Five Challenges for Intelligent Text Entry Methods

Per Ola Kristensson
With Notes by Anthony Jameson

■ *For text entry methods to be useful they have to deliver high entry rates and low error rates. At the same time they need to be easy to learn and provide effective means of correcting mistakes. Intelligent text entry methods combine AI techniques with human-computer interaction (HCI) theory to enable users to enter text as efficiently and effortlessly as possible. Here I sample a selection of such techniques from the research literature and set them into their historical context. I then highlight five challenges for text entry methods that aspire to make an impact in our society: localization, error correction, editor support, feedback, and context of use.*

A writing system is a culture-preserving device and a defining feature of civilization. The continual improvements and refinements of writing systems throughout history are a testament to their importance. In the last 10,000 years, the media of writing systems have changed from cave walls, rune stones, and sand to papyrus, wax tablets, parchment, paper, and—in our age—networked computers. Carvings, pieces of scribble, and elaborate texts have been generated from tools ranging from stones, iron rods, and fingers to styli, quills, pens, typewriters, and keyboards. In parallel the very heart of writing systems has also changed and adapted. The original logographic writing systems, such as the hieroglyphs, have throughout the course of Western civilization gradually been replaced by refined alphabetic systems. Between the 16th and 19th centuries the need for fast and accurate transcriptions of proceedings in courts and parliaments motivated research into various shorthand systems, culminating in the Pitman and Gregg shorthand writing systems that are still with us today.

This article is about the latest generation of writing systems—intelligent text entry methods. Such text entry methods use AI techniques to improve our writing systems. The high-performing speech and handwriting recognition systems we have available today would be impossible without decades of research and optimization. However, there is more to these systems than engineering. I explain why research on intelligent text entry methods is important in the next section. Then I sample selected text entry methods and set them into their historical context. I argue that for text entry methods to make a difference in society, we need to leave the narrow objective functions of high entry rates and low error rates. To that end I raise five challenges

for intelligent text entry: localization, error correction, editor support, feedback, and context of use.

Beyond the Desktop Keyboard

Our desktop keyboard has been with us with its familiar QWERTY layout since the typewriter was invented in the late 19th century. Contrary to popular belief, the main motivation behind the QWERTY layout was probably not to reduce typing speed (touch-typing was not invented at the time). Instead it solved an engineering problem: how to minimize the risk of mechanical jamming of the keys. Old typewriters worked much like pianos. When the user hit a key a hammer fell down onto an ink tape, which in turn caused an imprint on the paper. If two neighboring keys were hit in close succession there was a risk the two corresponding hammers would jam. The QWERTY layout solved this by distributing frequent letter-pair combinations to the left and right hand sides of the keyboard. This minimized the probability that two keys next to each other would be hit at the same time. As it turned out, such a bimodal distribution was reasonably efficient for touch-typing with the left and right hands, which is also why it provides us with a rather high entry rate and reasonably low error rate. This, in combination with society's tendency to keep technologies that are "good enough," is probably the primary reason why the desktop keyboard is still the de facto text entry method today (David 1985, Gould 1992).

However, some users suffer from repetitive strain injury (RSI) and prefer to avoid using a keyboard and mouse. Other users have a severe disability, such as paralysis, that prevents them from entering text except by using their eyes or perhaps by activating a few muscles. These users need alternative text entry methods.

Modern mobile phones easily support web browsing, e-mail, blogging, and so on. These applications all require a text entry method. Since the desktop keyboard is too large for a phone it is necessary to conceive practical alternative text entry methods that offer as high text entry rates as possible. If we also consider that users might want to enter text while walking outside on a windy and noisy street, the problem becomes even more complex. Is it possible to engineer efficient text entry methods that work well even under such difficult circumstances?

Leaving the mobile world aside for a moment, emerging mainstream technologies, such as wall-sized displays, multitouch tables, and volumetric displays, do not lend themselves to the use of desktop keyboards. For all these reasons, it is worth exploring text entry beyond the desktop keyboard. And let's not forget, it takes many hours of practice to become a skilled typist. Perhaps a more intuitive and efficient text entry method lies ahead of us and will replace the desktop keyboard altogether. If we don't search, we will never know if we are currently at a local or global optimum.

Exploring AI in Text Entry

AI opens up unique possibilities to create new text entry methods or improve upon existing ones. The main reason is that human languages are highly redundant (Shannon 1948). These redundancies can be captured in language models. Intelligent text entry methods take advantage of language redundancies in multiple ways. For example, they can reason about users' input and deduce users' intended words; or they can display predictions in a way that allow users' input to be used more efficiently.

