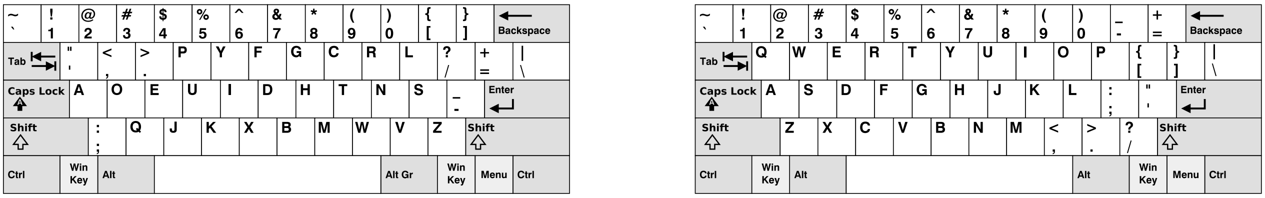


Fig. 3. Keyboard layouts for character mapping example (Left: Dvorak Keyboard, Right: QWERTY Keyboard).

C. Removing Finger Memory Bias Using Layout Transformation

However, this removes only the first source of bias—"computer" and "ismrfkdo" differ in relative familiarity, and we have finger memory of "put" but not "rfk." To remove the familiarity and finger memory biases, we need to define a layout transformation

$$t : C \rightarrow C \tag{3}$$

that remaps the keyboard. The transformation aims to remove word familiarity and finger memory from the current layout, while preserving physical speed. We can often base our choice on symmetries on Fitts' Law.

Continuing our QWERTY-Dvorak example, the alphanumeric keyboard is (approximately) physically symmetric in both vertical and horizontal axes, and touch typists use it in a symmetric way. Fitts' Law implies that we could reflect the keyboard about either the horizontal or vertical centerline with little effect on physical speed (for a two-hand touch typist) because movement distance changes are small (they were near-zero in classic typewriter layouts, but modern keyboard convention has seen an increased left-to-right keyboard slant that introduces small differences in distances). However, vertical reflection of the whole keyboard exchanges rarely used numerals—which may not be fully remembered—with letters; therefore, we will not use it. We could also use vertical reflection about the "asdf" line, but this leaves many letters, and even some words ("glad") completely unchanged.

Instead we use both vertical reflection about the "asdf" line and horizontal reflection to define t. So 2A ("a") maps to 2J (":") and 3A ("z") to 1J ("p"). This leaves some low-frequency right side characters unchanged; if our tests do not use them, this will not matter. This symmetry preserves invariants (specific finger, movement distance) that, according to Fitts' Law, determine typing speed. It varies other invariants (hand, horizontal direction) that have little effect for most people. However, it does exchange some frequent character positions from the top row to the more cramped bottom row; according to Fitts' Law (and our experiments) this has little effect on speed. It is guaranteed to remove finger memory, since every character is typed with a different finger from the original.

To compare the layouts using "computer," we apply t to the keycodes. For testing QWERTY, we ask the user to type

$$f_{\text{qwerty}}^{-1}(t(f_{\text{qwerty}}(\text{“computer”}))) \tag{4}$$

i.e., "ixrzvn,m." For Dvorak, we request

$$f_{\text{qwerty}}^{-1}(t(f_{\text{dvorak}}(\text{“computer”}))) \tag{5}$$

i.e., "clrmjdkx." This is illustrated in Fig. 4[2] (the "original" order is shown by light gray shading with the numerical order at the top of the key; the "mapped" order by checkers and a numeral at the bottom).

More abstractly, we need to define a set of invariants that should be preserved, and then define a transformation—generally, using symmetries—that preserves these invariants [25]. The specific symmetries and invariants depend on the physical keyboard and layout. Having defined the transformation t, for the "old" layout, we use $f_{\text{old}}^{-1}(t(f_{\text{old}}(s)))$, and for the "new" $f_{\text{old}}^{-1}(t(f_{\text{new}}(s)))$.

Before starting the comparison, it is important to confirm any theoretical argument that t does not introduce biases. It should be checked, as far as possible, at two levels:

1) Are the intended invariants (finger frequencies, row frequencies, etc.) preserved in the target language or corpus? (This can be checked computationally.)
2) Does the transformation really not affect typing speed? (This requires experimental verification.)

For the former, we sample the test corpus in both original and transformed forms, collecting statistics of the invariants, and testing whether they change substantially, but what if our invariance assumptions are wrong? For example, handedness or cramping on the bottom row may have more effect than expected. We need to measure the extent to which t distorts typing speed. So we ask our participants to type the same phrases before and after transformation. When we find that typing speed is not perfectly preserved, we have three options:

1) Where the bias is large relative to performance differences between layouts, we need to discard the invariant (and the symmetries and transformations derived from it) and find new better invariants.
2) Where the bias is small relative to the performance differences, we may safely ignore it.
3) Where it is comparable in size, another option is to correct it mathematically.

D. Test Set Generation

As with all comparison methodologies, we should generate our test set from a corpus reflecting the intended use—representative text from a language for computer keyboards, or an SMS archive for a mobile phone layout. Our methodology introduces some further requirements. Since its emphasis

