Reverse-Engineering Scanning Keyboards

Foad Hamidi and Melanie Baljko

Department of Computer Science and Engineering, York University
4700 Keele St., Toronto, Ontario, Canada, M3J 1P3
{fhamidi,mb}@cse.yorku.ca

Abstract. *Scanning* or *soft keyboards* are alternatives to physical computer keyboards that allow users with motor disabilities to compose text and control the computer using a small number of input actions. In this paper, we present the *reverse Huffman algorithm* (RHA), a novel Information Theoretic method that extracts a *representative latent probability distribution* from a given scanning keyboard design. By calculating the *Jensen-Shannon Divergence* (JSD) between the extracted probability distribution and the probability distribution that represents the body of text that will be composed by the scanning keyboard, the efficiency of the design can be predicted and designs can be compared with each other. Thus, using RHS provides a novel *a priori* context-aware method for reverse-engineering scanning keyboards.

Keywords: Scanning Keyboards, Information Theory, Huffman Algorithm.

1 Introduction and Background

For many people with disabilities, using conventional keyboards is a challenge due to the fine motor skills required. Using *scanning* or *soft keyboards* is a popular alternative indirect text entry technique that allows users to compose text using as little as one or two reliable input actions [8] (see figure 1, left). In this approach, an arrangement of alphanumeric symbols and system commands, referred to as *selectables*, is displayed on a screen and highlighted or put in focus in some order (hence, the use of the term scanning). The user selects a highlighted selectable by performing an *input action*, ranging from single button presses to the use of puff switches and EMG signals [3]. *Focus advancement*, the movement of focus over the hierarchy, can either be triggered by a separate input action (i.e., *active*) or automatic (i.e., *passive*). While active focus advancement requires more effort, it provides more control over the pace of the interaction and is preferred by some users.

The selectables are often grouped. In order to select a selectable, the user has to select its group by performing the input action when it is in focus. This action will select the group, after which only its subgroups, one of which contains the desired selectable, will be highlighted. The user has to keep selecting subgroups that contain the desired selectable until it is highlighted by itself, at which point an input action would add it to the created text. Scanning keyboards are often speeded up by various techniques, such as text completion or text prediction and have been effectively used by

people with disabilities in various contexts, especially as part of *voice output communication aids* (VOCAs) that synthesize created text into speech using a *text-to-speech* (TTS) module [1, 6] and are used by individuals with little or no functional speech, writing or gesture (the so-called *communication disorders*). The scanning keyboards examined here can also be used in other contexts, such as mobile computing, in which small keyboard size and number of input actions is desirable [8, 9].

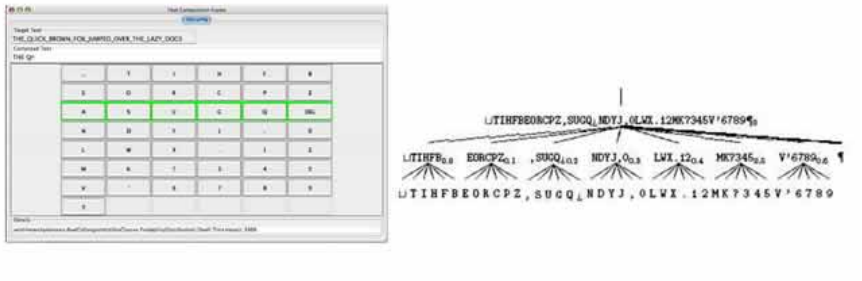

Fig. 1. A scanning keyboard with selectables highlighted in rows and columns (left) and the corresponding encoding tree (right)

