Comparing Typing Speeds for Keyboard Layouts Fairly and Efficiently

Joonseok Lee¹ and R.I. (Bob) McKay²

¹: JS Lee is a Computer Scientist with an interest in intelligent systems and human-computer interaction; he is a research student in College of Computing, Georgia Institute of Technology.

²: Bob McKay is a Computer Scientist with interests in evolutionary computation and its applications, with particular reference to adaptive support for human-computer interaction; he is an associate professor in the Structural Complexity Laboratory, School of Computer Science and Engineering of Seoul National University.

Corresponding Author: Bob McKay
Email: rimsnu@seoul.ac.kr

Abstract

With the advent of ubiquitous computing, we have seen a proliferation of input modes for many kinds of devices. Notably, with the advent of programmable devices such as smart phones, we can easily design and program new layouts in software. Without a standard methodology for evaluating these new layouts, user interface designers have had to design and conduct individual tests for each new design.

To ease this problem, we propose an efficient, accurate, and general-purpose methodology for evaluating the speed of a new layout for a specific physical input device, assuming the prior existence of a standard layout which is used as a comparator. Specifically, it overcomes the problems of relative familiarity and finger memory in a new way, by appropriately mapping both new and old layouts.

To illustrate the proposed methodology, we provide an application: comparing a new mobile phone key layout with the standard ABC key layout for 12-key mobile phone keypads. As well as explaining the method, this application demonstrates the accuracy and simplicity of our proposed evaluation method.
## Contents

### Introduction

### Background
- Ignoring the Familiarity Issue
- Restricting Testing to Novices
- Provision of Training Sessions
- Detailed Observation of the Learning Progress

### Methodology
- Familiarity Issues
- Handling Location Memory through Character Mapping
- Removing Finger Memory Bias using Layout Transformation
- Test Set Generation
- Overall Experimental Protocol

### Test Example: Mobile Phone Keypad Layout
- Applying the Test Protocol
- Adapting the Methodology
- Familiarity Difference (Bias) Threshold
- Experimental Comparison: Settings
- Result and Analysis

### Discussion

### Conclusion
- Advantages of the Proposed Method
- Assumptions and Limitations
- Epilogue

### NOTES

### References
Many forms of device have keyed input mechanisms – whether physical or virtual. These are primarily used for character input (though other forms, such as musical input, also exist). A core question, with any such mechanism, is what layout to use. That is, given the pattern of keys, how should the input space – the characters – be allocated to the keys. In many cases, competing layouts have been proposed. For computer keyboards, for example, the QWERTY layout is widely used for English as a 'de facto' standard. (Alden, Daniels, & Kanarick, 1972). However, it competes with the more efficient Dvorak layout. Another example can be seen in Korea. South and North Korea use quite different keyboard layouts for the same language, see Figure 1 (Jung, 1996).

In addition to these well-known cases, we often encounter the need to comparatively evaluate two different keyboard layouts - frequently in the circumstance where one layout is well-known, and the other unfamiliar. Most notably, many researchers have aimed to optimize layouts for specific languages or for other kinds of special purpose. Today, there is an explosion of proposed new keyboard layouts, especially in mobile phones. Hundreds have been suggested, implemented, and distributed on many kinds of mobile phones. But to win acceptance for a new, potentially more efficient layout, it is essential to have a methodology for comparative evaluation against the previous design it proposes to replace. Objectively showing that a new design is more efficient, by a standard, verified method, will substantially enhance its influence.

Figure 1. Korean Keyboard Layout (Top: South Korea, Bottom: North Korea)
We present a methodology for testing ultimate speed of newly-designed layouts, based on a comparison with an existing (familiar) one. Assuming that we can easily find skilled users in a standard layout, we base the method on that layout, because we are primarily interested in the efficiency that the new layouts will achieve after training and familiarization, not their efficiency for novice users. As in previously used methods, we compare the elapsed time to type the same text with each layout. However we adapt the comparison to eliminate some important sources of bias.

There are several obstacles to fair comparisons. The users’ familiarity with the current layout means that a direct comparison with the new layout will certainly be unfair. The core problem is precisely the familiarity of the users with the ‘old’ layout. Users know exactly where each key is, whereas they have to search for each key on the new layout. Thus it is totally unfair to compare the time taken to type the same raw text: the old layout will certainly take less time, even if the new layout is actually substantially more efficient (after training).

There are two obvious solutions, which unfortunately both have serious limitations: using only complete novices for testing (so that both layouts are equally unfamiliar), or providing the test participant sufficient time for complete familiarization with the new layout.

In the former case, it may be difficult to find complete novices. For example, it would be hard to find people who are completely unfamiliar with the QWERTY layout (Noyes, 1983). Even when they are available, they are likely to be an atypical sample in other ways – young children or non-native speakers. Moreover no feasible experiment can approximate the long-term behavior of trained users. For example, it takes months or even years of substantial effort to become highly proficient on a QWERTY layout. A week’s training – typical in such experiments – can hardly be expected to provide an accurate estimate (few typing students attain touch typing proficiency in a week, for example).

The latter approach is even more problematic, since it generally involves an assumption that the ”convergence” seen after a period of training – again, often a week – is a reliable estimator of likely speed after long-term training. Of course, this ”convergence” is nothing of the sort – typing speed after one week’s training may level off, but it is a long way from the speed that a practiced typist will eventually attain (as anyone who has learnt QWERTY will be aware). As we discuss in Section , there is some question whether the speed after a short training period is a useful estimator of ultimate speed.

To overcome these problems, we suggest a protocol enabling a more impartial 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 personalized mobile phone keypad layout with the standard ”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, and its possible future extensions.

Background

Over the last two decades, many authors have suggested new layouts for various kinds of character input device. They usually evaluated its validity through human testing. Asking a group of people to type some standard texts with both a standard and the new
layout, and comparing the elapsed time (i.e. the second approach listed above), was the most widely-used methodology. While many authors made efforts to deal with the difference in familiarity, the extent to which they were able to eliminate it remains open to serious question.

In order to evaluate previous testing protocols, we set several conditions which an ideal testing methodology should satisfy:

1. As far as practicable, the effects of any difference in familiarity between the two layouts should be eliminated.
2. Comparisons should be at as close-to-expert level as is achievable.
3. The evaluation methodology should not make it difficult to recruit a large number of participants from a wide variety of backgrounds.
4. In particular, the testing regime should not require a large investment of time on the part of the participants (since this is likely to conflict with the previous criterion).

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

- Ignoring the familiarity issue
- Restricting testing to novices
- Providing training session
- Observing learning progress

Unfortunately, none of these strategies can meet all our criteria, as discussed below.

**Ignoring the Familiarity Issue**

The easiest way to deal with the familiarity issue is to construct participant groups independent of their familiarity with the current layout. In the analysis phase, one simply ignores the issue, and compares performance without regard to familiarity (thus unfairly biasing the analysis toward the more familiar layout). Unfortunately this may be difficult to avoid when comparing physically dissimilar keyboards. A number of researchers have used this approach when their primary concern was the initial acceptance of new design, rather than the ultimate speed. In this case, we may be able to conclude that the new design is acceptable even if it is significantly slower than the control.

MacKenzie et al. (MacKenzie, Nonnecke, McQueen, Riddersma, & Meltz, 1994) used such a method for comparing three text input methods on a pen-based computer (hand printing, QWERTY-tapping, and ABC-tapping). They tested 15 participants, without providing any training time. They argued that it was acceptable to not provide training session as they were interested in the walk-up acceptance.

Lewis et al. (Lewis, LaLomia, & Kennedy, 1999) used a similar strategy for evaluating key layouts for stylus input. A total of 12 participants, well-experienced with QWERTY, participated in the experiments. In the discussion section, the authors acknowledged the unfairness of the comparison, commenting that "typing speed results indicate that participants lowered their typing speed when using unfamiliar layouts."

Green et al. (Green, Kruger, Faldu, & St. Amant, 2004) used this method to evaluate stick keyboards (both multi-tap and lexicon types) against a QWERTY layout. Green gathered 10 participants, who were familiar with the QWERTY keyboard, but not with multi-tap methods. After a small number of experiments, they merely commented that
stick keyboards were acceptable in comparison with QWERTY. Since they were trading off portability against speed, no deeper comparison was needed.

This approach, of simply ignoring familiarity issues, is sometimes adopted because of lack of data on the control input method. Mittal and Sengupta (Mittal & Sengupta, 2009) tested their improvised mobile phone keypad layout with 6 participants. They noted that it was impossible to compare learning curves with existing layouts due to the absence of studies on their learning curves.

With this strategy, it is relatively easy to conduct experiments in a short time. Thus, it makes it easy to invite a large number of participants, although the majority of previous studies have tested only a small number of participants (5 to 20). That is, criteria 3 and 4 are satisfied. However criteria 1 and 2 are explicitly ignored through disregarding any difference in familiarity between layouts, so we would just be able to conclude that our new design is acceptable, not that it will ultimately be faster.

Restricting Testing to Novices

When we are unable to test the competing layouts at expert level, using novices for both layouts can be the fairest way of comparison. Unfortunately, however, it may be extremely difficult to find novice users for the current standard layout because of its ubiquity. This method may also result in serious bias because novices with the standard layout may be atypical users. Nevertheless, some previous research tried this approach, mainly due to its simplicity and fairness.

Hirsch (Hirsch, 1958) used 55 non-typists in his experiment comparing typing speeds between QWERTY and Griffith’s Minimotion (Griffith, 1949) layouts. After a pre-trial of the experiment, only 40 of the novices were selected for the actual trial. Hirsch anticipated that participants would be faster with the Minimotion layout due to its easily-recognizable alphabetical order. Nonetheless, the results indicated that novice users were good at QWERTY. Hirsch concluded that the participants might have had some experience on QWERTY standard, so he acknowledged that his intention of inviting complete novices at QWERTY was not fulfilled.

The learnability evaluation of Dvorak layout conducted by Harnett (Harnett, 1972) used a similar method. Harnett reported that a novice user can reach up to 50 WPM (words per minute) in 5 hours of training. Another participant who could type 50 WPM with QWERTY was able to reach 35-40 WPM with Dvorak after 4 hours of training.