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

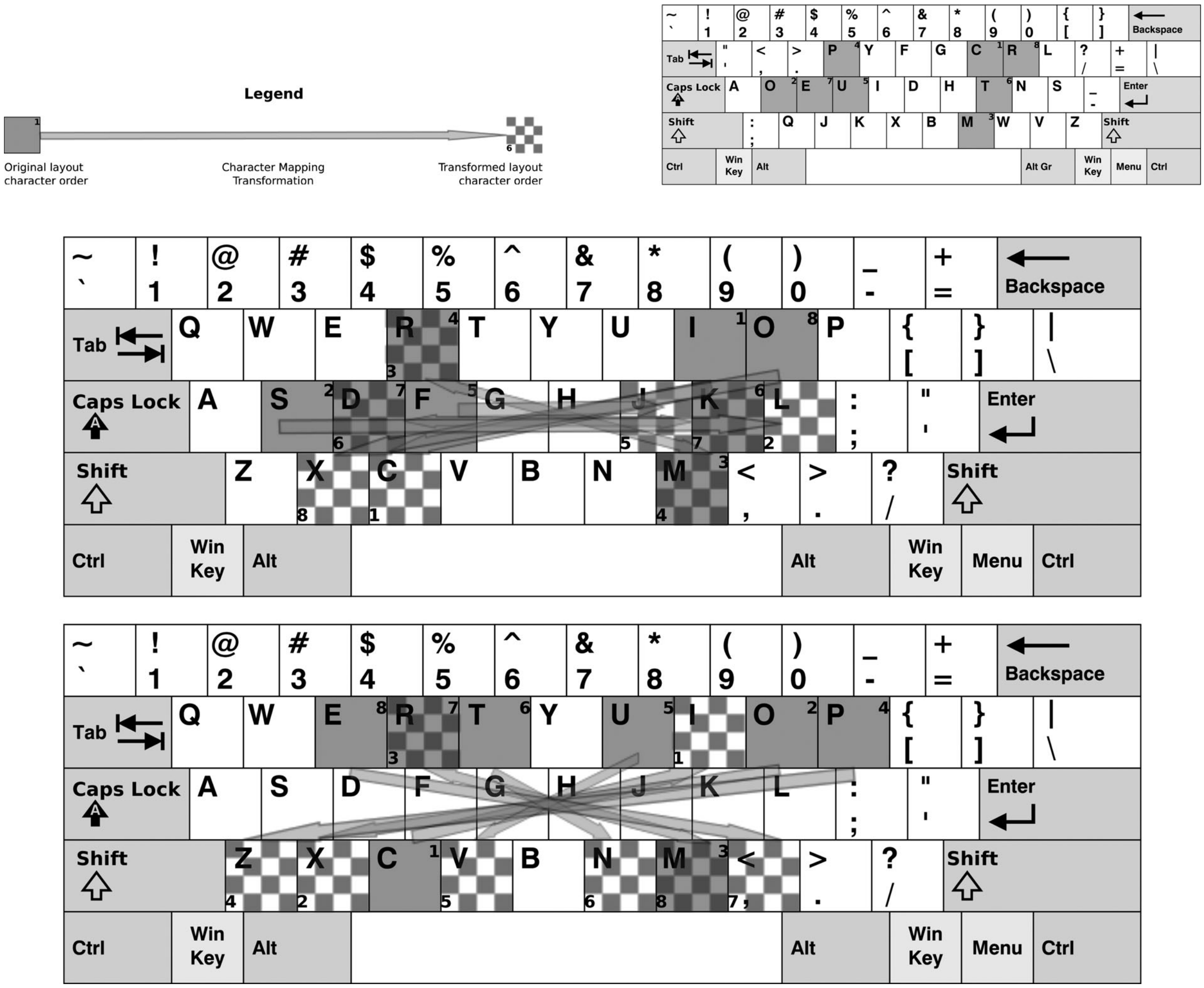


Fig. 4. Character mapping example using transformation "t." *Top*: Original Dvorak Key Strokes; letter "o" is in position 2B. *Middle*: Original and Mapped Dvorak Key Strokes on QWERTY Keyboard; Dvorak letter "o," at position 2B (QWERTY letter "s"), is mapped to position 2I, i.e., QWERTY letter "l". *Bottom*: Original and Mapped QWERTY Key Strokes; The second character of "computer," letter "o," initially in position 1I, is mapped to position 3B, i.e., the letter "x."

is speed of testing, each participant will only see a small sample of sentences, but small samples can still generate biases. If the transformation for one layout generates "uatspn," the other "wxkjlq," experienced English typists are likely to type the first faster than the second because the letters are more frequently used. Since such pairs would increase experimental noise, we prune them from the corpus—please see Appendix A for the mathematical definition of familiarity $\mathfrak{F}$ and bias $\mathfrak{B}$.

E. Overall Experimental Protocol

Fig. 5 illustrates the overall protocol. We first prepare a set of words from an appropriate corpus, filtered by relevant criteria, such as word length, the occurrence of special characters, etc. We also need to define the letter stroke coding, and the two functions $f_{\rm old}$ and $f_{\rm new}$. For QWERTY and Dvorak, we use the mapping in Fig. 2. We also define the transformation t as above. We then convert each word into two test strings by applying functions (6) and (7) to each word

$$f^{-1}_{\rm qwerty}(t(f_{\rm qwerty}(s))) \tag{6}$$

$$f^{-1}_{\rm qwerty}(t(f_{\rm dvorak}(s))). \tag{7}$$

For each pair, we check the bias $\mathfrak{B}$, eliminating those above our threshold. The remaining pairs make up our test set.

IV. Test Example: Mobile Phone Keypad Layout

We illustrate the methodology with an application to a personalised multigram-based (PM) mobile phone keypad layout we originally proposed in [24] and evaluated with a learning-based methodology. The layout uses up to four strokes on 10 out of the 12 keys of the traditional 12-key numeric keypad, permitting it to code the 26 alphabetic characters plus up to 14 frequently used multigrams, as shown in Fig. 6(b). We compare it with an "ABC" 1- to 3-stroke layout, widely used on Samsung phones and illustrated in Fig. 6(a).

We focus on two-handed use: the left thumb covering the left column, the right thumb the right, and either covering the center. It is used by those concerned with speed, who might switch to a faster layout if available. Two-handed use is faster

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

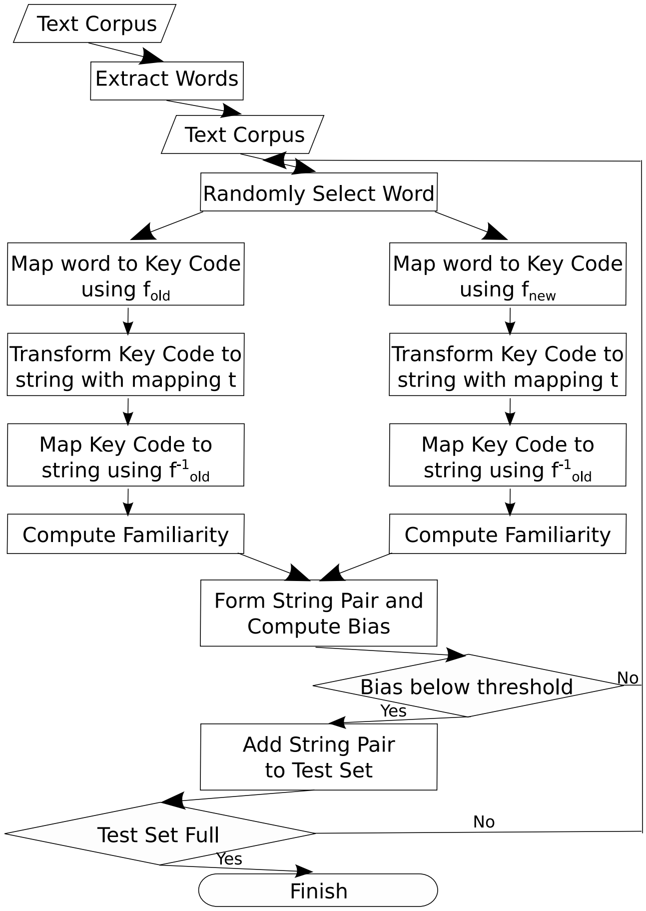
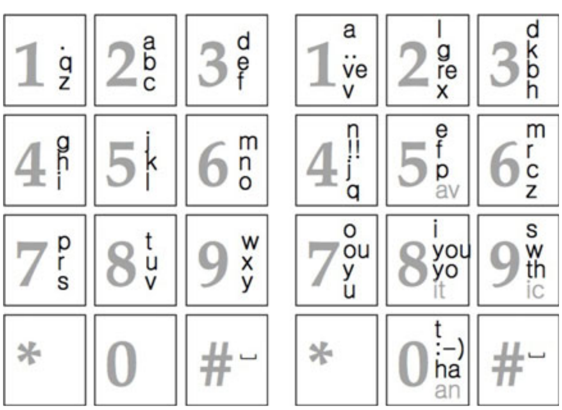


Fig. 5. Overall flow of the experimental protocol.



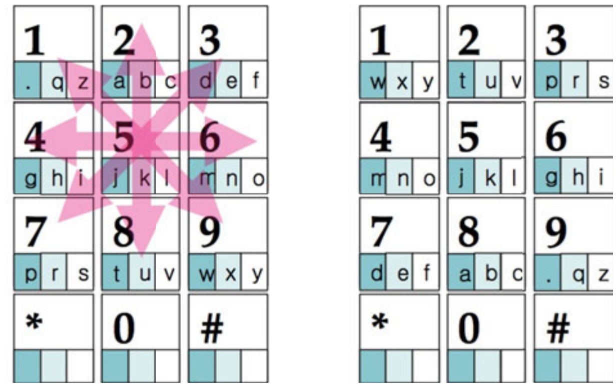
(a) Samsung ABC (b) Personalised Multigram

Fig. 6. Layouts for test keypads.

than one-handed for computer keyboards [26], [27], which we can reasonably generalize to mobile phone keypads. The PM layout reflects a user's personal use; for this evaluation, we use one particular archive (derived from the first author's SMS history), and evaluate both layouts against this archive.