Among the different factors affecting the performance of scanning keyboards, the layout of the keyboard (i.e., the ordering and placement of the selectable) is an important design aspect that affects the performance directly [3]. In this work, we focus on analyzing this aspect of scanning keyboard design. Many variations of scanning keyboard layout design exist [13, 14]. In *linear* design, selectables are not divided into subgroups and are presented to the user one after the other in some order. For example, in an alphabetical linear design, the user moves among selectables one after the other in alphabetical order. The shortcoming of this approach is that to make a selection, all the selectables that are placed before the desired selectable have to be traversed. In other words, a shortcut to the selectable does not exist. This method is often improved by placing the most frequent letters of the alphabet at the beginning of the list so that fewer focus advancements are required to reach them. Another popular approach, the *row-column* design, groups the selectables into categories as mentioned above. There are many possible variations on the row-column design that aim to improve the performance by having subgroups within groups or arranging selectables according to the occurrence probability of their corresponding symbols or commands. Scanning keyboards can either have a *static* or a *dynamic layout*. Static layouts, which we focus on in this work, remain the same during the interaction but dynamic layouts might change after each selection, usually rearranging themselves so that the next most likely selectable is easier to select [8]. There is a tradeoff between the speedup that dynamic layouts can provide and the disorientation users might experience due to selectable rearrangements.

We present an *a priori* method (i.e., a method that can precede human participant experiments) for the analysis and comparison of scanning keyboard designs. This method is context-aware, in the sense that information about context of use is incorporated in the evaluation, and can be used to make predictions about the efficiency of

a particular design prior to conducting user experiments, thus decreasing cost and effort considerably.

2 Models of Scanning Keyboards

Different descriptive and predictive models have been developed for the evaluation and design of indirect text entry methods for both assistive technology [1] and mobile computing [9]. The goals of work in this vein include the development of tools for the prediction of outcome (in terms of text entry rate and other efficiency measures) and for the principled design of novel and improved entry techniques. One popular model is the *keystroke-level model* (KLM) [2] that predicts expert error-free task completion times by breaking down a task into sub-tasks and summing up the sub-task times and the additional required overhead. When applied in a priori analysis, the metrics essentially boil down to *mean encoding length* (MEL): the mean number of input actions per character (or the mean amount of time required per character, depending on how "encoding" is characterized). MEL is sometimes normalized in terms of words per minute (wpm).

In prior work, we recognized the similarity of indirect text entry to a code transmission task [1]. In the popular variant of scanning keyboards that uses automatic focus advancement - the encoding alphabet ($\sum$) corresponds to actions that consist of a mix of *dwell periods* (i.e., time it takes for focus to traverse from one element to the next) and input actions. In this context, MEL is a more apt and generalizable evaluation metric than KSPC [7] since it does not assume an equal-cost encoding alphabet and takes into account the time required for focus advancement, if any. It followed from our observation that Information Theoretic source and channel coding techniques could be applied [1]. The behavior of the scanning keyboard can be expressed using an encoding tree and the focus advancement rules, if any. The *encoding tree* is a directed acyclic graph in which each leaf node corresponds to a single selectable in W and each internal node corresponds to a group of selectables that are highlighted during interaction with the system (see figure 1, right). The dynamic behavior of the system is captured by the notion of *focus*. At any given point in time a single node in the graph is in focus. The focus moves in the tree corresponding to the movement of highlighting on the interface of the scanning keyboard. In order to select a selectable, the user has to traverse the focus on the tree from the root to the leaf with the corresponding selectable.

For each selectable, the path from the root of to the corresponding leaf is unique and expresses a prefix-free encoding. Given an encoding tree, MEL can be calculated by weighing the length of each of these encodings by the empirically occurring probability of the symbol corresponding to the selectable in extant text.

3 Reverse-Engineering Scanning Keyboards

We present the *reverse-Huffman algorithm* (RHA), a novel method for *a priori* evaluation of indirect text entry designs. RHA is developed with the view that even extant

interfaces are formulable as solutions to the source coding problem and any indirect text entry technique can be seen as an optimal solution for some set of parameters. It aims to extract these parameters and compare them with ones relevant to a target user population.

We hypothesize that there is a hierarchy, often latent, in every scanning keyboard design. This hierarchy expresses the relative importance of the selectables in terms of their occurrence frequency and can also be characterized using a probability distribution. While the existence and ordering of this hierarchy is clear in linear layouts where the selectables are ordered by their frequency or alphabetic occurrence, it is less apparent in more complex layouts that use groups and subgroups. Even in designs where other considerations, such as the user's familiarity with popular layouts such as the alphabetical or QWERTY layouts, inform the layout design, there is still a latent hierarchy that expresses the relative ease of selection for each selectable. In the design of scanning keyboards, it is desirable to make the more frequent selectables easier to select (in terms of time and input actions needed), thus an important design goal is to require a smaller number of input actions to select a more frequent selectable. This is the principle behind using MEL and other related metrics [8]. Thus, this hierarchy would capture the design rational behind a specific layout and recovering it would allow us to compare different designs with each other.

