Pattern Matching on Stylus Keyboards: A Powerful Approach to Faster and Easier Pen-based Text Entry

Per-Ola Kristensson
Department of Computer and Information Science
Linköpings universitet
581 83, Linköping, Sweden

ABSTRACT

Much work has been done in developing alternative methods of writing for pen-based computers, including handwriting recognition, optimization of keyboard layouts, and specialized writing systems. Generally there is a trade-off between writing speed and the effort required by the user to write the text. My doctoral dissertation is about developing text entry systems maximizing the writing speed but minimizing users' effort by taking advantage of the redundancy of the human languages and viewing the legitimate input strings in the language as patterns mapped on a keyboard layout. I present my work on using pattern matching algorithms to take advantage of these constraints to develop a shorthand writing system combined with a stylus keyboard, allowing fast text entry without the need to learn any custom alphabet or a specialized writing system. I also present some of the feedback and output interfaces I believe can greatly enhance the user experience when using pattern matching text entry systems. I conclude by discussing performance evaluation and planned future work.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Input Devices and Strategies, Interaction Styles

Additional Keywords and Phrases: Shorthand, stylus keyboard, virtual keyboard, gesture-based interfaces

INTRODUCTION

Text entry including writing emails, research papers, patents and program code, constitute one of the most frequent tasks performed on a computer. Pen computers such as PDAs and Tablet PCs need a fast text entry method. Unfortunately both handwriting and speech recognition are limited by low speed and recognition problems [2, 5]. As a result the HCI community has researched alternative methods of efficient text entry. These can roughly be divided into three broad categories: the stylus keyboard, gesture-based text entry, and intelligent text entry systems.

The Stylus Keyboard

One approach is the stylus keyboard, also known as on-screen, graphical, virtual or soft keyboard. The obvious need is to miniaturize the QWERTY desktop keyboard. QWERTY was designed to prevent mechanical jamming in typewriters. As a result, the inventor C. L. Sholes deliberately distributed the frequent letter key combinations as far apart as possible on the left and right sides of the keyboard. When using a single point input device, such as a stylus, this results in frequent zigzag movements from the left to the right—clearly a suboptimal solution for pen entry. As a result, much effort has been invested in optimizing the stylus keyboard for stylus tapping. Getschow et al. [3] pioneered the approach of minimizing the statistical movement distance between the keys based on digraph frequencies. MacKenzie and Zhang [8] used such a model to heuristically derive a better stylus keyboard. Zhai et al. [12] introduced algorithmic methods of finding near optimal stylus keyboards, also incorporating other desirable properties such as maintaining a semi-alphabetic ordering among the keys and connecting frequently used words. Such a keyboard was dubbed ATOMIK (see Figure 2) and has a theoretically estimated text entry peak rate of 45 wpm.

Gesturing Text

A fundamentally different approach to computer text input is to let users write or gesture the text using the pen. Such an approach may be more natural to users since it utilizes the affordances of the pen better. The straight-forward approach is to let the users handwrite each individual letter, as in for example *Jot*, the single letter text entry method on Microsoft Pocket PC. However, the individual letters in natural alphabets, such as the Roman alphabet, are unnecessarily complex, and the slow text entry speed can be improved by simplifying the letters. For pen-computing an early approach is the well-known *Unistrokes* alphabet [4] that defines simplified gestures for the letters, giving the most frequent letters the simplest gestures. Another important design consideration in Unistrokes is the explicit delimitation of characters by designing every gesture to be entered as a single stroke, thereby avoiding the difficult segmentation problem in handwriting altogether. Since Goldberg and Richardson presented Unistrokes many alternative gesture based methods have been proposed. See [14] for a recent overview.

Intelligent Text Entry Systems

The previously described approaches all let the user input each character verbatim. However, character frequencies are not uniformly distributed, a fact Shannon used in demonstrating his seminal work on information theory [9]. Means of interactively exploiting language regularities in pen-based text entry have also been explored in the HCI community, both in gesturing and stylus keyboarding. A particularly intriguing example is *Dasher* [10], where the user slides a stylus (or other pointing device, such as an eye-tracker) towards different dynamic areas representing characters to be inputted, whose size dynamically changes depending on the probability of the character represented by the area appearing after the system's previous output. However Dasher demands constant visual attention from the user since the user has to visually scan a dynamically changing graphical scene.