As there is no familiarity difference between layouts for novice participants, this approach satisfies criterion 1. It also does not require too much time and resources, so satisfies criterion 4. However, criterion 2 (expert-level comparison) is explicitly ignored. Criterion 3 is also difficult to fulfil, because it is generally difficult to find perfect novices for a widely-used standard layout.

Provision of Training Sessions

A minor variant of the first strategy provides a little time for participants to familiarize themselves on the new layout before evaluation. Although participants may experience a large difference in familiarity between the two layouts at first, they may narrow the gap before the actual test. Previous research in this category generally allowed a very short
Butts and Cockburn (Butts & Cockburn, 2002) evaluated multi-tap mobile phone keypads against the two-key method. They provided sufficient time for participants to practice on each keyboard until they felt comfortable. They noted that this training time was usually less than one minute, even for those without any prior experience. Strictly speaking, providing training sessions cannot practically eliminate the difference in familiarity between keyboards. They defined "familiarity" by the subjective feeling of each participant, rather than a measurable index. There was no real attempt to measure whether each participant was sufficiently familiar with each layout to ignore the familiarity difference. Indeed, given the wide spread of multi-touch layouts at that time, one might reasonably be skeptical whether participants were really estimating their equal-familiarization time, or perhaps their boredom threshold.

Gong and Tarasewich (Gong & Tarasewich, 2005) provided a short training time for evaluating their alphabetically constrained mobile phone layout in comparison with an unconstrained optimized one and with ABC-layout. The training session provided two sample sentences to be typed. They did not check whether the training time was enough for each participant, whether by subjective feeling or an objective index. Nevertheless, this minimal training time may have reduced the large errors occurring in the very first use of a layout.

In short, this approach has both pros and cons. It satisfies criteria 3 and 4, since it does not require much extra time or other resources for practice. It may not perfectly fulfil criterion 1, but does try to reduce the gap. Criterion 2, however, is not attainable, as it is impossible for the participants to practice sufficiently within the short time.

**Detailed Observation of the Learning Progress**

The final option is to provide a sufficiently long training time for participants to make substantial learning progress. Once the learning rate starts to tail off, the speed is plotted against training time, and this graph is used to estimate the optimum typing speed that each participant will eventually attain. Progress through time can be compared between the old and new layout even though participants may have some experience in the old design. At the start, we can generally expect a higher speed from the old layout, but the new design may overtake it in the long run. From such data, we may (if we are bold) conclude that the new layout will eventually be superior to the current standard based on this plot.

Michaels compared an alphabetically-ordered computer keyboard layout with the QWERTY standard (Michaels, 1971). He tested 30 participants over 25 sessions for both layouts, concluding that alphabetical order has no advantage over QWERTY.

Thomas et al. (Thomas, Tyerman, & Grimmer, 1997) used a similar approach in comparatively evaluating three kinds of wearable computer (forearm mounted keyboard, virtual keyboard, and Kordic keypad). They provided an hour of training time, conducting 6 sessions over 3 weeks.

MacKenzie and Zhang (MacKenzie & Zhang, 1999) compared a new mobile keypad against the QWERTY layout. Five participants took part in 20 experimental sessions, spread over a week, and each lasting 45 minutes. They plotted the results, concluding
Table 1: Evaluation of traditional Keyboard Comparison Approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Fairness</th>
<th>Expert-level</th>
<th>Scalability</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignoring the familiarity issue</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Restricting testing to novices</td>
<td>○</td>
<td>×</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td>Providing training session</td>
<td>×</td>
<td>×</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Observing learning progress</td>
<td>○</td>
<td>○</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

that the new design was much easier to learn, and that it would be faster after a sufficient amount of practice (around 15 sessions).

Ingmarsson et al. (Ingmarsson, Dinka, & Zhai, 2004) applied this strategy in testing television-based appliances. Five participants were involved, over ten experimental sessions.

Hwang and Lee conducted a similar experiment on a QWERTY-like mobile phone keypad layout (Hwang & Lee, 2005). In evaluation, they conducted 5 sessions of experiments with 20 participants over 5 days. Based on similar plots, they argued that the QWERTY-like layout was not only easy to learn, but also ultimately faster than the more common ABC-layout.

We also previously employed this method in (Lee & McKay, 2010), with 10 participants, to evaluate a Personalized Multigram (PM) layout against the ABC 12-key layout. We conducted sufficient repetitions of training with the PM layout for each participant to overtake their initial ABC speed, and drew the conclusion that the PM keypad is more efficient through comparing progress in typing speed.

Although these examples appear successful in validating the new designs, there are a number of limitations. Most important is the experimental cost – around a week of participants’ time. Thus condition 4 is not satisfied. Participants are required to dedicate themselves to the experiment for a substantial amount of time, while the three earlier strategies require from each participant only about an hour of effort. Moreover, in many scenarios, we need a specific machine or equipment for testing. This results in a resource limitation, restricting the pool of participants to very small numbers of people. For these reasons, the preceding experiments used only very narrow pools of participants – mostly laboratory members. This limitation, however, can cause serious problems in statistical reliability. In (MacKenzie & Zhang, 1999), the number of participants was limited to 5. All were CS (Computer Science) majors, and only one was female. Although the results seemed to show enhanced performance from the new design, it may be reliably extended only to CS-major male graduate students, covering a very small segment of society. In other words, condition 3 is not satisfied.

As discussed above, none of the four strategies widely-used for testing can guarantee to satisfy all of our criteria. In order to conduct effective and reliable experiments, we need a new protocol which can meet all of these criteria.

Methodology

Familiarity Issues

There are two main kinds of familiarity issue in comparing keyboard layouts. First, participants may have different levels of recollection of where each key is located. An expert
user can immediately and unconsciously locate each key to be typed. A medium-experience user may be able to recollect where each key is located quickly, and be able to find it in around a second. A novice may have little or no idea about the layout, and so waste a long time just looking for each key. Such variation is common for the current standard layouts such as QWERTY. For a new design, on the other hand, no one knows where to find target characters.

To take another example, an expert with the ABC mobile phone layout could locate the letter ‘t’ immediately, moving the thumb to key #8 and pressing it once. A novice may take some time to figure out the alphabetical order and eventually find ‘t’ on the keypad. The main difference between the two participants is the time necessary to locate the key, rather than the time required to complete the movement or press the key.

If we are to undertake fair comparisons, we need to remove this gap, and to standardize the level of recollection to a higher level, because keyboard efficiency is only highly relevant when it is based on sufficiently trained participants. Of course, layouts may have different levels of learnability, taking different amounts of time for people to become expert. This issue is beyond the scope of this paper.

A second category of familiarity is finger memory. Frequently used phrase or multi-grams are easy to remember not only because of the memory of individual key locations, but because of finger memory of the sequence on the keyboard. For example, most experienced QWERTY users do not need to separately locate the ‘t’, ‘h’, and ‘e’ on the keyboard to type ‘the’ in English. The more experienced is the user, the more such words may be remembered through finger memory of the sequence. Thus we also need to eliminate bias due to this kind of memory.

To deal with such familiarity issues, and to remove the bias effect, there are two alternatives: standardization to the higher level of an experienced user (equilibrating the less-known layout to the memory level of the familiar layout); or to the lower level of a novice, (penalizing the memory of the expert down to the same level as the new user). For the first disparity, memory of where the characters lie on keyboard, we propose a compensation approach, applying a character mapping method which transfers the user’s knowledge of the familiar layout to the new layout. For the latter problem, finger memory, we use a penalty strategy, eliminating any benefit from finger memory from both layouts.

Handling Location Memory through Character Mapping

In order to estimate the expected time required to correctly type a given text, we need to remove the unnecessary delay of looking for keys. It is not difficult to find experimental participants who are sufficiently familiar with the current layout, but it is practically infeasible to bring them to the same level of familiarity with a new design in a short period of time. Thus, we need a methodology to transfer their proficiency with the current keyboard to the new design. We detail the proposed methodology here, but note that this is just a general description, so it may need further adaptation to specific cases. To explain the method, we need to define some terminology. First, we need to assign a unique number to each key, the keystroke code. This can be coded in various ways, depending on the physical device. For a computer keyboard, for example, an integer value may be assigned to each of around 100 keys. For a multi-tap mobile phone keypad, we need to code both the key and stroke location, to fully specify each keystroke.
Formally, we define the function

\[ f_{\text{layout}}(l) : L \rightarrow C \]  

which maps a letter \( l \) to a keystroke code in the given layout, where, \( L \) is the set of letters in the target language and \( C \) is the set of keystroke codes. This mapping \( f_{\text{layout}} \) is bijective: that is, each alphabet letter in the target language \( L \) corresponds to exactly one keystroke in \( C \). It therefore has an inverse bijection

\[ f_{\text{layout}}^{-1}(c) : C \rightarrow L \]  

which, conversely, maps a key code \( c \) to the corresponding character in the target language, for the specific layout.

Through iterative application, \( f_{\text{layout}} \) specifies the sequence of keystrokes corresponding to a character string, and \( f_{\text{layout}}^{-1} \) specifies the character string that a given sequence of keystrokes will produce. Since no ambiguity will result, we use the same notation (\( f \) and \( f^{-1} \)) to denote mappings between strings of characters and the corresponding sequences of keystrokes.

Suppose we have an 'old' layout and a 'new' one that we wish to compare; we assume that our experiment participants are familiar with the 'old', in the sense that they know \( f_{\text{old}} \) instinctively – that given a string \( s \), they can immediately type \( f_{\text{old}}(s) \). On the other hand, they cannot immediately find \( f_{\text{new}}(s) \). Of course, this leads to unfair comparisons. But suppose that, instead, we ask them to type \( f_{\text{old}}^{-1}(f_{\text{new}}(s)) \) using the old keyboard. That is, they are asked to apply the mapping \( f_{\text{old}} \) (which they know) to the string \( f_{\text{old}}^{-1}(f_{\text{new}}(s)) \). What they will actually do is produce the string

\[ f_{\text{old}}(f_{\text{old}}^{-1}(f_{\text{new}}(s))) = f_{\text{new}}(s) \]

which is what we desired – but using their current knowledge of the 'old' layout.