Our method extracts this probability distribution and measures its proximity to extant empirically occurring probability distributions. The result of this comparison predicts the efficiency of the design. This method is a tool for designers to analyze and compare their various designs before conducting experiments with human participants that is a necessary complement to our method and helps paint an accurate picture of what the interaction would eventually look like.

3.1 The Reverse-Huffman Algorithm (RHA)

The Huffman encoding or algorithm is a well-known method for generating efficient encodings [5]. Given a set of *selectables* (W), a probability distribution over W and an *encoding alphabet* ($\sum$), the Huffman algorithm produces an encoding tree with the smallest MEL for each symbol. The *outdegree* of the tree (k), which is the maximum number of children internal nodes can have, is another factor affecting its structure.

We have developed a *reverse Huffman algorithm* (RHA) that views any given encoding tree as the output of the Huffman algorithm for some input probability distribution. Given W, $\sum$ and an encoding tree, RHA extracts a *representative latent probability distribution* from the design. RHA answers the question of what probability distribution best describes the text that a given scanning keyboard can most efficiently create.

We describe the steps of the algorithm: Given a scanning keyboard design variant, we create its corresponding encoding tree. As mentioned before, this tree expresses the ordering of focus advancement over the selectables during interaction. We then transform the given tree into a new encoding tree in which every internal node has the same number of children or outdegree by adding *ghost leafs,* simple place holders that cannot be selected during interaction, to them.

Next, we generate a set of linear constraints on the relationships between the probabilities associated with the selectables that describes the hierarchy of the tree. For a given tree, the constraints describe the infinite set of latent probability distributions that can create the tree if input to the Huffman algorithm.

The constraints are defined as follows. First, the sum of all probabilities must be 1. Second, the probability of each internal node is set to the sum of the probabilities of its immediate children. Next, the selectables are ordered by performing a breadth-first traversal of the tree that visits leaf-nodes in decreasing order of probability. Constraints are created expressing this relationship; the probability of each selectable is more than the probability of the selectable following it. These linear constraints identify the aforementioned set of latent probability distributions. Inputting any member of this set into the Huffman algorithm would result in the modified encoding tree.

Next, a single representative latent probability distribution is identified by using an empirically occurring probability distribution. The empirically occurring probability distribution is derived from extant text corpora and can capture a context in which the design will be used. For example, in our experiments we have used two empirically occurring probability distributions: PCw, derived from a corpus of chat logs and PFw, derived from a corpus of formal English. The differences between the two distributions reflect and express the difference between the texts (due to factors such as different topics of discussion, spelling conventions or degree of language formality among others) created in each context in probabilistic terms. Their incorporation makes our method *context-aware.* Previous research has stressed the importance of incorporating such information about the context in which text will be created [4].

Given an empirically occurring probability distribution, we choose from the set of latent probability distributions, the probability distribution with the minimum sum of absolute differences between its corresponding probability values and the values of the empirically occurring distribution as the representative latent probability distribution. This allows the design rational to be expressed as a single probability distribution that is comparable with other designs, once the same input parameters are applied to them. As will be shown next, the representative distribution is used to measure the efficiency of a design with respect to extant text.

3.2 The Jensen-Shannon Divergence (JSD) as a Measure of Efficiency

The *Jensen-Shannon Divergence* (JSD) is a well-known measure for calculating the distance between two probability distributions [7]. We hypothesized that the JSD value between a representative latent probability distribution and an empirically occurring probability distribution measures the efficiency of the given design for the creation of text similar to the empirically occurring text. To test this hypothesis, we conducted an experiment in which we performed RHA on 36 scanning keyboard design variants. The designs included of 15 variants described by Venkatagiri [11]. These included both row-column variants and linear variants. The row-column variants included the familiar QWERTY layout, as well as, designs informed by symbol occurrence probabilities. The linear designs also included the alphabetical layout, as well as, designs informed by symbol occurrence probabilities. The designs also in-

cluded 21 variants created automatically by the Huffman algorithm using varying input probability distributions and outdegree values. Together, these design variants formed a diverse set that included both manually and automatically designed layouts. We calculated MEL and JSD values using two, previously described, empirically occurring probability distributions: PCw and PFw.