SHORTHAND WRITING ON A STYLUS KEYBOARD

In an attempt to make exploitation of language regularities a natural component of text entry, Zhai and Kristensson [13] present a text entry technique that combines gesturing and stylus keyboarding, dubbed SHARK (Shorthand Aided Rapid Keyboarding). It is based on an observation in stylus keyboard research that for the most common words, users tend to remember the pattern of the word [11]. For the most frequent words ($\approx 100-200$ words) SHARK defines shorthand gestures, called *sokgraphs* [14] that are single stroke gestures traced by the letter keys comprising a word. Figure 1 shows the word *the* as a sokgraph (note that sokgraphs are recognized independent of scale and translation in SHARK). The rest of the vocabulary is typed using a stylus keyboard. According to the well known Zipf's law, about 50% of the word occurrences in English are composed of the 100 most frequent words, hence SHARK offers a potential speed-boost compared to ordinary stylus keyboards for about half of the text mass.

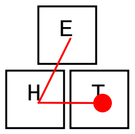

Figure 1: The word "the" as a sokgraph on the ATOMIK layout (shown in Figure 2). The circle indicates the start position.

Motivating Principles

Zhai and Kristensson [13] present five motivating principles for SHARK:

- *Scale and location independency*: SHARK recognizes sokgraphs independent of scale and translation, hence the writing need not be visually guided.
- *Efficiency*: SHARK is efficient compared to handwriting or specialized alphabets. In SHARK a single stroke denotes an entire word, compared to for example the Unistrokes alphabet, where one stroke equals one character. Also, SHARK is preferably defined on an optimized stylus keyboard such as ATOMIK [14], thus making SHARK movement efficient.
- *Duality*: SHARK uses dual methods of input, both optimized stylus keyboarding which can be used to enter any word, and sokgraphs for the most frequently used words in the language.
- *Zipf's law effect*: Zipf's law tells us that defining sokgraphs for only the most frequently used words would still yield a significant speed increase in about 50% of the writing.
- *Transition from tapping to gesturing*: The mapping between a sokgraph and its tapping pattern (serially tapping the letter keys of the word) is similar. Hence a user may decide to start writing the sokgraph instead of tapping the word, once the user has learned the pattern of the word.

Feasibility of the SHARK paradigm

Zhai and Kristensson [13] also conducted a feasibility study showing that users could learn about 15 sokgraphs per 45 minute training session. Hence, although artificial, SHARK seems to be easy to learn for users never exposed to the system before.

SHARK^2

Paradigm Shift

While we believe SHARK to be a system demonstrating great potential in pen-based text writing, we later realized that much improvement could be made in the paradigm. A critical weakness in SHARK is the alternation of tapping and gesturing, which not only complicates the text entry process, but also is an obstacle towards a gradual and smooth transition from novice to expert performance in going from serially tapping the letter keys, to gesturing the sokgraph. Both visually and kinematically, tapping and gesturing are distinctively different actions. Hence, the *duality* principle mentioned earlier, might not be as advantageous in practice as initially believed [13]. For this reason Kristensson and Zhai [6] revised the SHARK paradigm to allow *all* words in a large vocabulary ($\approx 10{,}000-20{,}000$ words) to be gestured directly.

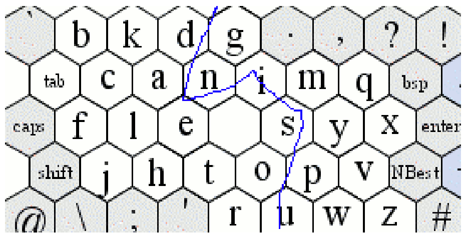

Figure 2: The user is writing "using" on the ATOMIK layout using the SHARK^2 system.

Multi-channel Recognition Architecture

To develop a system capable of efficiently searching among thousands of sokgraphs is a serious challenge for primarily three reasons. The first reason is that there is no natural source of sokgraphs to collect, thus we cannot use traditional well-understood data-driven methods to devise our classifier (see [1] for a recent review of such methods). The second reason is that we do not know which of the many features of the SHARK system (shape of the sokgraphs, the absolute location, number of corners, etc.) that are actually both efficient in separating sokgraphs from a computational stand-point, and are intuitive to the user. The third reason is that even if suppose we do know the most efficient features of the sokgraphs and have developed classifiers to recognize them, there is as of today no method of integrating multiple-classifier scores in a sound mathematical way that does not require training data. In essence we have a bootstrap problem: we need data to use the well-understood methods of pattern classification, but since our input system is novel, no such data exists, and cannot exist if we never build a working system in the first place.