To make this more concrete, assume that we want to compare the Dvorak layout (assuming that this is a generally unfamiliar layout) with the familiar QWERTY layout, using "computer" as the test word. We may use this string "computer" directly to measure speed in the QWERTY layout, but we need the transformation described above for Dvorak. Thus we apply the nested function:

\[ f_{\text{qwertty}}^{-1}(f_{\text{dvorak}}("\text{computer}")) \]

using, for example, the key coding shown in Figure 2, and the layouts shown in Figure 3. \( f_{\text{dvorak}}("\text{computer}" \) returns the sequence of \(<1\text{H}, 2\text{B}, 3\text{G}, 1\text{D}, 2\text{D}, 2\text{H}, 2\text{C}, 1\text{I}\>>\). Applying \( f_{\text{qwertty}}^{-1} \) then returns the string "ismrfkdo". So measuring the time to type "ismrfkdo" using QWERTY layout tells us about the physical time required to type "computer" in the Dvorak layout – because they are exactly the same physical sequences – without requiring our participants to know Dvorak. Of course, there are remaining issues relating to the relative familiarity of "computer" and "ismrfkdo", and finger memory of sequences such as "put" but not "rfk"; we deal with these issues next.
**Figure 2.** Example of Key Coding

<table>
<thead>
<tr>
<th></th>
<th>!</th>
<th>@</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>^</th>
<th>&amp;</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>{</th>
<th>}</th>
<th>Backspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>1A</td>
<td>1B</td>
<td>1C</td>
<td>1D</td>
<td>1E</td>
<td>1F</td>
<td>1G</td>
<td>1H</td>
<td>1I</td>
<td>1J</td>
<td>1K</td>
<td>1L</td>
<td>1M</td>
</tr>
<tr>
<td>Caps Lock</td>
<td>2A</td>
<td>2B</td>
<td>2C</td>
<td>2D</td>
<td>2E</td>
<td>2F</td>
<td>2G</td>
<td>2H</td>
<td>2I</td>
<td>2J</td>
<td>2K</td>
<td>Enter</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>3A</td>
<td>3B</td>
<td>3C</td>
<td>3D</td>
<td>3E</td>
<td>3F</td>
<td>3G</td>
<td>3H</td>
<td>3I</td>
<td>3J</td>
<td>Shift</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ctrl</th>
<th>Win Key</th>
<th>Alt</th>
<th>Alt Gr</th>
<th>Win Key</th>
<th>Menu</th>
<th>Ctrl</th>
</tr>
</thead>
</table>

**Figure 3.** Character Mapping Example: Dvorak and QWERTY

<table>
<thead>
<tr>
<th></th>
<th>!</th>
<th>@</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>^</th>
<th>&amp;</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>{</th>
<th>}</th>
<th>Backspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>~</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>[</td>
<td>]</td>
</tr>
<tr>
<td>Caps Lock</td>
<td>&quot;</td>
<td>&lt;</td>
<td>&gt;</td>
<td>P</td>
<td>Y</td>
<td>F</td>
<td>G</td>
<td>C</td>
<td>R</td>
<td>L</td>
<td>?</td>
<td>/</td>
<td>=</td>
</tr>
<tr>
<td>Shift</td>
<td>A</td>
<td>O</td>
<td>E</td>
<td>U</td>
<td>I</td>
<td>D</td>
<td>H</td>
<td>T</td>
<td>N</td>
<td>S</td>
<td>-</td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>Ctrl</td>
<td>~</td>
<td>!</td>
<td>@</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>^</td>
<td>&amp;</td>
<td>*</td>
<td>(</td>
<td>)</td>
<td>{</td>
<td>}</td>
</tr>
<tr>
<td>Caps Lock</td>
<td>Q</td>
<td>W</td>
<td>E</td>
<td>R</td>
<td>T</td>
<td>Y</td>
<td>U</td>
<td>I</td>
<td>O</td>
<td>P</td>
<td>{</td>
<td>}</td>
<td>[</td>
</tr>
<tr>
<td>Shift</td>
<td>Z</td>
<td>X</td>
<td>C</td>
<td>V</td>
<td>B</td>
<td>N</td>
<td>M</td>
<td>&lt;</td>
<td>&gt;</td>
<td>?</td>
<td></td>
<td>Shift</td>
<td></td>
</tr>
<tr>
<td>Ctrl</td>
<td>~</td>
<td>!</td>
<td>@</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>^</td>
<td>&amp;</td>
<td>*</td>
<td>(</td>
<td>)</td>
<td>{</td>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ctrl</th>
<th>Win Key</th>
<th>Alt</th>
<th>Alt Gr</th>
<th>Win Key</th>
<th>Menu</th>
<th>Ctrl</th>
</tr>
</thead>
</table>

Removing Finger Memory Bias using Layout Transformation

It seems unfair to compare typing speeds for "computer" and "ismrfkdo" directly, because of the greater familiarity of "computer" than "ismrfkdo". As we can easily recognize the word "computer", and as we have a lot of experience in typing "computer" in our familiar layout, it is likely to be fast. For "ismrfkdo", however, there will be some recognition time to confirm its spelling and map it to finger movement. Even if the word "ismrfkdo" is the result of transformation from a wonderful keyboard layout, far faster than QWERTY, the result of this experiment may not demonstrate this advantage. For impartial comparison, we need to remove this familiarity advantage from the more familiar layout.

To achieve this, we define a layout transformation

\[ t : C \rightarrow C \]  

The aim of this transformation \( t \) is to re-map the keyboard layout in such a way as to remove the effects of word familiarity and finger memory from the current layout, while not affecting the physical speed of the layout.

Each layout differs in the frequency of use of each hand, finger, or row/column. Thus (Ferguson & Duncan, 1974) Dvorak (Dvorak, 1943) and Griffith (Griffith, 1949) designed their new layouts based on the observation that QWERTY over-uses the left hand, especially the index finger. These traits should not change during the test.

In order to minimize this change, we base our transformation on symmetries of the layout: most layouts have physical symmetries that do not affect (or more accurately, only very weakly affect) typing speed. For example, it is reasonable to assume that the ultimate (post-training) typing speed of the QWERTY layout would be only very weakly affected if the layout were horizontally reflected about the mid-line (of course, the accuracy of this assumption will need to be separately assessed). Why do we think so? Because the same fingers (but on the opposite hand) would be used to form the characters, and all required movements would be of the same magnitude, simply reversed in direction. That is, the symmetry (reflection about the mid-line) preserves some invariants (specific finger, movement distance) that we think determine typing speed, while varying other properties (hand, horizontal direction) that we assume have little effect (in specific cases – highly "handed" individuals, or people with specific injuries – these assumptions may not be valid, hence the corresponding conclusions will need to be suitably qualified).

More generally, we need to define a set of invariants that should be preserved, and then define a transformation – generally, using symmetries – that preserves these invariants (Olver, 1995). The specific symmetries and invariants involved may depend on the specific physical keyboard and layout. For example, the full QWERTY keyboard also has a vertical symmetry, but we should probably not make use of it, because the top row (numbers) is less familiar to most users than the "letters" part of the keyboard, so that typing speed will be affected if numbers are mapped to letters and vice versa. Vertical reflection about the mid-line of the letter part of the keyboard would be feasible, but will result in no change to the middle row ("asdf..."), and thus may not completely remove familiarity bias (a small number of words, such as "glad", would be completely unchanged, and a larger number may be only slightly changed). Other fairness considerations also intrude. For example, the Korean keyboard assigns consonants (C) to the left hand side and vowels (V)
to the right hand side. Since all Korean syllables have the structure CV, CVC or CVCC, this results in a particular cadence of left-right hand alternations. Exchanging these would result in a very unfamiliar typing feel.\(^1\)

Combining these transformations, we might decide to use a transformation using both horizontal and vertical reflection about the mid-line of the letters. So \(2A(”a”)\) maps to \(2J(”:\”)\) and \(3A(”z”)\) to \(1J(”p”)\). (This leaves some right hand side characters and the numbers unmapped; so long as our tests don’t use these characters, this won’t matter).

Now, to compare the two layouts using our chosen string ”computer”, we apply the transformation \(t\) to the requested keycodes. In our case, for testing QWERTY, we ask the user to type the string

\[
f^{-1}_{\text{qwerty}}(t(f_{\text{qwerty}}(”computer”)))
\]

that is, the string ”ixrzvn,m”, while for Dvorak, we request the string

\[
f^{-1}_{\text{qwerty}}(t(f_{\text{dvorak}}(”computer”)))
\]

that is, the string ”clrmjdkx”. More abstractly, in testing the ”old” layout, we use the string \(f^{-1}_{\text{old}}(t(f_{\text{old}}(s)))\) and in testing the ”new”, the string \(f^{-1}_{\text{old}}(t(f_{\text{new}}(s)))\) (in all cases, we ask the user to type using the old, familiar layout).

Before starting the actual experiment, it is important to check whether the layout transformation leads to biases. These should be checked, as far as possible, at two levels:

1. Are the intended invariants (finger frequencies, row frequencies etc.) in fact preserved in the target language or corpus? (This can be checked computationally.)

2. Does the transformation significantly affect typing speed? (This requires experimental verification.)

The former is simply a matter of sampling the test corpus in both original and transformed forms, collecting statistics of the invariants, and confirming whether they are substantially changed. It is easy to do, but relies on the assumption that we have correctly identified the invariants. What if our assumptions are wrong? For example, handedness, or the cramping effect of typing on the bottom row, may be greater than we have assumed. To identify this, we need to carry out experiments, measuring the extent to which our chosen invariants preserve typing speed (specifically, we may ask our test participants to type the same phrases before and after application of the transformation). Of course, in most cases with human experiments, we won’t get perfect invariance – there probably will be some effect from swapping hands or exchanging rows. When we find that typing speed is not perfectly preserved under some invariant transformation, we have three main options:

1. Where the bias is large relative to the performance differences between the layouts, we may need to discard the invariant (and the corresponding symmetries and transformations) and find new invariants that better preserve typing speed.