Curiously, language redundancy has been exploited since at least the creation of Nova Ars Notaria ("the new note art") in 12th century Europe. This was a shorthand system attributed to the monk John of Tilbury. In it, letters were simplified into simple line marks. To enable faster writing of common word stems, they were compressed into sequences of simple line marks and dots. These word stems were found by frequency analysis of the Book of Psalms. (Melin 1927).

Despite being eight centuries old Nova Ars Notaria neatly illustrates the basic principles behind efficient text entry. First, users' efforts in articulating their intended words should be minimized. Second, language redundancies should be modeled and exploited (Kristensson 2007, Melin 1927). In this vein, there is a long history of AI contributions to text entry design, particularly in the early work on handwriting and speech recognition.

However, as this special issue illustrates, making efficient use of AI techniques in user interfaces is often easier said than done. Jameson alludes in the introduction of this special issue to the need for a binocular view that takes both AI and HCI perspectives into account. This article takes this perspective by presenting five challenges for intelligent text entry: localization, error correction, editor support, feedback, and context of use. To set the stage, we start by sampling the state of the art in intelligent text entry.

Intelligent Text Entry Methods

The usefulness of text entry methods is directly linked to entry and error rates. Unfortunately, the literature is inconsistent in how to measure text entry performance, and there are not even standardized measures of entry and error rate. In this article entry rate is measured in words per minute, where a word is defined as five consecutive charac-

ters. Unless otherwise specified, error rate is measured as the character-level edit distance between the stimulus text and users' response text, divided by the number of characters in the stimulus text. This provides an error rate between zero and one.

Text entry research has resulted in literally hundreds of systems, and as a consequence it is not feasible to survey them all. What follows is a sampling of recent intelligent text entry methods. Also note that entry and error rates cannot be readily compared across different studies because experimental setups are rarely consistent across studies, and participants' general performance may vary dramatically across samples. The accepted procedure to definitely conclude one method is better than another is to compare the methods directly against each other in the same experiment. Nevertheless, it is informative to get a sense of the text entry performance of various techniques, even if those performance measures are not directly comparable. So with that caveat, the sampling below lists entry rate measurements for all techniques, if such data is available.

Speech and Handwriting Recognition

Perhaps the most natural form of input besides a keyboard is to enable users to write text using speech or handwriting recognition.

Speech recognition has the potential to be very fast. A user can speak up to 200 words per minute (Rosenbaum 1991). In practice, individual performance, fatigue, recognizer design, accents, and noise characteristics of the environment and the microphone dramatically affect recognition performance, which in turn affects users' entry and error rates. Encouragingly, our recent findings indicated that novice users could speak at a mean entry rate of 13 words per minute while walking around outdoors in a windy environment. Ignoring recognition delay, an expert user had a mean entry rate of 45 words per minute while walking around outdoors. The corrected error rate (on the word level) was less than 2 percent. This indicates that speech recognition has the potential to be an efficient mobile text entry method (Vertanen and Kristensson 2009).

Another possibility is to use handwriting recognition. Handwriting recognition has evolved tremendously (Tappert, Suen, and Wakahara 1990; Plamondon and Srihari 2000). Our recent longitudinal user study showed that state-of-the-art handwriting recognition results in a text entry rate of about 25 words per minute after several hours of practice (Kristensson and Denby 2009). In addition, both entry and error rates were almost identical to what was provided by a de facto QWERTY on-screen keyboard. This means that a modern handwriting recognizer can be regarded as a reasonably fast text entry method for pen-based interfaces.

Intelligent On-Screen Keyboards

One of the most common text entry methods on today's mobile touch-screen devices is an on-screen keyboard. This is an image of a keyboard layout rendered on a touch-sensitive screen. The typical on-screen keyboard uses the familiar QWERTY keyboard layout, and this is the only keyboard layout that will be considered in this article.[1] On-screen keyboards are easy to learn and use as users benefit from transfer learning from their experience with their desktop keyboard counterparts. Since they lack tactile sensation feedback, users do not have a sense of the difference of being between two keys or at the center of one. This causes users to easily mistakenly hit an unintended neighboring key. In addition, unlike desktop keyboards, users are not provided any tactile feedback when a key press is registered by the system. This may cause users to mistakenly believe a key press was registered when it in fact was not.

On-screen keyboards are not very fast when used with a stylus or a single finger, such as the index finger. The mean entry rate varies depending on the study but is typically in the range of 15–30 words per minute (MacKenzie and Soukoreff 2002). This is in part because of the bimodal distribution of frequent letter key pairs, as mentioned earlier in this article.