A. Applying the Test Protocol

As illustrated in Fig. 5, we need to assign keystroke codes, and define the mappings f_{ABC} and f_{PM} assigning characters to



(a) Reflection on the Center Key (b) Transformed Layout

Fig. 7. Keypad layout transformation.

the code. We then generate the string pairs, which we filter for bias before generating the ultimate set of test string pairs. For the experiments, we used a bias threshold of 0.85 (see Appendix A for details).

1) Assigning Keystroke Codes: In both PM and ABC, we used 10 out of the 12 keys $(0, \ldots, 9)$ to represent letters and multigrams. It is an ambiguous keypad, in which more than one letter must be assigned to each key, using the number of strokes to disambiguate. Thus, the keystroke codes designate both the key used, and the number of strokes (in this case, $1, \ldots, 4$, although the ABC layout does not use 4).[4] To simplify explanation, we represent the key through its row and column position. Thus, each code consists of a three tuple (row, column, stroke), with $1 \leq row \leq 4, 1 \leq column \leq 3$, and $1 \leq stroke \leq 4$. The coding defines the positions of the "#" and "$*$" keys, but these are excluded from both character codings.

Following Fig. 5, we need to convert each word to a sequence of key codes. Because our coding covers every possible position for both layouts, this mapping is straightforward. For example, "hello" will be converted to $<$(2, 1, 2), (1, 3, 2), (2, 2, 3), (2, 2, 3), (2, 3, 3)$>$ under f_{ABC}.

2) Layout Transformation: To define the layout transformation t, we first define the invariants, for which we chose

1) The number of strokes for a letter.
2) The frequency of consecutive use of the same key.
3) The frequency of consecutive use of the same hand.
4) The average distance of movement of the thumbs.
5) The balance between the hands.
6) The distance from each key to the center.

Neglecting the key "0," the nine keys fall into a 3×3 matrix. We reflect each key about the center ("5") key. Letters on key "5" remain the same. This is depicted in Fig. 7. We preserve the number of strokes. The transformation conserves all the listed invariants. This analysis is consistent with those of MacKenzie and Soukoreff [20] and Clarkson *et al.* [22]. We confirmed

[4]To be precise, the ABC layout cycles again through the keys after reaching three strokes; this permits rapid correction in the case, where the user inadvertently makes an extra stroke. We do not examine this issue here, beyond noting that the same correction mechanism is available for both layouts, albeit with a slightly longer cycle for PM.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

experimentally that any effects on typing speed were small (see Appendix B). Since most characters are typed with different thumbs under the transformation, and even those that use the same thumb have reversed movement directions, we believe that this transformation eliminates the effects of finger memory.

B. Adapting the Methodology

However, we have glossed over an important issue. Our protocol assumes that the key spaces are identical—that the mappings f are bijective, but the PM layout uses the key "0," and up to four strokes. Neither is used by ABC. We have to adapt the protocol.

1) Additional Keypad Mapping: Because the "0" key is not used in ABC, transformation t must map it to one of the keys that are used (so that f_{ABC}^{-1} will be well defined). In a multitap two-thumb environment, the distance of movement of the thumbs is of minor significance. More important issues are whether movement is required at all, or if so, whether it can be overlapped with tapping by the other thumb, and most important of all, the number of strokes [24]. That paper used an evaluation function based on the number of keystrokes and consecutive uses of the same key and/or thumb. The simplest choice is thus to map key "0" to key "8," the nearest location. This will not affect whether we need to use the same thumb for consecutive strokes (since both are on the middle row, which can use either thumb), but it can change whether consecutive characters will be on the same key. So we adapt further; in such cases, key "0" is mapped to key "5" instead. We denote this additional keypad mapping as $m : C \to C$. To use this, we need to apply $f_{ABC}^{-1} \circ t \circ m$ instead of $f_{ABC}^{-1} \circ t$, in order to make it a well-defined mapping. This does not induce noticeable biases (see Appendix C).

2) Stroke Mapping: What should we do about letters where PM uses four strokes so that f_{ABC}^{-1} is undefined? Any alternative using four strokes would require two separate ABC letters, on either the same key or two different keys. In either case, it adds extra delay—either moving to a different key, or flagging the end of a letter—substantially biasing toward the ABC layout. However, the PM keypad is optimized to use these four-stroke positions for the lowest frequency letters and multigrams. Hence, we chose instead to map these cases to the corresponding three-stroke letters on the ABC keypad. The differences in typing speed were small, and because they are rare, the effect on overall typing speed even smaller. We introduced a small penalty of 1400 ms per person for the measured PM typing speeds to compensate (see Appendix C for the details of our estimation of this compensation).

C. Experimental Comparison: Settings

1) Text Corpus and Derived Word Pairs: We used an archive of mobile phone messages from the first author.[5] We dropped special characters, which would require complex lookup in the ABC layout (and for most, in PM as well). Since PM does include some special characters as multigrams, this results in some bias against PM. We also excluded words of fewer than three or more than ten letters. Of the 554 words originally extracted from the archive, 178 remained after filtering. We randomly selected a test set of 100 (discarding and resampling any that exceeded our threshold of 0.85 for $\mathfrak{B}$).

Ideally, we would test each participant on all 100 pairs—typing 200 random-appearing sequences in total. In preliminary tests, participants found 200 random strings in a session unduly tiring (more so than 200 recognizable words). They reported that 100 were acceptable, but we decided to be cautious and limit sessions to 50 words. We could have subjected each participant to four sessions of 50 words each, but this would increase the load on participants, conflicting with one of our primary criteria. Instead, we formed our participants into groups of four. For each group, the 100 pairs were randomly sorted, then split into samples of 25 (i.e., 50 words), each participant being allocated one of the samples. We had 116 participants so that each pair was tested on 29 participants. The randomization into samples was repeated for each group so that no two participants saw the same set of 25 words, or the same word order. For each pair and participant, we also randomized whether the PM or ABC string was presented first.

2) Experiment Participants: There were 60 male and 56 female participants, aged from 18 to 60 (mean 26.1, SD 6.0). Their occupations ranged over students, academic researchers, software engineers business and management, and games and entertainment; their areas of study ranged over science and medicine, engineering, social science, business, humanities, education and art and design.

We did not control for previous ABC experience, which would have required a further round of detailed measurement. Subjectively, all had some experience, ranging from highly skilled to limited. We used the direct measurement of their typing speed (since all strings were typed using the ABC layout) as a proxy for their ABC layout experience.

3) Procedure: We did not need a training session, because the methodology directly uses current facility in the ABC layout. Each participant undertook only one test. We instructed them to type as fast and accurately as possible, and not to pause while typing a word. We gave no information about how the test strings were constructed, describing them as "50 random strings."

The user starts by typing their name, then chooses a question bank (as instructed by the experimenter) in the range A–D[6] that determines the 25 word pairs. Clicking "START" commences the experiment. The software presents a string to type; pressing "OK" terminates input. The software displays current progress (out of 50 strings), and time for the previous string—from the first stroke to pressing the "OK" button—therefore, the user may rest between strings. After the 50 strings, the system displays the mean times for the ABC and PM layouts. The physical environment is detailed in Appendix D.

D. Results

1) Overall Results: We averaged the elapsed time for strings generated by ABC and PM layouts (henceforth, ABC and PM strings), applying the correction for 4-stroke to 3-stroke

[5] This archive was actually used to define the PM layout [19].