2. Where the bias is small relative to the performance differences, we may safely ignore it, on the basis that the performance differences could not be the result of the biases.

\(^1\)Actually, there is a further complication for Korean: syllables such as VC have no representation in Korean orthography, so that it would not be possible to present the transformed words to the experiment participants. Repairing these syllables to representable forms would add extra letters, thus destroying the fairness of the transformation.
Where the resulting bias is comparable in size to the performance differences, another option is to correct the (by now, known) bias in comparing the typing speeds mathematically.

**Test Set Generation**

The preceding explanation shows how we can transform a test corpus to remove sources of bias in comparatively evaluating the physical usability of two keyboard layouts. But where does this corpus, used to generate the test pairs, come from? What restrictions should we impose to ensure statistical stability?

The first requirement is clear: the corpus should reflect the intended use of the layout. For example, many users will use a computer keyboard primarily for typing text in their primary language. For these uses, a corpus of representative text will be most appropriate. On the other hand, a mobile phone layout might be primarily intended for typing short messages, so that an SMS database might form an appropriate corpus; a smart phone’s heaviest use might be for on-line posting, so that a database of on-line postings would be most suitable. Whatever the keyboard, these considerations apply whether using the methodology proposed here, or more traditional comparison methodologies. We thus do not address the choice of underlying corpus further.

However our methodology imposes some further requirements. Although the above procedure is unbiased, in the sense that it eliminates the effects of familiarity with the original layout, sampling can still generate chance biases. For example, suppose the transformation for one layout generates the string "uatspn", the other "wxkjlq". Both are equally meaningless, but the characters in the former are much more frequent in normal English than the latter. Thus practiced English typists are likely to type the former substantially more rapidly. These examples may add significant noise to our results, requiring large testing samples for statistical stability, even if they are equally distributed for the two layouts. To overcome this, we post-prune our transformed corpuses, removing any pairs where the letter-level familiarity of the two transformed strings is too different.

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:

**Definition 1.**

\[
\mathfrak{F}(s) = \frac{1}{n} \sum_{k=1}^{n} \frac{p_T(s_k)}{p_U(s_k)}
\]

\[
= \frac{|L|}{n} \sum_{k=1}^{n} p_T(s_k)
\]

$p_T$ is the frequency of each letter in the target alphabet

$p_U$ is the corresponding uniform distribution

$s_k$ is the $k$th character of string $s$

Thus $\mathfrak{F}$ is a measure of the excess frequency of the particular string over what would be expected from a uniform distribution.

The metric $\mathfrak{F}$ is unbiased with respect to the string length, as we show in Theorem 1.
Theorem 1. The expected value of $\mathcal{F}(s)$ is independent of $n$, the length of string $s$.

Proof. From Definition 1,

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

(9)

Taking expectations:

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

(10)

$$= \frac{|L|}{n} \sum_{k=1}^{n} E[p_T(s_k)]$$

(11)

$$= \frac{|L|}{n} \sum_{k=1}^{n} \frac{1}{|L|}$$

(12)

$$= \frac{|L|}{n} \frac{n}{|L|}$$

(13)

$$= 1$$

(14)

That is, expected value of $(s)$ is 1, regardless of the length of the string $s$. \qed

Applying the familiarity concept to each string pair enables us to check the letter-level fairness. If the two strings transformed from a word in the corpus are equally familiar, then they won’t bias the overall results; if they differ substantially in familiarity, then they may substantially affect them, requiring a larger number of samples to ensure fairness. Because $\mathcal{F}(s)$ is unbiased with respect to string length as proved in Theorem 1, we can directly compare the familiarity of pairs, using a single criterion that doesn’t depend on the length of the underlying word. We simply exclude string pairs with too large a difference in familiarity.

Formally, we define the bias $\mathcal{B}$ of a pair of strings $(s_1, s_2)$ as:

Definition 2.

$$\mathcal{B}(s_1, s_2) = |\mathcal{F}(s_1) - \mathcal{F}(s_2)|$$

(15)

From the definition, the value of $\mathcal{B}$ lies in the range $[0 \ldots |L|]$, with smaller values corresponding to fairer comparisons. The distribution of $\mathcal{B}$ may depend on the language and the specific text corpus, so these factors should be considered in determining how to choose appropriate pairs. The simplest approach is to simply set a threshold, and to generate pairs, discarding any whose bias exceeds the threshold, until sufficient have been generated. For more precise control, we can use stratified sampling, applying independent sampling for each non-overlapping stratum from the overall distribution. This method tends to improve the representativeness of the resulting sample by reducing sampling error (Neyman, 1934), but seems unnecessarily sophisticated for most applications.
Figure 4. Experimental Protocol Overall Flow
Overall Experimental Protocol

Putting all these together, we propose a protocol for comparing a new key layout for a keyboard with a more familiar "old" layout, as shown in Figure 4:

To illustrate, we briefly explain how to apply it in comparing the Dvorak layout with QWERTY. As shown at the top of Figure 4, we first prepare a set of words from an appropriate text corpus. These words need to be filtered by relevant criteria such as word length, and (depending on the application) the occurrence of special characters etc. We also need to define the letter stroke coding, and the two functions $f_{QWERTY}$ and $f_{Dvorak}$. We use the mapping shown in Figure 2. We must also define the transformation $t$. Here, we make use of the left-right hand symmetry, and the vertical symmetry of the letter area. Thus we interchange columns A-J, B-I etc., and the rows 1 and 3, as described earlier. We then convert each word into two test inputs. In parallel, we apply the following two nested procedures to each word:

$$f_{qwerty}^{-1}(t(f_{qwerty}(s)))$$ (16)

$$f_{qwerty}^{-1}(t(f_{dvorak}(s)))$$ (17)

This gives us a pair of strings. For each pair, we check the bias $B$, eliminating those above our bias threshold. Collecting these pairs, we generate the final test set that we will test on our experimental participants.

Test Example: Mobile Phone Keypad Layout

To evaluate this methodology, we illustrate it in more detail with an application to a personalized multigram-based (PM) mobile phone keypad layout. This layout was originally proposed in (Lee & McKay, 2010), and comparatively evaluated using a classical evaluation methodology (detailed observation of the learning progress using only a limited number of participants). The layout uses the traditional 12-key numeric keypad, and our comparison evaluates the new layout against the widely-used "ABC" multi-stroke layout, in which letters are allocated three to a key, starting from the key "2", except for the letters ".", "q" and "z", which are allocated to key "1". The new layout uses up to four strokes on 10 of the 12 keys of the keypad, permitting it to code the 26 alphabetic characters plus up to 14 frequently-used multigrams. The layout is shown in Figure 5.

Lee and McKay focussed on two-handed use, in which the left thumb is used for the left column, the right thumb for the right column, and either thumb as convenient for the center column (in general, this will be the opposite thumb to that used for preceding and following characters). They focussed on this use because it is the one used by faster users – those more concerned with speed – who would be more likely to switch to a faster layout if it were available. Two-handed use has been shown to be faster than one-handed use with computer keyboard (Hünting, Laubli, & Grandjean, 1980; Maxwell, 1953), and this can be applied to mobile phone keypads. We note that the PM system layout is intended to reflect the user’s personal use of the phone, as reflected in a personal archive; for this evaluation, we use one particular archive (derived from the first author’s SMS history), and evaluate both layouts against this archive.
A KEYBOARD LAYOUT COMPARISON METHODOLOGY

Figure 5. Personalized Multigram-based keypad layout proposed by Lee and McKay

Applying the Test Protocol

As illustrated in Figure 4, we need to assign keystroke codes and define the mappings $f_{ABC}$ and $f_{PM}$ assigning characters to the code, generating the string pairs, which we filter for bias before generating the ultimate set of test string pairs.

Assigning Keystroke Codes.

In both PM and ABC, we used ten of the twelve keys (0, \ldots, 9) to represent letters and multigrams. Since this is an ambiguous keypad, with insufficient keys to directly represent the alphabet, more than one letter must be assigned to each key, using the number of strokes to disambiguate. Thus the keystroke codes must designate not only the key used, but the number of strokes (in this case, 1, \ldots, 4, though the ABC layout does not use 4). To simplify explanation in the remainder of this discussion, we represent the key indirectly, through its row and column position. So overall, each keystroke code consists of a 3-tuple $(\text{row}, \text{column}, \text{stroke})$, with $1 \leq \text{row} \leq 4$, $1 \leq \text{column} \leq 3$, and $1 \leq \text{stroke} \leq 4$. Note that this coding defines the positions of the "#" and "*" keys, although these are excluded from both character codings. The key code corresponding to each position is illustrated in Figure 6.

Following the flow chart in Figure 4, our next step is converting 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 $f_{ABC}$.

Layout Transformation.

We now need to define a suitable layout transformation, to equalize between the ABC and PM layouts, so that both are using unfamiliar words. To ensure that we do this fairly, we first define some invariants that should be satisfied by our transformation. These invariants should be sufficient to ensure fairness. Deciding on these invariants is a subjective process,
though they will later be objectively validated. For the 12-key mobile phone keypad, we decided on the following invariants:

1. The number of strokes required for a letter should not change.
2. The frequency of consecutive use of the same key should remain unchanged.
3. The frequency of consecutive use of the same hand should remain unchanged.
4. The average distance of movement of the thumbs should not change.
5. The balance between the hands should remain the same.
6. The distance from each transformed key to the center should be equal to that of the original key.