Next, for each variant, we calculated the normalized ratio of JSD to MEL, calculated as the JSD value over the percentage of improvement of MEL over the worst-case MEL. To clarify the analysis, we grouped the design variants based on their outdegrees and compared two groups of variants: the first group consisted of variants with outdegree 4 and the other group consisted of designs with outdegrees of 6, 6.25 and 7. Figure 2 shows the results.

A correlation analysis of the results showed that the two metrics are positively correlated and a correlation coefficient test showed that the correlation is significant for both PCw and PFw. The results were the same for both design variant groups. The design variants with outdegrees of 4 showed a significant positive correlation for both PCw ($\beta = 7.21$, $t(3) = 22.3$, $p < .005$) and PFw ($\beta = 7.26$, $t(3) = 16.6$, $p < .005$). Also, the design variants with outdegrees of 6, 6.25 and 7 showed a significant positive correlation for both PCw ($\beta = 2.91$, $t(4) = 5.54$, $p < .005$) and PFw ($\beta = 3.13$, $t(4) = 4.15$, $p < .005$).

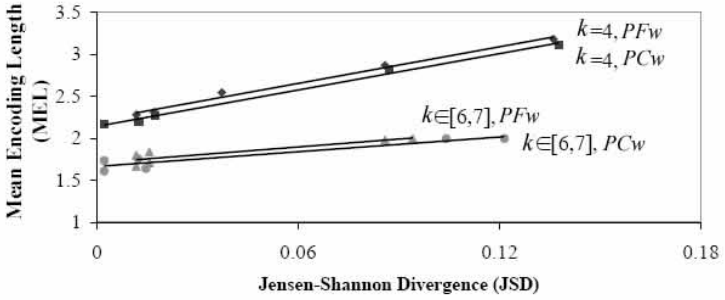

Fig. 2. Relationship between JSD and MEL for designs with outdegrees, *k*, of 4, 6, 6.25 or 7

Further ANOVA analysis revealed that the correlation between MEL and JSD is not affected by other independent variables. In these tests, the dependant variable was the normalized ratio of JSD to MEL and the independent variables were (1) derivation technique, with two levels indicating whether the design variant was generated using the Huffman algorithm or manually-derived; (2) probability distribution input to RHA, with two levels indicating whether or not the probability distribution used to generate the design variant and the probability distribution used as input to the RHA are the same; and (3) outdegree size, with two levels: large (values larger than 6) and small (values smaller or equal to 6). If JSD is reflective of MEL, we would expect that none of the independent variables will have a significant effect on our dependent variable. ANOVA analysis showed that the effect of all the independent variables is insignificant.

That JSD is positively correlated with MEL confirms our hypothesis that JSD can be an alternative to MEL. Our conjecture has been that JSD effectively measures the divergence of a layout's latent probability distribution and the one that occurs empirically and that this divergence is a fundamental factor in scanning keyboard evaluation. Thus, both MEL and JSD are essentially capturing the "optimality" of the design, but are doing so from different perspectives.

JSD provides several benefits. Firstly, it is easier to interpret than MEL; it has a lower bound of zero and tends to infinity. Moreover, it is valid to compare JSD values of designs that differ from each other in terms of factors such as encoding structure and keyboard layout. In order to analyze design choices, the researcher can build up a distribution of JSD values and develop an intuition about how to interpret particular JSD values prior to implementation. For instance, in our data analysis, the range of JSD values encountered was [0.002, 0.322]; since our data set included design variants that are known to be highly inefficient (e.g., linear QWERTY designs), we know that JSD values around the 0.3 threshold are indeed indicators of very poor designs. A corollary of this finding is that a design is doomed (in terms of its optimality), no matter how clever its underlying design principles, if its representative latent probability distribution diverges by around 0.3 from the empirically occurring probability distribution. A divergence of zero means that the a priori probability distribution is the same as the probability distribution that occurs empirically — that is, the design variant is perfectly matched to the application domain. In sum, a non-zero JSD value means that there will be a difference between how the scanning keyboard is designed and how it actually gets used in its domain. The larger the JSD, the more the actual application domain differs from how the design was envisioned to be used.