[6] The "M" button denotes "Manual," and is used only for debugging.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

TABLE II
MEAN $\pm$ SD OF ELAPSED TIME FOR ABC AND PM STRINGS (SECONDS)

	Elapsed Time
ABC Strings	8.16 ± 2.58
PM Strings	6.53 ± 2.02
PM Strings (Corrected)	6.58 ± 2.02

TABLE III
IMPROVED, SIMILAR, AND WORSENED WORD PAIRS WITH PM LAYOUT, BY THE MEAN $\pm$ SD ELAPSED TIME OF EACH PARTICIPANT

	Improved	Similar	Worse
Count	16.34 ± 2.45	2.36 ± 1.56	6.30 ± 2.18
Percentage	65.34%	9.45%	25.21%

conversion described earlier. Participants took 19.32% less corrected time for PM strings than ABC (see Table II). The biggest improvement was 34.86%; only 2 out of 116 participants had lower performance with PM (-1.43% and -10.52%). The Shapiro–Wilks normality test gave no indication that the data are nonnormal; for the Anderson–Darling test, normality for the ABC and PM data can be rejected at the 0.1 level, but not at 0.05. Since normality was uncertain, we undertook both parametric and nonparametric tests. A one-sided t-test indicated that the elapsed time for PM was significantly less than for ABC ($t = -5.3684$, $p < 10^{-7}$), as did a one-sided Wilcoxon rank-sum test ($z = -5.0240$, $p < 10^{-6}$).

We also counted the strings that had substantially improved performance (PM faster than ABC by over 10%), similar performance, or worse performance (PM slower by over 10%) (see Table III). About $^2/_3$ show better performance with PM, but $^1/_4$ are worse. This is important, as we will see later.

2) Detailed Analysis by Groups: Because the protocol takes much less user effort (5 to 10 min) than traditional protocols, we could conduct experiments with many more participants, from much more diverse backgrounds. We could thus evaluate different population characteristics, as we illustrate with age, gender, and previous ABC keypad experience in Table IV. We used the Kruskal–Wallace test (under the Holm–Bonferroni correction for multiple comparisons, although in this case the correction did not affect results) to determine whether each characteristic was linked to ABC performance, corrected PM performance or $\%$ improvement. Significant effects (at the 1% level) are bolded.

For age groups, there are two important observations:

1) Overall speed decreases substantially with age.
2) Nevertheless, the speed improvement between the two layouts remains almost constant, at around 20%.

Gender has no significant effect on any of the characteristics considered.

3) Effect of Familiarity: We designed our protocol to discount the effects of familiarity. How well did this work? We ranked participants by overall average time (ABC plus PM), and divided into ranges of ten people. The ABC and corrected PM performance were (inevitably, given the definition) closely related to the rank. If familiarity affected speedup, we would expect to see differences in speedup between the groups. Despite a very large range of raw speeds (a factor of over 3), the improvement from ABC to PM is very stable, improving by around 20%, with the minima in the middle of the speed range. For novices, the traditional analysis in [24] implies that we should see better performance with PM: there is no familiarity difference to overcome; therefore, any reasonable analysis should confirm it. Skilled users will show reduced differences under most protocols, because of the familiarity bias. We see little or no difference; therefore, our protocol has successfully eliminated this bias.

4) Comparison With Other Methodologies: We previously used two traditional methods [18], [24] to compare PM and ABC layouts (Methods 1 and 2 in Table V). One was a typical learning-based method. The other, briefly outlined in Section II, used a theoretical model of factors influencing human performance adapted from those of MacKenzie and Soukoreff [20], Clarkson *et al.* [22] and of Moradi and Nickabadi [23], and ultimately derived from Fitts' Law. Those experiments used the same corpus as described above (SMS messages), but without filtering. We compared this with our methodology (4).

All three agree that PM outperforms ABC, but disagree on the scale of performance difference. Filtering drops some of the multigrams that contribute to the PM layout's speed; therefore, it may reduce PM's advantage. To understand this, we repeated the theoretical analysis of (1) with the filtered corpus (3). Text filtering is the only difference between methods 1 and 3; therefore, this gave us a direct measure of the effect of filtering. Thus, filtering should reduce PM's performance advantage by around 17% (the difference between 1 and 3). This estimated difference should be roughly correct even if the theoretical model is inaccurate.

The new method has a relatively high cognitive load: it uses random-seeming strings, harder to remember than the English words of the training approach. During the "character recognition" phase [28], the eye reads slower than its maximum rate—just fast enough to feed copy to the hand as needed [29]. The next phase, buffering to short-term memory, can handle only 4–8 letters, preventing further look ahead [30]. Recognizable words may permit longer buffering (and hence greater speed). Participants subjectively confirmed their need to repeatedly check the next character. This reduces performance relatively more for faster typing, reducing any performance difference. This is a systematic issue with the new protocol—it is inherently conservative, underestimating performance differences between layouts.

If we assume that cognitive issues largely explain the difference between the estimates of methods 3 and 4, and filtering those between methods 1 and 3, we are left with an unexplained 9% difference between methods 1 and 2. However, this is a minor issue. The small sample size and large standard deviation of method 2 mean that we cannot reliably conclude that there is any improvement at all, much less that such improvement is 54% rather than 45% [18]. In addition to small sample size, method 2 is subject to an additional, difficult-to-quantify source of bias: it requires extrapolation of ultimate speed from early performance. This may partly underlie any remaining difference.

5) Results for Individual Word Pairs: Further evidence for the methodology's validity comes from an unexpected quarter.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
MEAN ELAPSED TIMES BY GROUPS (SECONDS)

	Gender		Age				ABC Proficiency (Rank)											
	M	F	≤ 20	21: 25	26 : 30	≥ 31	1: 10	11: 20	21: 30	31: 40	41: 50	51: 60	61: 70	71: 80	81: 90	91: 100	101: 110	111: 116
Sample Size	60	56	10	58	37	11	10	10	10	10	10	10	10	10	10	10	10	6
Mean ABC	8.02	8.31	**6.73**	**7.64**	**8.54**	**10.9**	**4.34**	**5.47**	**6.11**	**6.51**	**7.32**	**8.08**	**8.22**	**8.77**	**9.40**	**10.39**	**11.74**	**13.97**
p Value	0.39		0.00068				0 (< minimum representable)											
Mean Raw PM	6.45	6.61	5.14	6.16	6.88	8.55	3.51	4.42	4.80	5.44	6.10	6.26	6.55	6.97	7.54	8.50	9.11	10.85
Mean Adj. PM	6.5	6.66	**5.19**	**6.21**	**6.93**	**8.60**	**3.57**	**4.47**	**4.85**	**5.50**	**6.16**	**6.32**	**6.61**	**7.03**	**7.60**	**8.54**	**9.16**	**10.91**
p Value	0.52		0.00027				0 (< minimum representable)											
% Improv.	18.81	19.83	22.79	18.69	18.78	21.09	17.85	18.26	20.61	15.49	14.81	21.72	19.62	19.82	19.13	17.75	21.97	21.95
p Value	0.42		0.58				0.58											

TABLE V
PREDICTED TYPING SPEED IMPROVEMENT FOR PM LAYOUT UNDER DIFFERENT METHODOLOGIES

	Text Filtering	Test Method	Improvement	
			Mean	SD
1	No	Theoretical Analysis	45%	
2	No	User study – Direct Training	54%	35%
3	Yes	Theoretical Analysis	28%	
4	Yes	User study – our protocol	19.5%	7.5%

TABLE VI
MEAN $\pm$ SD OF MODEL-BASED FITNESS VALUES, FAMILIARITY ($\mathfrak{F}$), AND ELAPSED TIME (SECONDS)

	Fitness Value	Familiarity ($\mathfrak{F}$)	Elapsed Time
ABC Strings	2.26 ± 0.49	1.24 ± 0.38	8.14 ± 3.60
PM Strings	1.56 ± 0.42	1.25 ± 0.50	6.51 ± 3.28

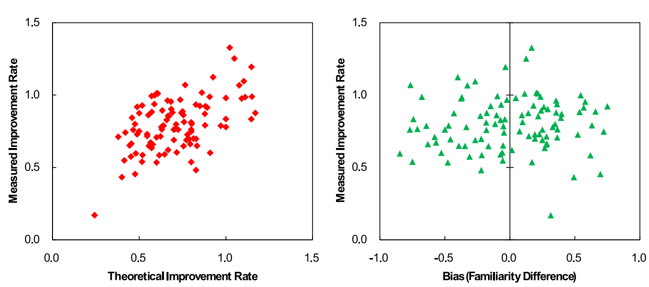


Fig. 8. Measured improvement rate versus (Left) model-based estimates for each word, (Right) familiarity ($\mathfrak{F}$) difference for each word.