To satisfy (as far as possible) all the invariants listed, we make use of symmetry. Except for the key "0", which is actually not used in the ABC keypad, the 9 keys fall into a $3 \times 3$ matrix (the "0" key is used in the PM layout; we address this issue later). Thus we can reflect each key about the center ("5") key. Keys "1" and "9" are mutually exchanged, for example. Thus after keys "2" and "8" have been exchanged, letters "t", "u" and "v" are located on key "2", while "a", "b" and "c" on key "8". Letters on key "5" remain the same. This is depicted in Figure 7. To satisfy the requirement to preserve the number of strokes, we simply leave these unchanged – "a" still requires one stroke, "b" two, and "c" three. Because the mapping completely exchanges keys, the frequency with which the same key is used for consecutive characters is unchanged; similarly, because the hands are completely reflected, the frequency of consecutive use of the same hand is unaffected. The distance the thumbs must move is also preserved, only the direction being changed. Similarly, since the hands are completely exchanged, the right hand overhead now applies to the left and vice versa, so any (im)balance remains unchanged. Lastly, the distance from the center is also preserved due to the symmetry about the center.
We now need to confirm whether the layout transformation results in changes in our primary criterion, typing speed. For this, at the theoretical level, we can simply compare letter frequency before and after the conversion, as shown in Table 2. In this, we used the well-known letter frequency table from (Beker & Piper, 1982). Unfortunately this table does not measure the frequency of period characters ("."). For this, we used a result from (Sigurd, Eeg-Olofsson, & Van Weijer, 2004), that average English word length is 3.6 letters and average sentence length is 18.9 words. Thus we estimate the average sentence length is 85.9 letters (3.6 × 18.9 + 17.9, including spaces), assuming that each sentence has one period (more or less).

Based on this data, we can calculate the frequency of each row, column, and number of strokes. Row frequency is distributed 32.5%, 36.3%, and 31.2% for upper, middle, and lower. After reflection, the resulting change (exchanging upper and lower rows) only changes the frequency by 1.3%. For columns, the letters are distributed 30.2%, 29.9%, and 39.9% (left, middle, right). After exchanging left and right columns, we see a 9.7% difference. The number of strokes is not affected by this conversion. In sum, our layout transformation results in only a relatively small change in the invariants.

We supplemented this with a simple experiment measuring whether reflection in this way affects physical typing speed. There were seven participants; they were presented with every possible bigram from the Roman/English alphabet (including the period(.), a total of $27 \times 27 = 729$), and asked to type them. We measured the time required to type each bigram. Ignoring the 9 transitions (between two letters both using key "5") that are unchanged by the reflection, this gives us 360 bigram pairs, between the original and reflected bigram. 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 truly sufficient to preserve physical typing speed, we would expect these values to be equal.

Figure 8 shows the elapsed time for each pair. If the reflection has no effect on typing speed, they should lie on the line $y = x$. Using regression on the data, we get $y = 0.9814x$ with a coefficient of determination of 0.6887. That is, our results are a good approximation to the assumption that points are distributed with $y = x$, so that reflection has little effect.
Table 2: Letter Frequency Comparison before and after Layout Conversion

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

on typing speed.

For a more detailed analysis, we split the points into five groups depending on their direction of movement: horizontal move (e.g., key '4' to '6'), vertical move (e.g., key '1' to '7'), northeast-southwest (NE-SW) diagonal move (e.g., key '7' to '3'), southeast-northwest (SE-NW) diagonal move (e.g., key '4' to '9'), and no move (e.g., key '5' to '5'). Table 3 shows the mean and standard deviation for each group and overall. Overall, reflection affects typing speed by less than 10 milliseconds, though there is a slightly larger effect – still only around 2% – for movements with a vertical component. We statistically tested the null hypothesis \( H_0 : \mu_{\text{before}} = \mu_{\text{after}} \), against the alternative hypothesis \( H_1 : \mu_{\text{before}} \neq \mu_{\text{after}} \), using Student’s test statistic \( t \):

\[
(t) = \frac{\bar{X}_{\text{before}} - \bar{X}_{\text{after}}}{s_p \sqrt{\frac{1}{n_{\text{before}}} + \frac{1}{n_{\text{after}}}}}
\]

where \( \bar{X} \) is the mean, \( s_p \) is the pooled standard deviation, and \( n \) is the sample size.

\(^2\)Because we use reflection on the central key, the other four directions overlie these.
Figure 8. Average Elapsed Time (milliseconds) for Each Letter Pair (before and after reflection)

\[
t = \frac{9.47 - 0}{\frac{124.18}{\sqrt{360}}} = 1.45
\]  

At a significance level of 99%, we find \( t_{0.005}(360) = 2.61 \), so although there is some possibility of a small difference, we don’t have definite evidence of it.

Adapting the Methodology

However we have glossed over an important issue in this discussion. Our basic protocol assumes that the key spaces of the layouts are identical. That is, that the mappings \( f \) are bijective. In fact, this may not be the case, as in this example. The PM layout uses two

<table>
<thead>
<tr>
<th>Direction</th>
<th>Direction 1 (milliseconds)</th>
<th>Direction 2 (milliseconds)</th>
<th>Gap (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>→ 949 ± 199</td>
<td>← 945 ± 198</td>
<td>4 ± 096</td>
</tr>
<tr>
<td>Vertical</td>
<td>↓ 865 ± 224 ↑</td>
<td>901 ± 199</td>
<td>−36 ± 134</td>
</tr>
<tr>
<td>Northeast Diagonal</td>
<td>↘ 909 ± 203 ↑</td>
<td>888 ± 199</td>
<td>21 ± 087</td>
</tr>
<tr>
<td>Southeast Diagonal</td>
<td>↘ 908 ± 211 ↘</td>
<td>860 ± 211</td>
<td>48 ± 145</td>
</tr>
<tr>
<td>No move</td>
<td>○ 1161 ± 192 ○</td>
<td>1149 ± 219</td>
<td>12 ± 143</td>
</tr>
<tr>
<td>Overall</td>
<td>939 ± 224</td>
<td>929 ± 221</td>
<td>10 ± 124</td>
</tr>
</tbody>
</table>
Table 4: Effect of Key "0" Mapping (in milliseconds)

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>T-test Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untransformed Strings ($f_{ABC}^{-1}$)</td>
<td>702 ± 124</td>
<td></td>
</tr>
<tr>
<td>Mapping &quot;0&quot; ($f_{ABC}^{-1} \circ t$)</td>
<td>697 ± 127</td>
<td>0.14</td>
</tr>
<tr>
<td>Reflection and Mapping &quot;0&quot; ($f_{ABC}^{-1} \circ t \circ m$)</td>
<td>666 ± 137</td>
<td>0.99</td>
</tr>
</tbody>
</table>

parts of the key space unused by the ABC layout: up to four strokes (rather than three), and the key "0". So we have to adapt the protocol slightly to fit these aspects. We address the latter first.

**Additional Keypad Mapping.** Because the "0" key is not used in the "ABC" layout, our transformation $t$ will need to map it to one of the keys that are used (so that $f_{ABC}^{-1}$ will be well-defined for it). As we argued in (Lee & McKay, 2010), for two-thumb use, the distance of movement of the thumbs is relatively unimportant. There, we used an evaluation function based solely on the number of keystrokes and consecutive uses of the same key and/or thumb. Therefore, the simplest choice would be simply to map key "0" to key "8", the nearest location. This won’t change 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" will be mapped to key "5" instead. We call this additional keypad mapping as $m : C \rightarrow C$. (Now, 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.)

We need to verify experimentally that this will not introduce a substantial bias. We used a similar experimental setup to the previous, this time with 12 participants. We requested them to type every pair consisting of a letter followed by key "0" (empty space). We measured the timings for these, for the corresponding strings with that letter moved to key "8" (or more precisely, replacing space " " by the letter "t"), and also for the combined effect of our reflection transformation and movement of key "0" to key "8". The results are summarized in Table 4. In both cases, the t-test statistic is much less than the 99% significance value (2.77), so we don’t have sufficient evidence to accept the hypothesis that the mappings change the typing speed. Figure 9 shows the timing pairs for untransformed strings versus mapping "0" alone, and reflection plus mapping "0". Again, points are located very near the line $y = x$.

**Stroke Mapping.** What should we do about letters (including multigrams) for which the PM layout uses four strokes? $f_{ABC}^{-1}$ is undefined for these, so we have to find an alternative. Unfortunately, any alternative actually using four strokes must therefore involve two separate letters – either on the same key or on different keys. In either case, this will add an extra delay – either moving to a different key, or triggering the end of a letter – thus substantially biasing toward the ABC layout. However we note that 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, reasoning that the differences in typing speed would be small, and that because they are rare, the overall effect on typing speed would be even smaller.
Figure 9. Average Elapsed Time for Each Letter Pair.
Untransformed vs Mapped "0" (left)
Untransformed vs Reflected and Mapped "0" (right)

Table 5: Familiarity $\mathcal{F}$ Statistics: Mean±Standard Deviation (Median)

<table>
<thead>
<tr>
<th>Text Type</th>
<th>ABC Keypad</th>
<th>PM Keypad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile messages</td>
<td>1.325 ± 0.396(1.035)</td>
<td>1.163 ± 0.542(0.726)</td>
</tr>
<tr>
<td>Magazine article</td>
<td>1.331 ± 0.320(1.289)</td>
<td>1.176 ± 0.438(1.182)</td>
</tr>
<tr>
<td>Random string</td>
<td>0.987 ± 0.361(0.956)</td>
<td>1.012 ± 0.394(0.975)</td>
</tr>
</tbody>
</table>

To measure the extent of this effect, we computed the frequency of 4-stroke characters in the PM test set (3.78%, 35 out of 925 characters for one group). From the data in Table 2, we computed the mean excess time for typing a 2-stroke character over the corresponding 1-stroke character (156.2 ms), and for a 3-stroke character over the corresponding 2-stroke character (171.7 ms). Thus it seems reasonable to estimate that the excess for a 4-stroke character will be around 160 ms. That is, the typing time over the 35 characters in the PM test set may be underestimated by about 160 ms. This is not far from the estimate in (Kinkead, 1975), that 95% of keystrokes are completed within 330 ms, and 83% within 125 ms. Overall, the PM times are underestimated by about $160 \times 35 = 5600$ ms for each group of 4 participants. Therefore, we include a correction of 1400 ms per person, or $1400 \div 25 = 56$ ms per word, in the PM results below.