The interpretation of MEL, on the other hand, is more complex. Given a particular MEL value, how can one know whether it indicates a reasonably efficient design? It is difficult to know the lower and upper bounds in advance — they depend on the empirically observed probability distribution used to derive MEL. Two cases must be identified. In the first case the evaluation probability distribution is precisely the same as the one used to generate the layout and in the second case it differs. For the first case, the worst-case MEL, for any outdegree size, occurs when the codewords are equiprobable, where MEL will be equal to $\log|\sum|(|W|)$, where $|W|$ is the size of the set of selectables and $|\sum|$ is the size of the encoding alphabet. The best-case MEL is 1, but this occurs only for boundary cases — for encoding alphabets of size equal to the number of source symbols and for probability distributions so skewed that one source symbol is many times more likely to occur than any other (the precise factor depends on the size of the encoding alphabet). For the second case: the worst- and best-case MELs actually depend on the nature of the divergence between the two probability distributions. So, in sum, given a particular MEL value, it is difficult to ascertain whether it is a "good" or "bad" value — doing so requires an analysis that needs to take into account the size of the encoding alphabet, the shape of the empirically occurring probability distribution, and the divergence between the two probability distributions used for the design and the evaluation of the scanning keyboard design. The lower bound for MEL values differs according to each combination of these three parameters.

4 Conclusion

Our work is built upon the observation that a set of probability distributions that expresses the design rational in terms of the relative importance of the selectables is latent in a given scanning keyboard layout design. We presented the *reverse-Huffman algorithm* (RHA), an Information Theoretic reverse engineering method that extracts a representative latent probability distribution from a given scanning keyboard design variant.

We postulated that the *Jensen-Shannon Divergence* (JSD) between the extracted probability distribution and an empirically occurring probability distribution is a useful metric for the *a priori* evaluation and comparison of design variants. We applied RHA to 36 design variants, extracted their representative latent probability distributions and calculated JSD values using two different empirically occurring probability distributions. We showed that JSD is positively correlated with *mean encoding length* (MEL) and that it is more straightforward to interpret its values. Thus, we established the usefulness of JSD as a metric in this context and have shown that both JSD and MEL are alternative ways of looking at a given design from different perspectives.

5 References

1. Baljko, M. and Tam, A.: Indirect text entry using one or two keys. In: Proc. *Proceedings of ASSETS'06*, pp. 18--25 (2006)
2. Card, S. K. and Nordmann, R.: The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23, 396-410 (1980)
3. Felzer, T. and Nordmann, R.: How to operate a PC without using the hands. In: Proc. of ASSETS'05, pp. 198-199, (2005)
4. Grinter, R. E. and Eldridge, M. A.: y do tngrs luv 2 txt msg? In: Proc. of ECSCW'01, Wolfgang Prinz, Matthias Jarke, Yvonne Rogers, Kjeld Schmidt, and Volker Wulf (Eds.), pp. 219--238 (2001)
5. 5. Huffman, D. A.: A method for the construction of minimum-redundancy codes. In: Proc. of the Institute for Radio Engineers, 40, 9, pp. 1098--1102 (1952)
6. Kullback, S. and Leibler, R. A.: On information and sufficiency. Annals of Mathematical Statistics, 22, 79--86 (1951)
7. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory, 37, 1, 145--151 (1991)
8. MacKenzie, I. S.: The one-key challenge: Searching for a fast one-key text entry method. In: Proc. of ASSETS'09, pp. 91--98 (2009)
9. MacKenzie, I. S. and Soukoreff, R.W. Text entry for mobile computing: Models and methods, theory and practice. Human-Computer Interaction, 17, 147--198 (2002)
10. Majtey, A. P., Lamberti, P. W., Martin, M. T., and Plastino, A.: Wootters' distance revisited: a new distinguishability criterium. European Physics Journal D, 32, 413--419, (2005)
11. Venkatarigi, H.S.: Efficient keyboard Layouts for sequential access in augmentative and alternative communication. Augmentative and Alternative Communication, 15, 2, 126--134 (1998)