In Table III, approximately $1/4$ were slower with PM than ABC. We investigated three possible sources for the differences:

1) The physical motions, which should be predictable by the physical model.
(2) Differences in familiarity ($\mathfrak{F}$), which should therefore be predictable from $\mathfrak{F}$.
(3) Other causes.

We can verify the first two by computing the relevant differences (fitness and $\mathfrak{F}$) for each pair, and comparing them to the differences in elapsed time. The overall statistics are shown in Table VI . In Fig. 8, we see a reasonable correlation (0.53) between measured and model-based estimates of improvement: a substantial portion of the difference is due to the physical differences between strings. While $1/4$ are slower in PM, all these cases are predicted by the physical model.

On the other hand, there is no correlation (0.01) between differences in $\mathfrak{F}$ and relative performance: we have successfully eliminated any influence of familiarity on our results.

However, we cannot completely exclude other influences on the results (since the correlation with the physical model was not 1.0). Much of the remaining variation is likely to be noise.

V. CONCLUSION

A. Advantages of the Proposed Method

We proposed a general screening methodology for comparing the speed of new key layouts with a previous familiar one. It is complementary to previous training-based methods, with major advantages in testing speed (hence rapid screening), and supporting more diverse testing. It offers the following advantages:

1) *Timeliness*: results are rapidly available. The new protocol required on average 6 min—at most 13—per person. The experiments in [18] required ~10 sessions, each ~10 min spaced over a few days to eliminate fatigue effects. For rapid development, these differences could be critical.
2) *Resource Reduction*: the protocol reduces the time requirements for experiments, as well as the equipment costs. Because of the speed of testing, we needed only one device, at a cost of ~US$70. Training methods would probably have required one device per person.
3) *Complementary Biases*: the protocol has some biases (principally, from inexact symmetries), but eliminates others arising in previous methods, notably the extrapolation required in training-based protocols. Calibrating between the two may be the best means to get a clear picture of overall biases, and thus to generate unbiased estimates.
4) *Wide Participation*: the elimination of training sessions reduces the commitment required by participants. With training-based methods, testees must invest substantial amounts of time. With the new protocol, it has been easy to find participants: we just need to explain what we are doing and ask "Can you spare 10 min?" We readily enrolled the 116 participants (29 data points for each set of string pairs). For the earlier, training-based experiments, we had to use "captive" participants, and to request a substantial investment of effort, limiting us to an unacceptably small sample of ten participants. Moreover, we may invite experts, skilled users, and novices, ignoring any differences in familiarity. We can form statistically useful subgroups, and compare performance by groups,

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

as in Section IV, providing both a statistically reliable analysis, and one with useful detail.

B. Assumptions and Limitations

We assume we are comparing two layouts for exactly the same keyboard. This assumption is often not precisely correct. The QWERTY and Dvorak layouts differ slightly, in which keys are used for alphabetic characters. Comparing ABC and PM, we had to adapt and compensate for slight differences in key use, but we were able to do so, and to quantify fairly accurately the effect of these adaptations. Nevertheless, the adaptation is not straightforward, and requires careful analysis.

We assume we can identify suitable invariants and symmetries preserving them. Since they will rarely be exact, we must quantify failures in invariance, either to compensate for them or estimate bounds on their effects. Thus, we assumed that motion scale was important, but direction unimportant; but there was some indication that the direction of vertical motion does affect speed, although not enough to change the conclusions.

The symmetries may also be inexact. Mobile phone keypads are typically completely symmetric (but users may not be). QWERTY keyboards typically are not, being generally staggered slightly so that horizontal reflections may change distances. The extent of such effects need to be quantified, and if possible compensated.

Finally, we tacitly assumed that language distributions on the keyboard are largely unaffected by symmetries. Even in English, this is not quite true—most vowels are on the top alphabetic row in QWERTY so that vertical reflection will move them to the bottom. Fortunately, English does not have a very systematic allocation of vowels and consonants—either can appear in almost any position in a word, the only real restriction being that long same-type sequences are rare. Resulting effects are similarly rare, and generally ignorable, but this is language specific. In Korean, the vowels and consonants are on opposite sides of the keyboard, and there are strict rules about their sequence. Thus, reflecting the keyboard would result in syllables that would not be accepted by the input method at all. Even if this were overcome (e.g., by linearizing the alphabet rather than writing it in syllable blocks), the resulting cadence would have an unnatural feel, and thus affect typing speed.

The work is limited in another way: it can only compare layouts with the same physical structure. With the proliferation of touch screens and "soft" keyboards, we are seeing completely new keypad structures for screen-based text input [31], or new physical ways of using old layouts [32]. Our methodology may not be appropriate for these.

The methodology underestimates long-term performance, due to cognitive load, as discussed in Section IV-D4. Underestimates are only a minor problem (getting a 30% improvement when we only expected 20% will be a pleasant surprise); therefore, we have not worried about compensating it, although it is relatively easy. The cognitive differences between "random-seeming" strings and recognizable language should be the same for the old and new layouts, but we can readily estimate the difference for the old layout: we have performance figures for the transformed strings, and can get them for the original strings. The same correction can then be applied to estimate the ultimate performance for the new layout.

1) Learnability Versus Ultimate Speed: In some senses, this is the most important limitation of our paper. While our methodology gives us good estimates of the likely typing speed of the new layout, it gives us no information about the learning effort. For already-skilled users, learning effort is often the dominant issue—and acceptability to existing users is usually the determining factor in acceptance of a new layout. New users, in principle, can concentrate on final speed, since they will have to invest learning effort for any layout, but cost and availability of input devices in the new format has often been a barrier in the past, even for new users. The rise of "soft" input methods, will probably make this less important in future [31], [33] so that acceptance of new layouts by new users may increase. Nevertheless, even in this software-based future, learnability will continue to be an important factor in acceptance, with more learnable layouts having an inherent advantage over less learnable. Thus, adoption of new devices can be seen as a two-objective optimization by users (learnability versus ultimate speed), with different users placing different weights on the two objectives according to their circumstances and goals.

Our methodology addresses only the issue of ultimate speed. It does not address learnability at all (and cannot readily be adapted to address it). Thus, it will probably be deployed in real applications in tandem with learnability analyses. Nevertheless, it forms a complement to the far-more-expensive learnability analyses. Its first benefit is as a screening test. If a new layout does not offer an advantage in ultimate speed, it will certainly fail (since however simple it is to learn, it cannot beat the incumbent in learnability) so that there is little point in conducting an expensive learnability test. Second, because our testing methodology reduces user time commitments, it can be applied to far wider ranges of users, thus providing more detailed population-specific data than can realistically be generated by learnability analyses.