**Familiarity Difference (Bias) Threshold**

The final stage of our protocol is to test for any differences in familiarity between the paired letter sequences generated by the protocol. To determine a suitable threshold for dropping word pairs, we investigated the familiarity $\mathcal{F}$ and bias $\mathcal{B}$ distribution for three kinds of texts: the mobile phone messages used here, a magazine article (Altman, 2009), and random strings, as shown in Table 5 and Figure 10. In the latter, we see that the bias

\footnote{For random strings, $\mathcal{F}$ is close to the theoretical value of 1; it is a little higher, around 1.3, for real text. Under the PM keypad layout, the mean falls to around 1.1}
A KEYBOARD LAYOUT COMPARISON METHODOLOGY

Figure 10. Bias distribution for Mobile messages (SMS), Magazine article (ART), and Random String (RDM). Vertical axis is for Bias $B$, and horizontal axis for cumulative percentile distribution.

$B$ is fairly uniformly distributed at first, but starts to rise more steeply when it reaches a value a little under 1.0; we chose 0.85 as a threshold, resulting in about 85% of pairs being accepted.

Experimental Comparison: Settings

Text Corpus and Derived Word Pairs. For the text corpus, we used an archive of mobile phone messages from the first author.\(^4\) 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, this results in some bias against PM. We also excluded words of length less than 3 or more than 10. Of the 554 words originally extracted from the archive, 178 remained after filtering. We then randomly selected a test set of 100 words (discarding and resampling any that exceeded our threshold of 0.85 for the bias $B$).

Ideally, we would test every participant on every one of the 100 pairs – i.e. typing 200 random-appearing sequences in total. However there is a trade-off. By the time 200 words are typed in a session, fatigue sets in and performance falls off. Splitting the words into separate sessions would be possible, but requiring participants to undertake multiple sessions induces much more reluctance in participants, thus conflicting with one of the goals of our protocol. We chose instead to form our participants into groups of four. For each group, the 100 word pairs were randomly split into samples of 25 (i.e. 50 words) in random order, and each participant was allocated one of the samples. In total, we tested 116 participants so that each pair was tested on 29 participants. To remove any possible

\(^4\)This archive was actually used to define the PM layout; for a detailed discussion of why we view personalized layouts as useful, and how the PM layout was derived from the archive, please refer to (Lee & McKay, 2010)
sampling bias, the randomization into samples was repeated for each group, so that no participant saw the same set of 25 words, or the same word order. For each pair and each participant, we also randomized whether the word derived from PM layout or ABC layout was presented first.

**Experiment Participants.**

We tested 116 participants, 60 male and 56 female. Their ages ranged from 18 to 60, distributed as in Figure 11. Their occupations and major field are shown in Tables 6 and 7. We did not directly control for previous experience with the ABC layout, since this would have required a further round of detailed measurement. Subjectively, some were highly

![Figure 11. Age Distribution of Experiment Participants](image)

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Detailed Occupation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undergraduate Students</td>
<td>27</td>
<td>Undergrad Student, Professional Master Student</td>
</tr>
<tr>
<td>Professors and Researchers</td>
<td>39</td>
<td>Professor, Ph. D Student, M.S Student, Research Scientist</td>
</tr>
<tr>
<td>Software Engineers</td>
<td>20</td>
<td>Web Search Engine, Internet Service, Technology Planning, System Software Engineer</td>
</tr>
<tr>
<td>Business and Management</td>
<td>6</td>
<td>Business Management, Secretary, Human Resource Management</td>
</tr>
<tr>
<td>Games and Entertainment</td>
<td>12</td>
<td>Game Service Planning, Development, Design, Marketing</td>
</tr>
<tr>
<td>Others</td>
<td>12</td>
<td>Insurance Agent, Fund Manager, Pharmacist, Nurse, Soldier, Teacher</td>
</tr>
<tr>
<td>Total</td>
<td>116</td>
<td></td>
</tr>
</tbody>
</table>
Table 7: Major field of Study of Experiment Participants

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Detailed Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Science and Medicine</td>
<td>11</td>
<td>Physics, Biology, Medical Science, Nursing Science, Pharmacy</td>
</tr>
<tr>
<td>Engineering</td>
<td>46</td>
<td>Computer Science and Engineering, Electrical Engineering, Mechanical Engineering, Chemical Engineering, Civil Engineering</td>
</tr>
<tr>
<td>Social Science and Business</td>
<td>40</td>
<td>Business and Administration, Management, Law, Economics, Journalism and Mass-communication, Psychology, Geography, International Relations</td>
</tr>
<tr>
<td>Humanities and Education</td>
<td>10</td>
<td>Literature, Linguistics, Information, Education</td>
</tr>
<tr>
<td>Total</td>
<td>116</td>
<td></td>
</tr>
</tbody>
</table>

skilled, while others had more limited experience, but all had some experience. Instead, we used the direct measurement of their typing speed (since all strings were presented to be typed using the ABC layout) as a proxy for their ABC layout experience.

This test was run only once for each participant. We did not need or allow any training session, because the methodology directly uses whatever is their current facility in the "old" (ABC) layout, and internally equilibrates any differences in familiarity. We instructed each participant 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, just describing them as "50 random strings."

Experimental Device and Software.

To perform the experiments, we needed a programmable phone (to run our software) with an ABC-layout physical keypad. This is not an easy combination to find, and in fact the device we used is quite old: a Samsung Electronics model SCH-M470. The physical size is $101.5 \times 53 \times 16.8 \text{ mm}$, and it uses the Qualcomm MSM7200 chipset running at 400MHz, with 64MB RAM and 256MB ROM. The operating system is Microsoft Windows Mobile Version 6.0 Professional, and 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 Figure 12.

When we start the software, it first requires us to type our name. We then choose a question bank (in the range A to D), which determines which partition of 25 of the 100 word pairs we are allocated – each user is informed by the experimenter, which bank to choose. When we are ready, we start the actual experiment by clicking the "START" button. The software presents on the screen a string to be typed. When we have finished typing the string, we press the "OK" button on the keypad (located above key "3"). At the bottom of the screen, the software displays the current progress (out of the 50 strings), together with the time taken for the previous string. The elapsed time is measured from the first stroke of the question to the moment the "OK" button is pressed – thus the participant

---

5The "M" button is not used in experiments; it denotes "Manual", and is used for software debugging.
may pause to rest for a short time between strings. After the 50 strings, the program brings up an alert window showing the mean times corresponding to the ABC and PM layouts.

Result and Analysis

**Overall Results.** We measured the elapsed time for typing each word during the test. To estimate the overall improvement from the ABC to PM layout, we averaged the elapsed time for strings generated by the ABC layout (henceforth, ABC strings) and PM layout (PM strings). We then applied a correction to compensate for the 4-stroke to 3-stroke conversion described earlier. As we see from Table 8, participants took 19.32% less (corrected) time to type PM strings than ABC. The biggest improvement was 34.86%; only 2 out of 116 participants recorded a lower performance with the PM layout (-1.43% and -10.52%). We

<table>
<thead>
<tr>
<th></th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Strings</td>
<td>8.16 ± 2.58</td>
</tr>
<tr>
<td>PM Strings</td>
<td>6.53 ± 2.02</td>
</tr>
<tr>
<td>PM Strings (Corrected)</td>
<td>6.58 ± 2.02</td>
</tr>
</tbody>
</table>
Table 9: Improved, Similar, and Worsen word pairs with PM Layout, by the average elapsed time of each participant.

<table>
<thead>
<tr>
<th></th>
<th>Improved</th>
<th>Similar</th>
<th>Worse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>16.34 ± 2.45</td>
<td>2.36 ± 1.56</td>
<td>6.30 ± 2.18</td>
</tr>
<tr>
<td>Percentage</td>
<td>65.34%</td>
<td>9.45%</td>
<td>25.21%</td>
</tr>
</tbody>
</table>

Table 10: Mean Elapsed Times by Age Groups (seconds)

<table>
<thead>
<tr>
<th>Age Group</th>
<th>Count</th>
<th>ABC</th>
<th>PM</th>
<th>PM (Corrected)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under 20</td>
<td>10</td>
<td>6.73</td>
<td>5.14</td>
<td>5.19</td>
<td>22.79%</td>
</tr>
<tr>
<td>21 to 25</td>
<td>58</td>
<td>7.64</td>
<td>6.16</td>
<td>6.21</td>
<td>18.69%</td>
</tr>
<tr>
<td>26 to 30</td>
<td>37</td>
<td>8.54</td>
<td>6.88</td>
<td>6.93</td>
<td>18.78%</td>
</tr>
<tr>
<td>Over 31</td>
<td>11</td>
<td>10.90</td>
<td>8.55</td>
<td>8.60</td>
<td>21.09%</td>
</tr>
</tbody>
</table>

tested the difference between the two layouts using a Wilcoxon Rank Sum Test. (Wilcoxon, 1945). We found a p-value of 0.00000025 for the hypothesis that the PM layout is not faster than the ABC layout – that is, we can accept that the PM layout is faster with more than 99.9% confidence.

For another perspective on this, we counted how many string gave substantially improved (PM is faster than ABC by more than 10%), similar or worse (PM is worse than ABC by more than 10%) performance with PM layout. From Table 9, we see that about two thirds of pairs show better performance in the PM layout, but one fourth are worse.

**Detailed Analysis by Statistical Groups.** Because the protocol we introduced here can be completed by participants with relatively little effort (generally within 5 to 10 minutes), it is feasible for us to conduct experiments with a much larger number of participants, from a much wider range of backgrounds, than is possible with traditional protocols (which typically need to use "captive" populations). This makes it more feasible for us to evaluate the effects of different characteristics of the population. We illustrate with some detailed analysis, based on age, gender, occupation, and major field of study.

In Table 10, we can see the average speed and overall improvement for each age group. There are two important observations here, which would be difficult to evaluate with other, more limited protocols:

1. It seems clear that overall speed decreases substantially with age.
2. Nevertheless, the speed improvement between the two layouts remains almost constant, at around 20%.