To handle this challenge Kristensson and Zhai [6] developed a multi-channel recognition architecture where each channel does not necessarily have enough discriminative power on its own, but the combination (or integration) of the channels results in high-accuracy matches. The primary recognition channels are *shape* and *location*, where the shape channel examines the overall normalized shape similarity and calculates a score (due to space concerns we refer the reader to [6] for the details), and the location channel measures the absolute position of the user's pen trace on the stylus keyboard. In essence, preference is given to sokgraphs that are positioned closer to the user's pen trace (see [6] for the details). This can be viewed as a relaxation of the *Scale and location independence* principle in the original SHARK system, and was necessary to separate the large amount of possible sokgraphs now available.

Dynamic Channel Weighting. A "tweak" in the channel architecture is that the weighting of the location channel depends on the user's writing speed. For each word we calculate a Fitts' law estimation of the normative writing time. If the user is slower than this amount, we assume the user used visual attention when writing the word and hence we increase the weight of the location channel. Great care must be used when doing this kind of dynamic weighting because we do not want rare words (but which happens to be slightly more similar in shape to the input trace) to be given preference for a much more common (and thus more likely) word that is closer in location, just because the user produced a fast sokgraph. This is particularly true for a very short word, such as "as", that has many close neighbors on the ATOMIK layout, for example "cop", "coop" and "co".

Channel Integration

As previously stated there is no known mathematically sound method of integrating channel scores. However, a reasonable assumption is that the distance from the user's input to a template gesture follows a Gaussian distribution. Under this assumption we can integrate the channels using the Gaussian probability density function, where the mean is zero, and the standard deviation can be seen as a parameter weighting the contribution of the channel (see [6] for more details).

Language Models

Although we are satisfied with the performance of the SHARK^2 system some confusions and recognition mistakes cannot be avoided. Also, the user may write a sokgraph too "sloppy", making a "correct" classification difficult. Some degree of confusion also stems from the fact that short sokgraphs (two or three letters) can be partly or completely ambiguous when the lexicon is large enough. An example is the word "the" that has some nearest neighbors that are close in both shape and location such as "whig", "old" and "thee" on the ATOMIK layout. "thee" is in fact completely ambiguous to "the". To resolve such issues the current SHARK^2 system uses a language model that computes the bigram probabilities of the words in the *N*-best list given the previous word outputted from the system. The highest ranked word is then returned to the user.

FEEDBACK AND OUTPUT INTERFACES

All recognition based systems can occasionally make errors and SHARK^2 is not an exception in that regard. Some errors can be automatically corrected by a higher-order language model as discussed above. Other errors are more difficult to correct and require user intervention. Errors can be attributed to either the user or the recognizer, or both. The user can make an error by for example misspelling the word (tracing incorrect letter keys) or simply writing the sokgraph too imprecise, making accurate recognition impossible. The recognizer can make an error by having a too imprecise input in a too crowded classification space.

Morphing

To partly prevent the user errors we morph the user's pen trace towards the sokgraph that was recognized by the system. Since the key components of the system are based on the geometrical shape and location proximity, by looking at the animation the user gets a partial understanding of why the input trace was classified as that particular sokgraph.

N-best List Selection

To handle recognition errors, we mark all words outputted by the system that have other highly probable candidates with a distinct background color. When the user presses the stylus on such a word a linear *N*-best list pops up. The user can now drag the stylus toward any desired candidate in the list. The selection is finalized by lifting the stylus. We chose to use a linear list rather than a pie menu because it was found in an early feasibility study that pie menus suffer from the user's hand obscuring half the pie. A linear menu can pop up to the left or the right of the invocation point depending on whether the user is left or right handed.

EVALUATION

Extensive evaluation of a text entry system such as SHARK^2 is complex for many reasons (see [14] for an in-depth discussion on the matter). First, a task that is sufficiently natural but highly measurable has to be selected. Second, the issue of whether to allow users to correct their input or not is difficult: should we allow or force the users to correct their input and in that case how do we take these corrections into account when calculating the final performance scores? Third, the amount of training time taken by the users obviously affects the user's performance. We cannot know where a user's performance levels off until we actually reach that training point. Hence a time-consuming longitudinal study is needed to thoroughly evaluate the system. Although I do plan to conduct such a study, currently we only have expert estimates gathered from letting two users repeatedly write a particular set of words as fast as they can. These "expert speed" estimates ranges from 44-110 wpm for different sentences. The lowest score of 44 wpm was obtained by letting a user repeatedly (and completely correctly) write the sentence "The quick brown fox jumps over the lazy dog" using a 20,000 words large lexicon. It is worth mentioning in this context that the theoretical maximum speed obtainable by the optimized stylus keyboard ATOMIK is estimated to about 45 wpm. Hence, the system shows great potential as a faster alternative for text entry on pen-based computers.