C. Epilogue

Despite the limitations of the proposed method, it improves both ease and accuracy of layout comparison between "new" and "old" designs. Ease and accuracy are closely related, because we can conduct the easier experiment more times, increasing the statistical stability and eliminating important sources of bias.

In some cases, the methodology can replace previous approaches, especially if ultimate typing speed is the primary goal. More broadly, it complements learning-based methods, using our methodology to gain rapid assurance that a new layout is worth investigating. Once this is determined, it is straightforward to expand the sample pool to gain accurate understanding of the speed-up for different classes of users, in tandem with learning-based methods to more accurately estimate the overall learning effort. This has other benefits. Since the sources of bias are very different, the two methods can independently validate each other.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

APPENDIX A
CORPUS FAMILIARITY BIAS

A. Theory

Formally, we define the letter-frequency familiarity ($\mathfrak{F}$) of a string $s = (s_1, \ldots, s_n)$ of length n over a set L of letters as follows.

Definition 1: (Familiarity)

$$\mathfrak{F}(s) = \frac{1}{n}\sum_{k=1}^{n}\frac{p_T(s_k)}{p_U(s_k)} = \frac{|L|}{n}\sum_{k=1}^{n}p_T(s_k)$$

p_T is the frequency of each letter in the target alphabet,
p_U is the corresponding uniform distribution over L,
s_k is the kth character of string s,

where $\mathfrak{F}$ measures the excess independent-sample probability of the particular string, over sampling from a uniform distribution. It is unbiased with respect to string length (see Theorem 1).

Theorem 1: The expected value of $\mathfrak{F}(s)$ is independent of n, the length of string s.

Proof: Taking the expectation of $\mathfrak{F}(s)$ in Definition 1,

$$\mathbb{E}[\mathfrak{F}(s)] = \frac{|L|}{n}\mathbb{E}\left[\sum_{k=1}^{n}p_T(s_k)\right] = \frac{|L|}{n}\sum_{k=1}^{n}\mathbb{E}[p_T(s_k)]$$
$$= \frac{|L|}{n}\sum_{k=1}^{n}\frac{1}{|L|} = \frac{|L|}{n}\frac{n}{|L|} = 1. \qquad (8)$$

That is, the expected value of $\mathfrak{F}(s)$ is 1, independent of n.

Applying familiarity to each pair enables us to check letter-level fairness. If two strings transformed from a word in the corpus are equally familiar, they would not bias the overall results; if they differ substantially, they may affect them, requiring more samples to ensure fairness. As $\mathfrak{F}(s)$ is unbiased in string length see Theorem 1), we can directly compare the familiarity of pairs, ignoring string length in the criterion. We exclude string pairs with a large difference in familiarity.

We define the bias $\mathfrak{B}$ of a pair of strings (s_1, s_2) as follows.

Definition 2: (Bias)

$$\mathfrak{B}(s_1, s_2) = |\mathfrak{F}(s_1) - \mathfrak{F}(s_2)| \qquad (9)$$

$\mathfrak{B}$ lies in $[0 \ldots |L|]$ with smaller values for fairer comparisons. The distribution of $\mathfrak{B}$ may depend on the text corpus, and should be considered in choosing appropriate pairs. The simplest approach is to set a threshold, discarding pairs whose bias exceeds it.

B. Experiment Details

To determine a suitable threshold for dropping word pairs, we investigated the familiarity $\mathfrak{F}$ and bias $\mathfrak{B}$ distribution for three kinds of texts: the mobile phone messages used here, a magazine article [34], and random strings, as shown in Table VII and Fig. 9. In the histogram, we see that the bias $\mathfrak{B}$ is fairly uniformly distributed in general, but rises much more steeply from a little before the 90th percentile, at a bias a little under 1.0; we chose 0.85 as a threshold, coincidentally resulting in acceptance of about 85% of pairs.

TABLE VII
FAMILIARITY $\mathfrak{F}$ STATISTICS: MEAN±SD [MEDIAN]

Text Type	ABC Keypad	PM Keypad
Mobile messages	1.325 ± 0.396 [1.035]	1.163 ± 0.542 [0.726]
Magazine article	1.331 ± 0.320 [1.289]	1.176 ± 0.438 [1.182]
Random string	0.987 ± 0.361 [0.956]	1.012 ± 0.394 [0.975]

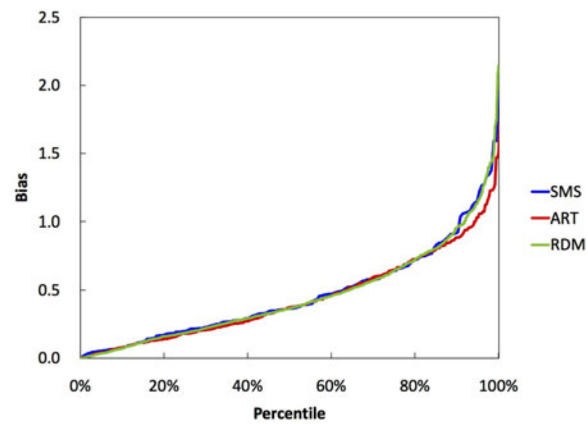


Fig. 9. Bias distribution for mobile messages (SMS), magazine article (ART), and random string (RDM). Vertical axis is for Bias $\mathfrak{B}$, and horizontal axis for cumulative percentile distribution.

TABLE VIII
LETTER FREQUENCY BEFORE AND AFTER LAYOUT CONVERSION

Original Letter			Converted Letter			Gap
Letter	Frequency	Key	Letter	Frequency	Key	
a	8.09%	2	t	8.97%	8	0.88%
b	1.47%	2	u	2.74%	8	1.27%
c	2.76%	2	v	0.97%	8	1.79%
d	4.21%	3	p	1.91%	7	2.30%
e	12.58%	3	r	5.93%	7	6.65%
f	2.20%	3	s	6.26%	7	4.06%
g	1.99%	4	m	2.38%	6	0.39%
h	6.03%	4	n	6.68%	6	0.65%
i	6.90%	4	o	7.43%	6	0.53%
j	0.15%	5	j	0.15%	5	0.00%
k	0.76%	5	k	0.76%	5	0.00%
l	3.99%	5	l	3.99%	5	0.00%
m	2.38%	6	g	1.99%	4	0.39%
n	6.68%	6	h	6.03%	4	0.65%
o	7.43%	6	i	6.90%	4	0.53%
p	1.91%	7	d	4.21%	3	2.30%
q	0.09%	1	x	0.15%	9	0.06%
r	5.93%	7	e	12.58%	3	6.65%
s	6.26%	7	f	2.20%	3	4.06%
t	8.97%	8	a	8.09%	2	0.88%
u	2.74%	8	b	1.47%	2	1.27%
v	0.97%	8	c	2.76%	2	1.79%
w	2.33%	9	.	0.99%	1	1.19%
x	0.15%	9	q	0.09%	1	0.06%
y	1.95%	9	z	0.07%	1	1.88%
z	0.07%	1	j	1.95%	9	1.88%
.	0.99%	1	k	2.33%	9	1.34%

APPENDIX B
EFFECTS OF SYMMETRIES ON TYPING SPEED

One way to test whether layout transformation changes typing speed is to compare letter frequencies before and after conversion, as shown in Table VIII , based on Beker and Piper's well-known letter frequency table [35]. For "." we used a result from [36] that average English word and sentence lengths are 3.6 letters and 18.9 words, giving an average sentence length of 85.9 letters. Row frequency is distributed 32.5%, 36.3%, and 31.2% for upper, middle, and lower (calculated by summing

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

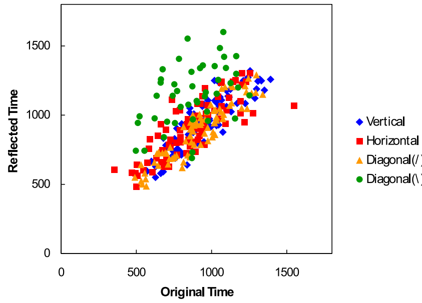


Fig. 10. Average elapsed time (milliseconds) for each letter pair (before and after reflection).