Table 11 shows the speed and improvement grouped by sex. Males tended to be slightly faster than females, but females showed a little more improvement. The difference,
Table 12: Mean Elapsed Times by Occupation Groups (seconds)

<table>
<thead>
<tr>
<th>Occupation Group</th>
<th>Count</th>
<th>ABC (seconds)</th>
<th>PM (seconds)</th>
<th>PM (Corrected)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undergraduate Student</td>
<td>27</td>
<td>7.74</td>
<td>6.21</td>
<td>6.26</td>
<td>19.08%</td>
</tr>
<tr>
<td>Researcher and Professor</td>
<td>39</td>
<td>7.14</td>
<td>5.83</td>
<td>5.88</td>
<td>17.63%</td>
</tr>
<tr>
<td>Software Engineer</td>
<td>20</td>
<td>9.22</td>
<td>7.23</td>
<td>7.28</td>
<td>20.95%</td>
</tr>
<tr>
<td>Business and Management</td>
<td>6</td>
<td>9.57</td>
<td>7.64</td>
<td>7.69</td>
<td>19.58%</td>
</tr>
<tr>
<td>Game and Entertainment</td>
<td>12</td>
<td>10.08</td>
<td>7.83</td>
<td>7.88</td>
<td>21.76%</td>
</tr>
<tr>
<td>Others</td>
<td>12</td>
<td>7.86</td>
<td>6.39</td>
<td>6.44</td>
<td>18.01%</td>
</tr>
</tbody>
</table>

Table 13: Mean Elapsed Times by Major Field of Study (seconds)

<table>
<thead>
<tr>
<th>Field of Study</th>
<th>Count</th>
<th>ABC (seconds)</th>
<th>PM (seconds)</th>
<th>PM (Corrected)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Science and Medicine</td>
<td>11</td>
<td>8.43</td>
<td>6.24</td>
<td>6.29</td>
<td>25.26%</td>
</tr>
<tr>
<td>Engineering</td>
<td>46</td>
<td>8.27</td>
<td>6.61</td>
<td>6.66</td>
<td>19.43%</td>
</tr>
<tr>
<td>Social Sciences and Business</td>
<td>40</td>
<td>7.44</td>
<td>6.13</td>
<td>6.18</td>
<td>16.90%</td>
</tr>
<tr>
<td>Humanity and Education</td>
<td>10</td>
<td>8.12</td>
<td>6.48</td>
<td>6.53</td>
<td>15.59%</td>
</tr>
<tr>
<td>Art and Design</td>
<td>9</td>
<td>10.46</td>
<td>8.27</td>
<td>8.32</td>
<td>20.39%</td>
</tr>
</tbody>
</table>

however, is negligible.

We also investigated the effect of occupation. From Table 12, students, researchers and professors tend to be a little faster than others. However they show slightly lower improvement from ABC to PM keypad. This may be partially a result of a slightly lower average age.

Table 13 summarizes the results by field of study. All groups except Art/Design (who were somewhat slower, even though they tend to be younger) showed similar speeds. This may reflect lower familiarity with electronic devices. On the other hand, the science/medicine group showed a substantially greater improvement between ABC and PM layouts than other groups.

**Effect of Familiarity.**

Before, we argued that our protocol would discount the effects of familiarity with the ABC layout. How well is this borne out in practice? To test this, we compared the level of improvement between ABC and PM layouts, for a each 10-person-range of participants divided by ranking in speed. This speed is calculated based on total average time, including both ABC and PM strings. If there were a systematic bias in our results due to familiarity, we would expect to see different levels of improvement.

Table 14 shows the results. As expected, we have a fairly stable improvement from ABC to PM layout in all groups: PM is faster than ABC by around 20%, independent of ABC speed, even though there is a very large difference in the raw speed itself. For novices, who are unfamiliar with either layout, we may expect to see better performance with the PM layout (assuming it really is faster, as the traditional analysis in (Lee & McKay, 2010) suggests), because there is no familiarity difference to overcome – almost any reasonable analysis should confirm the improvement. For skilled users, though, the familiarity difference means that most analyses are likely to see a reduced difference between
Table 14: Mean Elapsed Times by ABC Familiarity Level (seconds)

<table>
<thead>
<tr>
<th>Speed Rank</th>
<th>Count</th>
<th>ABC</th>
<th>PM</th>
<th>PM (Corrected)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 10</td>
<td>10</td>
<td>4.34</td>
<td>3.51</td>
<td>3.57</td>
<td>17.85%</td>
</tr>
<tr>
<td>11 - 20</td>
<td>10</td>
<td>5.47</td>
<td>4.42</td>
<td>4.47</td>
<td>18.26%</td>
</tr>
<tr>
<td>21 - 30</td>
<td>10</td>
<td>6.11</td>
<td>4.80</td>
<td>4.85</td>
<td>20.61%</td>
</tr>
<tr>
<td>31 - 40</td>
<td>10</td>
<td>6.51</td>
<td>5.44</td>
<td>5.50</td>
<td>15.49%</td>
</tr>
<tr>
<td>41 - 50</td>
<td>10</td>
<td>7.32</td>
<td>6.10</td>
<td>6.16</td>
<td>14.81%</td>
</tr>
<tr>
<td>51 - 60</td>
<td>10</td>
<td>8.08</td>
<td>6.26</td>
<td>6.32</td>
<td>21.72%</td>
</tr>
<tr>
<td>61 - 70</td>
<td>10</td>
<td>8.22</td>
<td>6.55</td>
<td>6.61</td>
<td>19.62%</td>
</tr>
<tr>
<td>71 - 80</td>
<td>10</td>
<td>8.77</td>
<td>6.97</td>
<td>7.03</td>
<td>19.82%</td>
</tr>
<tr>
<td>81 - 90</td>
<td>10</td>
<td>9.40</td>
<td>7.54</td>
<td>7.60</td>
<td>19.13%</td>
</tr>
<tr>
<td>91 - 100</td>
<td>10</td>
<td>10.39</td>
<td>8.50</td>
<td>8.54</td>
<td>17.75%</td>
</tr>
<tr>
<td>101 - 110</td>
<td>10</td>
<td>11.74</td>
<td>9.11</td>
<td>9.16</td>
<td>21.97%</td>
</tr>
<tr>
<td>111 - 116</td>
<td>6</td>
<td>13.97</td>
<td>10.85</td>
<td>10.91</td>
<td>21.95%</td>
</tr>
</tbody>
</table>

Table 15: Predicted Typing Speed Improvement for PM Layout under Different Methodologies

<table>
<thead>
<tr>
<th>No.</th>
<th>Text Filtering</th>
<th>Test Method</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>Theoretical Analysis</td>
<td>45%</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Human Experiment (Direct Training)</td>
<td>53%</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Theoretical Analysis</td>
<td>28%</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Human Experiment (Applying our methodology)</td>
<td>19%</td>
</tr>
</tbody>
</table>

The principal aim of the new methodology is to measure the differences in expected physically-limited typing performances between layouts, but to do so with lower per-participant experimental effort, enabling a much wider sample of participants to be used, and thus eliminating potential major sources of bias. So how well does it stack up against the classical methodologies?

We previously (Lee & McKay, 2010) compared the PM and ABC layouts using a theoretical analysis based on a proposed model of human performance and a classical experimental paradigm, allowing participants to build up familiarity with the new layout while we measured their performance, following their improvement sufficiently long to be able to extrapolate their long-term performance. (No. 1 and 2 in Table 15). We repeated the theoretical analysis with the text corpus used in this paper (after text filtering), and also applied our testing methodology with human participants (No. 3 and 4 in Table 15).

As the testing methodologies differ, we cannot expect identical results. Most important, all used the same basic text corpus of SMS messages. However the new methodology had to filter some words, because it could not readily evaluate strings containing characters that are directly represented in the PM layout but not in ABC, so these had to be discarded. Since we argued (Lee & McKay, 2010) that the ability to compactly represent the ABC and PM performances. Since we see little difference, it seems that the protocol has successfully eliminated the bias resulting from this familiarity difference.

Discussion
some such strings was an important contributor to PM’s greater speed. Thus omitting these strings will tend to reduce PM’s advantage. Fortunately, we have a direct measure of this: methods 1 and 3 (the model-based theoretical analyses) were identical except for the text filtering. Thus text filtering can be expected to reduce the PM performance advantage by around 17% (the exact reduction will depend on the accuracy of the model, but it should be roughly correct even if the model is fairly inaccurate).

Even if we assume that 17% of the difference between the two evaluations results from the different text samples, the overall difference between evaluations 2 and 4 was 34%. There remains a difference of 17% (classical experiments versus the new methodology) – too large to ignore. What can explain these differences? We see three likely causes.

First, we need to consider the differing cognitive loads in the two protocols. In the classical training approach, the text typed by the participants is real English words – hence easy to remember. In the new protocol, the strings look random. During the “Character Recognition” phase – the first phase in the process of typing (William E. Cooper, 1983) – the eye does not read with its maximum achievable speed (the required speed for comprehension). Instead, the eye reads the characters just fast enough to feed the copy to the hand as it is needed (Butsch, 1932). In the next phase, remembering the characters to be typed in short-term memory, we can buffer just 4 to 8 letters, preventing further look-ahead (Buzing, 2003). Overall, random-appearing strings may be much more difficult to remember because of unfamiliarity. Participants in our experiment confirmed that they had to check the next character repeatedly during typing. This inevitably reduced their overall performance, but to a greater extent for faster typing, thus reducing the overall difference in performance. This is a systematic issue with the new protocol – it is inherently conservative, and will underestimate any performance differences between new and old layouts. While it would be desirable to eliminate such biases, at least it is a bias in the right direction.

Second, in a traditional evaluation, the familiarity difference (between new and old layouts) is handled by use of learning curves. Generally, participants are assumed to be familiar with the standard layout, so if after a few sessions of learning, we can see performance with the new layout increasing to close to the performance with the standard, it is common to draw the conclusion that the new layout is better. However, better performance during initial training (several hours to several weeks) may not imply better ultimate performance. It is entirely possible that the new layout is just easier to learn, rather than more efficient. Conversely, it is also possible that a new layout which performs poorly in the initial stages may be better than the standard after more training. In other words, the extrapolation required by the classical approach is substantial, and may very well be simply wrongly estimated.