CONCLUSIONS AND FUTURE WORK

We have shown that pattern matching on stylus keyboards is a powerful concept that can be used to devise efficient and easier to learn shorthand writing systems. In addition we have shown how to engineer such a system for a large vocabulary. We have also presented interaction and visualization techniques to ease the user experience of pattern matching text entry systems such as SHARK or SHARK^2. Finally we have presented results from user evaluations indicating that users can learn sokgraphs in reasonable time, and that the potential expert speed performance exceeds the one theoretically estimated for optimized stylus keyboards. Currently I am working on minimizing the memory and CPU usage of the system, to make it possible to run the system with a very large vocabulary (about 57,000 words) and limited computing power. I plan to release a full version of SHARK^2 to the public and collect some real usage data by letting users voluntary allow the software to report their average speed and mean number of corrections. Future work will be concentrated on doing extensive user evaluation of initial learning time, user acceptance and generally the practical performance obtained by users using the SHARK^2 system in their natural environment. Experiences from these evaluations will guide future development and refinements of the system.

ACKNOWLEDGMENTS

SHARK and SHARK^2 were co-invented with my Ph. D. advisor Shumin Zhai. Parts of the work mentioned in this paper were done when the author was a graduate intern at the IBM Almaden Research Center in 2003 and 2004.

REFERENCES

1. Duda, R. O., Hart, P. E. and Stork, D. G. Pattern Classification, 2nd ed., Wiley Interscience, 2000.
2. Frankish, C., Hull, R. and Morgan, P. Recognition accuracy and user acceptance of pen interfaces, *Proc. CHI 1995*, pp. 503-510.
3. Getschow, C. O., Rosen, M. J. and Goodenough-Trepagnier, C. A systematic approach to design a minimum distance alphabetical keyboard. *Proc. RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference*, 1986, pp. 396-398.
4. Goldberg, D. and Richardson, C. Touch-typing with a stylus, *Proc. INTERCHI '93*, pp. 80-87.
5. Karat, C-M., Halverson, C., Horn, D. and Karat, J. Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems, *Proc. CHI 1999*, pp. 568-574.
6. Kristensson, P-O. and Zhai, S. SHARK^2 – A Large Vocabulary Shorthand Writing System for Pen-based Computers, *Proc. UIST 2004, CHI Letters*, (6)2, (*to appear*).
7. Kurtenbach, G., Sellen, A. and Buxton, W. An empirical evaluation of some articulatory and cognitive aspects of "marking menus", *Human Computer Interaction*, 1993, (8)1, pp. 1-23.
8. MacKenzie, I. S. and Zhang, S. X. The design and evaluation of a high-performance soft keyboard, *Proc. CHI 1999*, pp. 25-31.
9. Shannon, C. E. A mathematical theory of communication, *The Bell System Technical Journal*, 1948, 27, pp. 379-343, 623-656.
10. Ward, D., Blackwell, A. and MacKay, D. Dasher - A data entry interface using continuous gesture and language models, *Proc. UIST 2000, CHI Letters*, 2(2), pp. 129-136.
11. Zhai, S., Sue, A. and Accot, J. Movement Model, Hits Distribution and Learning in Virtual Keyboarding, *Proc. CHI 2002, CHI Letters*, (4)1, pp. 17-24.
12. Zhai, S., Smith, B. A. and Hunter, M. Human Performance Optimization of Virtual Keyboards, *Human Computer Interaction*, 2002, 17(2&3), pp. 229-270.
13. Zhai, S. and Kristensson, P-O. Shorthand Writing on Stylus Keyboard, *Proc. CHI 2003, CHI Letters*, (5)1, pp. 97-104.
14. Zhai, S., Kristensson, P-O and Smith, B. In search of effective text input interfaces for off the desktop computing, *Interacting with Computers*, 2004, 16, (*to appear*).