TABLE IX
EFFECT OF REFLECTION ABOUT KEY "5" BASED ON MOVE DIRECTION GROUPS (MILLISECONDS—MEAN ± SD))

	Direction 1		Direction 2		Gap
Horizontal	$\rightarrow$	949 ± 199	$\leftarrow$	945 ± 198	4 ± 96
Vertical	$\downarrow$	865 ± 224	$\uparrow$	901 ± 199	-36 ± 134
Diagonal(/)	$\swarrow$	909 ± 203	$\nearrow$	888 ± 199	21 ± 87
Diagonal(\)	$\searrow$	908 ± 211	$\nwarrow$	860 ± 211	48 ± 145
No move	$\circ$	1161 ± 192	$\circ$	1149 ± 219	12 ± 143
Overall		939 ± 224		929 ± 221	10 ± 124

original letter frequencies over each row). Vertical reflection only changes the frequency by 1.3%. Columns are distributed 30.2%, 29.9%, and 39.9% (left, middle, right). Exchanging left and right columns gives a 9.7% change. The number of strokes is not affected. In sum, the transformation generates only a relatively small change in the invariants.

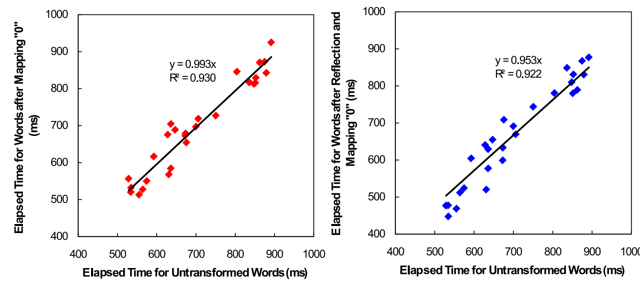
We also experimentally measured the effects on typing speed. Seven participants were asked to type each of the $27 \times 27 = 729$ possible bigrams (including the period "."). We measured the elapsed time. Ignoring the nine identity transformations (on key "5") gives us 360 pairs. Comparing the timing within each pair, averaged over all participants, gives us a measure of the effect of the transformation; if the invariants we have preserved are sufficient to preserve physical typing speed, we would expect these values to be equal.

Fig. 10 shows the elapsed time for each pair. If the reflection has no effect on typing speed, points should lie on the line $y = x$. Using regression, we get $y = 0.9814x$ with a coefficient of determination of 0.6887: a reasonable approximation to $y = x$, i.e., reflection has little effect. The low coefficient of determination could result either from noise or a systematic difference in timings. We split the points into five groups depending on their direction of movement: horizontal move (e.g., key "4" to "6"), vertical ("1" to "7"), NE-SW ("7" to "3"), E-NW ("4" to "9"), and no move ("5" to "5"). On inspection, the points appear to be distributed randomly about the $y = 0.9814x$ line, with the "no move" group at least as widely dispersed as the others. Table IX shows the mean and standard deviation for each group and overall. In each group, the mean gap is less than $1/3$ of the standard deviation so that the null hypothesis (that the gaps are randomly distributed) cannot be ruled out, and in fact seems highly likely.

Overall, reflection affects typing speed by less than 10 ms although there is a slightly larger effect—still only around 2%—for movements with a vertical component.

TABLE X
EFFECT OF KEY "0" MAPPING (IN MILLISECONDS)

	Mean ± SD	T-test Statistic
Untransformed Strings (f^{-1}_{ABC})	702 ± 124	
Mapping "0" ($f^{-1}_{ABC} \circ t$)	697 ± 127	0.14
Reflection and Mapping "0" ($f^{-1}_{ABC} \circ t \circ m$)	666 ± 137	0.99



(a) (b)

Fig. 11. Mean elapsed time for each letter pair: Untransformed versus transformed. (a) Mapped "0'. (b) Reflected and mapped "0."

APPENDIX C
EFFECTS OF BIJECTION CORRECTIONS

A. Remapping the "0" Key

To test whether the adapted mapping for the "0" key induces a bias, we used a similar setup to Appendix B, with 12 participants. We measured time to type each pair consisting of a letter followed by key "0" (space), for space moved to key "8" (i.e., "t"), and for the combined effect of transformation t and moving key "0" to key "8." The results are summarized in Table X, and we see the timing pairs for untransformed strings versus mapping "0" alone, and reflection plus mapping "0," in Fig. 11(a) and (b). Points are located near the line $y = x$: the transformation had little effect on typing speed.

B. Remapping Four-Stroke Characters

To measure the effect of mapping four strokes to three, we computed the frequency of four-stroke characters in the test set (3.78%—about 35 out of 925 characters for each group). From the data gathered in symmetry measurement experiments (see Appendix B), we computed the mean excess time for typing a 2-stroke character over the corresponding 1-stroke character (156.2 ms), and for 3-stroke over 2-stroke (171.7 ms). Thus, we can estimate the excess for the 4-stroke characters that should have been typed over the 3-stroke characters that were actually typed at around 160 ms per character: so the PM typing time for each group of four participants should be penalized by $160 \times 35 = 5600$ ms for 35 4-stroke characters, giving a correction of 1400 ms per person or 56 ms per string pair.[7]

[7]This is compatible with the estimate in [37], that 95% of keystrokes are completed within 330 ms, and 83% within 125 ms.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

(a) Samsung SCH-M470 (b) User Interface

Fig. 12. Phone model and user interface used in the experiment.

APPENDIX D
EXPERIMENTAL DEVICE AND USER INTERFACE

To perform the experiments, we needed a programmable phone with an ABC keypad: now a rare combination. We resorted to a quite old device: a Samsung Electronics model SCH-M470. The physical size is $101.5 \times 53 \times 16.8$ mm. It uses the Qualcomm MSM7200 chipset running at 400 MHz, with 64 MB RAM and 256 MB ROM, running Microsoft Windows Mobile Version 6.0 Professional. The program was developed in C++, using Microsoft Visual Studio 2008. The exterior appearance of the phone, and the user interface for our experiments, are shown in Fig. 12.

ACKNOWLEDGMENT

The Institute for Computer Technology of Seoul National University provided research facilities for the study.

Some figures[2] in this paper were adapted from wikipedia:

http://en.wikipedia.org/wiki/File:KB_South_Korea.svg

http://en.wikipedia.org/wiki/File:KB_North_Korea.svg

http://en.wikipedia.org/wiki/File:KB_United_States-NoAltGr.svg

http://en.wikipedia.org/wiki/File:KB_United_States_Dvorak.svg

They are used under a Creative Commons Attribution—Share Alike 3.0 Unported license:

http://creativecommons.org/licenses/by-sa/3.0/deed.en

They are provided by the authors under the same condition.

REFERENCES

[1] D. G. Alden, R. W. Daniels, and A. F. Kanarick, "Keyboard design and operation: A review of the major issues," *Human Factors*, vol. 14, no. 4, pp. 275–293, 1972.

[2] A. Dvorak, "There is a better typewriter keyboard," *National Business Education Quarterly*, vol. 12, no. 2, pp. 51–88, 1943.

[3] H. Jung, "Development of a common korean computer keyboard layout for south and north korea," in *Proc. Int. Conf. Korean Language Inform. Sci. Soc.*, 1996, no. 7, pp. 50–66.