For example, Strong compared typing speeds for QWERTY and Dvorak (Strong, 1956). For fair comparison, he trained 10 typists with Dvorak until they attained their previous QWERTY speed. Then, he invited 10 more typists who are familiar with QWERTY layout. Each group was trained with each layout for further 100 hours, and Strong recorded the learning curves. Surprisingly, the Dvorak group did not show better performance than the QWERTY group with further training, even though Dvorak is reported to very easy to learn in a short time (Harnett, 1972) and participants in Strong’s experiment also learnt it quickly. Thus we cannot simply assume that initial stage data can be a good estimator for ultimate speed.
Table 16: Model-Based Fitness Values, Familiarity ($\tilde{f}$) and Elapsed Time (seconds)

<table>
<thead>
<tr>
<th></th>
<th>Fitness Value</th>
<th>Familiarity ($\tilde{f}$)</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Strings</td>
<td>2.26 ± 0.49</td>
<td>1.24 ± 0.38</td>
<td>8.14 ± 3.60</td>
</tr>
<tr>
<td>PM Strings</td>
<td>1.56 ± 0.42</td>
<td>1.25 ± 0.50</td>
<td>6.51 ± 3.28</td>
</tr>
</tbody>
</table>

Third, there is the issue of statistical stability. The previously-reported results were based on a sample of only ten, too small for statistical reliability (an almost universal problem with this kind of experiment). Moreover seven of the ten participants were male Computer Science graduate students, whose performance may be atypical. These forms of bias could readily contribute substantially to the differences between the two sets of experimental results; note that the new protocol is able to avoid both sources of bias.

Results for Individual Word Pairs.

Further evidence for the validity of the methodology comes from an unexpected quarter. In Table 9, we saw that approximately one fourth of words were actually slower to type with the PM layout than with ABC. How can we explain this? We investigated various possible explanations for the performance differences:

1. The differences may be directly due to the physical motions, hence should be predictable based on our physical model (fitness estimate).

2. The differences may be due to differences in familiarity ($\tilde{f}$), in which case they should be predictable from the differences in $\tilde{f}$. 

Figure 13. Measured Improvement Rate vs. Model-Based Estimates for each word
3. The differences may be due to other causes.

We can verify the first two hypotheses by computing the relevant differences (fitness and $\bar{\mathcal{F}}$) for each pair, and comparing them to the differences in elapsed time. The overall statistics are shown in Table 16. From Figure 13 and Figure 14, we can see that there is a reasonable correlation (0.53) between the measured and model-based estimates of improvement, suggesting that a substantial portion of the difference as measured is indeed due to the physical differences in typing each string; especially, we note that while one fourth of strings are slower in PM than ABC layout, in all cases this result is predicted by the physical model (i.e. in no case is one difference positive and the other negative), so that the one-fourth of strings slower in PM than ABC are completely explained by the physical model.

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

On the other hand, we cannot completely exclude other influences on the results (since the correlation with the physical model was not 1.0). Very likely, the physical model is incomplete, and much of the remaining variation is due to noise.
Conclusion

Advantages of the Proposed Method

In this paper, we proposed a general procedure for comparing a newly-proposed key layout with a previous familiar one. It offers three main advantages over early direct-test comparison methods.

First, it allows a fairer comparison of the two layouts at the expertise level of the familiar layout, independent of the level of training in the new layout. Until now, it has generally been the practice to train participants on the new layout in order to reduce the familiarity difference between the layouts. This imposes serious limits on the quality of the comparison. In an experimental setting, this imposes very substantial experimental costs, but for realistic experimental scenarios, it is generally impossible to train participants to a level comparable to their experience on the familiar layout. Thus the comparison requires extrapolation of the performance on the "new" layout – an extrapolation that will inevitably be inaccurate. With the new method, we "borrow" the expertise on the "old" layout, removing the requirement to extrapolate. Thus we can expect more accurate predictions.

Second, this method widens the range of participants. There is no restriction on their familiarity with the "old" layouts, so we may invite experts, skilled users, and novices. Even though they may differ in familiarity or learning ability, the comparison is still valid. What is more, because we can realistically collect data from a large number of participants, we can form statistically-useful subgroups, and compare performance by groups – such as skill levels on the "old" layout – as we did in Section . Thus we can give not only a more statistically accurate analysis, but also a more detailed analysis, of predicted performance levels.

Finally, the method reduces the experimental cost in time, and potentially in equipment. This results from the elimination of training sessions. The experiment in the previous section, for example, took on average 6 minutes, and at maximum 13 minutes, for each participant. By contrast, the experiments in (Lee & McKay, 2010) typically required 10 sessions, each of around 10 minutes. We also had to space them over a few days to eliminate fatigue effects. Thus for our new experiments, it has been relatively easy to acquire participants even from new acquaintances, since we just need to explain what we are doing and ask "Can you spare 10 minutes for the test?". We were easily able to acquire around 120 participants (30 data for each string pairs) necessary to ensure good statistical reliability and to allow us to sub-group the data. For the earlier experiments, we had to use "captive" participants (in this case, our lab members and close friends and family members), and to request of them a substantial investment of effort, just to get an absolute minimum of 10 participants. Moreover, for the new method, only one device is needed, at a total cost of about US$70; for the old experiments, the device itself was an important limitation; even if we could have found more participants with this method, we would almost certainly have needed a number of devices so that we could progress the experiments in parallel.

Assumptions and Limitations

An important assumption of this work is that we are comparing two layouts for exactly the same keyboard. This assumption is often not precisely true. Even in comparing QWERTY and Dvorak keyboard layouts, there are small differences in exactly which keys
are used for alphabetic characters (and comparisons involving special characters may lead to problems, since intermediate-level touch typists may know all the alphabetic locations instinctively, but have to search for the punctuation characters). With the ABC versus PM comparison, we had to adapt and compensate for the slight differences in key use (the "0" key and four-tap sequences). But it is also instructive that we were able to do so, and to quantify fairly accurately the effect of these adaptations. Nevertheless, such adaptation of the method is not straightforward, and requires careful analysis in each case.

In addition, the method assumes that we can identify a set of suitable invariants, and symmetries preserving those invariants (so that we can obfuscate the "familiar" layout). Since the invariants will rarely be exact, we need to quantify the failures, so that we can either compensate for them, or at least estimate their maximum effect on the results. Thus we assumed that motion scale was important, but direction unimportant; in fact, there was some indication that the direction of vertical motion does have some effect on speed, though not enough to change the conclusions.

The symmetries may also be inexact. Mobile phone keypad are typically completely symmetric (though the human users may not be). On the other hand, QWERTY keyboards typically are not, being generally staggered slightly to the upper left, so that horizontal reflections may change distances. Again, the extent of these effects need to be quantified, and if possible compensated.

Finally, we have tacitly assumed that language distributions on the keyboard are largely unaffected by these symmetries. Even in English, this is not really true – for example, most vowels are on the top alphabetic row in QWERTY, so that vertical reflection will move them to the bottom. Fortunately, however, 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 vowel or consonant sequences of more than three are rare. Thus effects from this would be rare, and we would generally ignore it. In some languages like Korean, however, the vowels and consonants are on opposite sides of the keyboard, and there are very strict rules about their sequence. Thus reflecting the keyboard would result in syllables that could not be written at all in Korean; and even if this were to be overcome (for example, by linearizing the alphabet rather than writing it in syllable blocks), the resulting cadence would have an unnatural feel, and so affect typing speed.

The work is limited in another way: it can only compare layouts that use the same physical structure. As touch screens become more common, and "soft" keyboards proliferate, we are starting to see proposals not merely for new arrangements of the letters on pre-existing keyboards, but also for completely new keypad structures such as touch screen-based text input (Nesbat, 2003), or for new physical ways of using old layouts (Kushler & Marsden, 2006). Our methodology may not be appropriate for these applications.

Our methodology also tends to underestimate long-term performance with the new layout, due to cognitive load. As discussed in Section, the target strings look random, so that users may be slowed by the necessity to memorize them. Since underestimates are generally only a minor problem (if we get a 30% improvement from using the PM layout when we only expected 20%, we will be pleasantly surprised), we have not worried about compensating for this here, though it is relatively easy to do. The cognitive differences between "random-seeming" strings and recognisable language should be the same for the old and new layouts. But we can readily estimate this difference for the old layout, since we
have performance figures for the transformed (random-seeming) strings, and can get them for typing the original strings from which they were derived. The same correction can then be applied to estimate the ultimate performance for the new layout.

Finally, 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 required. For already-skilled users, learning effort seems to be the dominant issue; new users, however, can afford to pay attention primarily to the final speed, since they will have to invest learning effort whatever layout they choose. In the past, physical cost of input devices has been an important barrier to adoption of new layouts (for example, the failure of the demonstrably-more-efficient Dvorak layout (Dvorak, 1943) to displace QWERTY (Buzing, 2003)). With the rise of ”soft” input methods, it appears that this barrier will be less important in future (Raynal & Vigouroux, 2005; Nesbat, 2003).

Epilogue

Although the proposed method has some limitations, we believe it improves both the ease and accuracy of keyboard layout speed comparison between a ”new” design and an ”old” one. Those two aspects are closely interconnected, because we can conduct the comparatively easier experiment many more times, increasing the available data for statistical analysis, while also eliminating important sources of bias.

While in some cases, we believe this methodology can replace the previous approaches, especially if ultimate typing speed is the primary measure of importance, more generally we view it as complementary to learning-based methods; in other words, a combined approach may be most useful, using the above approach to quickly gain preliminary assurance from a relatively small sample that the new layout is worth investigating, and then expanding the sample pool with this method to gain accurate understanding of the speed-up for different classes of users, while also using learning-based methods to estimate the overall learning cost. Comparison of the results is useful in another way as well; since the sources of bias are very different, the two methods can independently validate each other.

NOTES

Acknowledgements. We express our sincere appreciation to the volunteers in our human evaluations, and especially to Joonyoung Lee and Haram Kim for helping to find additional volunteers.

Support. This research was supported by a Korea Research Foundation grant funded by the Korean Government (Project No. KRF-2008-313-D00943) The Institute for Computer Technology of Seoul National University provided research facilities for the study.


References