[4] I. S. MacKenzie, R. B. Nonnecke, J. C. McQueen, S. Riddersma, and M. Meltz, "A comparison of three methods of character entry on pen-based computers," *Human Factors Ergonomics Soc. Annu. Meeting Proc.*, vol. 38, no. 5, pp. 330–334, 1994.

[5] J. R. Lewis, M. J. LaLomia, and P. J. Kennedy, "Evaluation of typing key layouts for stylus input," *Human Factors Ergonomics Soc. Annu. Meeting Proc.*, vol. 43, no. 5, pp. 420–424, 1999.

[6] N. Green, J. Kruger, C. Faldu, and R. St. Amant, "A reduced qwerty keyboard for mobile text entry," in *Proc. Extended Abstracts Human Factors Comput. Syst.*, Apr. 2004, pp. 1429–1432.

[7] A. Mittal and A. Sengupta, "Improvised layout of keypad entry system for mobile phones," *Computer Standards Interfaces*, vol. 31, no. 4, pp. 693–698, 2009.

[8] L. Butts and A. Cockburn, "An evaluation of mobile phone text input methods," *Aust. Comput. Sci. Commun.*, vol. 24, no. 4, pp. 55–59, 2002.

[9] J. Gong and P. Tarasewich, "Alphabetically constrained keypad designs for text entry on mobile devices," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2005, pp. 211–220.

[10] R. S. Hirsch, "A human factors comparison of two keyboards, proposed for special purpose and limited application," IBM, Tech. Rep. RJ151, San Jose, CA, 1958.

[11] R. T. Griffith, "The minimotion typewriter keyboard," *J. Franklin Inst.*, vol. 248, pp. 399–436, 1949.

[12] J. Harnett, "Qwerty lives another day," *Office Equipment News*, vol. 15, no. 16, p. 57, 1972.

[13] S. E. Michaels, "Qwerty versus alphabetic keyboards as a function of typing skill," *Human Factors: J. Human Factors Ergonom. Soc.*, vol. 13, no. 8, pp. 419–426, 1971.

[14] B. Thomas, S. Tyerman, and K. Grimmer, "Evaluation of three input mechanisms for wearable computers," in *Proc. IEEE Int. Symp. Wearable Comput.*, 1997, pp. 2–9.

[15] I. S. MacKenzie and S. X. Zhang, "The design and evaluation of a high-performance soft keyboard," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, May 1999, pp. 25–31.

[16] M. Ingmarsson, D. Dinka, and S. Zhai, "TNT: A numeric keypad based text input method," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Apr. 2004, pp. 639–646.

[17] S. Hwang and G. Lee, "Qwerty-like 3x4 keypad layouts for mobile phone," in *Proc. Extended Abstracts Human Factors Comput. Syst.*, Apr. 2005, pp. 1479–1482.

[18] J. Lee and R. I. B. McKay, "Optimizing a personalized multigram cellphone keypad," Seoul National University Structural Complexity Laboratory, Seoul, Korea, Tech. Rep. TRSNUSC:2014:001, Nov. 2014. [Online]. Available: http://arxiv.org/abs/1412.1180

[19] J. Lee, H. Cho, and R. I. B. McKay, "Rapid screening of keyboard layouts," in *Proc. Conf. Syst., Man Cybern.*, Oct. 2012, pp. 894–899.

[20] I. MacKenzie and R. Soukoreff, "A model of two-thumb text entry," in *Proc. Graphics Interface*, 2002, pp. 117–124.

[21] P. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *J. Exp. Psychol.*, vol. 47, no. 6, p. 381, 1954.

[22] E. Clarkson, K. Lyons, J. Clawson, and T. Starner, "Revisiting and validating a model of two-thumb text entry," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2007, pp. 163–166.

[23] S. Moradi and A. Nickabadi, "Optimization of mobile phone keypad layout via genetic algorithm," in *Proc. 2nd Inform. Commun. Technol.*, vol. 1, 2006, pp. 1676–1681.

[24] J. Lee and B. McKay, "Optimizing a personalized cellphone keypad," in *Proc. Int. Conf. Convergence Hybrid Inform. Technol.*, vol. 6935, 2011, pp. 237–244.

[25] P. Olver, *Equivalence, Invariants, and Symmetry*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[26] W. Hünting, T. Läubli, and E. Grandjean, "Constrained postures of VDU operators," in *Proc. Workshop Ergonom. Aspects Visual Display Units*, 1980, pp. 175–184.

[27] W. C. Maxwell, "The rhythmic keyboard," *J. Business Educ.*, vol. 27, pp. 327–330, 1953.

[28] W. E. Cooper, "Introduction," in *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper, Ed. New York, NY, USA: Springer-Verlag, 1983, pp. 1–38.

[29] R. Butsch, "Eye movements and the eye-hand span in typewriting," *J. Educ. Psychol.*, vol. 23, no. 2, pp. 104–121, 1932.

[30] P. Buzing, "Comparing different keyboard layouts: Aspects of qwerty, dvorak and alphabetical keyboards," Delft Univ. Technol., Delft, The Netherlands, Tech. Rep. 60, 2003.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.

[31] S. B. Nesbat, “A system for fast, full-text entry for small electronic devices,” in *Proc. Int. Conf. Multimodal Interfaces*. Nov. 2003, pp. 4–11.

[32] C. A. Kushler, and R. J. Marsden, “System and method for continuous stroke word-based text input,” U.S. Patent 7 098 896, 2006.

[33] M. Raynal and N. Vigouroux, “Genetic algorithm to generate optimized soft keyboard,” in *Proc. Extended Abstracts Human Factors Comput. Syst.*, Apr. 2005, pp. 1729–1732.

[34] A. Altman. (2009). Why air travel is about to get worse. *Time Mag*. [Online]. Available: http://content.time.com/time/business/article/0,8599,1929324,00.html

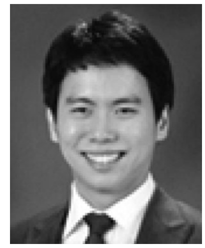
[35] H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*. New York, NY, USA: Wiley-Interscience, 1982.

[36] B. Sigurd, M. Eeg-Olofsson, and J. Van Weijer, “Word length, sentence length and frequency—zipf revisited,” *Studia Linguistica*, vol. 58, pp. 37–52, 2004.

[37] R. Kinkead, “Typing speed, keying rates, and optimal keyboard layouts,” *Human Factors Ergonomics Soc. Annu. Meeting Proc.*, vol. 19, no. 3, pp. 159–161, 1975.

Joonseok Lee is currently working toward the Ph.D. degree in computer science, studying in Statistical Machine Learning and Visualization Lab, Georgia Institute of Technology, Atlanta, GA, USA. He received the B.S. degree in computer science and engineering from Seoul National University, Seoul, Korea, in 2009.

His research interests include machine learning and data mining, especially in recommendation systems and collaborative filtering.



Hanggjun Cho (S’12) received the M.S. degree in computer science and engineering from Seoul National University, Seoul, Korea, in 2014.

He is currently with the Structural Complexity Lab. His research interests include human–computer interaction and natural language processing.

R. I. (Bob) McKay (SM’00) received the B.Sc. degree from the Australian National University, Canberra, Australia, in 1971, and the Ph.D. degree in theory of computation from the University of Bristol, Bristol, U.K., in 1976.

He has worked with CSIRO and the University of New South Wales, and joined Seoul National University, Seoul, Korea, in 2005. His research interests include intelligent systems, evolutionary computation, and ecological modeling.

Authorized licensed use limited to: COLORADO STATE UNIVERSITY. Downloaded on May 04,2024 at 00:00:49 UTC from IEEE Xplore. Restrictions apply.