However, both entry and error rates offered by on-screen keyboards can be improved by making two observations. First, the landing points on an on-screen keyboard pixel grid are of a much higher resolution than registered key presses on a physical keyboard. Second, human languages are highly redundant, and the regularities can be captured and exploited by a language model (Kristensson and Zhai 2005).

These two observations make it possible to infer a user's intended letter key, even though the user mistakenly hit a neighboring key. As we discussed previously above, such errors are easy to make on an on-screen keyboard. Further, Fitts's law (Fitts 1954) tells us that the time it takes to hit an individual key is proportional to both the size of a key and the distance to it. By relaxing the constraint that a user must hit an intended key exactly within its boundary we in effect temporarily artificially magnify the user's intended key. According to Fitts's law this increases the mean entry rate if users exploit the extra tolerance by pointing at keys faster (and therefore less precisely) than on a regular on-screen keyboard.

Goodman et al. (2002) present an on-screen keyboard that achieves this effect by combining a probabilistic model of users' landing points with a character-level language model. Their system uses this model to infer users' intended letter keys, as they are writing the text letter by letter.

We have previously explored an alternative

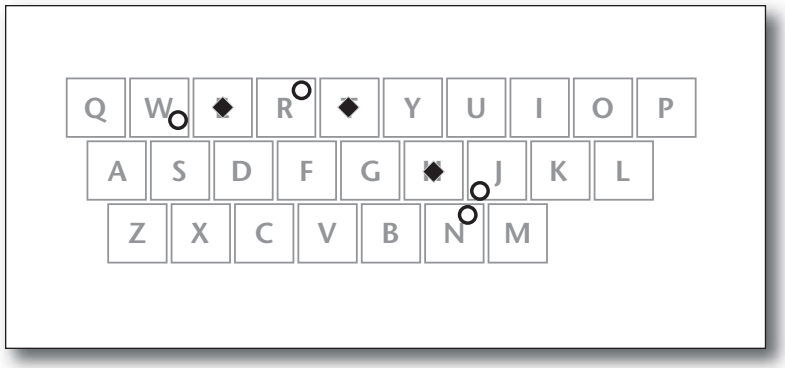

Figure 1. An Example of Automatic Correction Using the Elastic Stylus Keyboard.

approach that pattern-matches users' tapping traces against ideal point templates that represent entire words. We call it the elastic stylus keyboard (ESK). This system performs corrections by geometric pattern matching. A correction occurs on the word level and is triggered when the user slides the pen (or finger) from the left to the right. In figure 1, the user intended to tap the letter keys T, H, and E in succession (indicated by filled rhombi centered on the keys). However, the user accidentally missed all the intended keys and instead hit the neighboring keys R, J, N, and E in succession (landing points indicated by outlined circles). Despite the user missing all the intended keys, and despite a spurious extra tap when trying to hit the H key, the system would correctly infer that the user intended to hit the word *the*. In our experiments we used a lexicon with about 57,000 words. In a small formative expert study (involving the authors only) we found we could reach an entry rate of 45 words per minute using this system (Kristensson and Zhai 2005).

ShapeWriter (Zhai and Kristensson 2003, Kristensson and Zhai 2004) uses a radically different approach (figure 2). Instead of serially tapping the keys, ShapeWriter lets users gesture the shapes of words over an on-screen keyboard. Each word in a large lexicon has a corresponding template gesture that is defined as the connected lines that sequentially intersect the center points of the letter keys of the word. For example, the word *the* is a v-like shape on the QWERTY keyboard layout (see figure 3). The user's intended word is found by a pattern recognizer that primarily looks at the scale-translation invariant shape similarity between the user's gesture and the set of template gestures. In our user studies we found that ShapeWriter enabled novice users to write at a mean entry rate of 25 words per minute. In an accelerated learning study, where users repeatedly wrote multiple phrases, the same novice users reached a mean entry rate of 44 words per minute. The peak entry rate was 99 words per minute (Kristensson 2007).

Other Methods

A plethora of intelligent text entry methods have been proposed beside speech, handwriting, and on-screen keyboards. Darragh, Witten, and James (1990) present a system for typing prediction for desktop keyboards, and Masui (1998) uses such a system for a mobile text entry system.

Intelligent text entry methods are about inference, and inference is closely related to compression. Taking this thought one step further, one realizes that instead of predicting users' text one can try to decompress users' abbreviations. Abbreviations have a long history. Monks in medieval European monasteries often abbreviated words when copying bibles. For example, the Latin word *uerbum* ("word") was often abbreviated into ūbū (Janson 2004). Shieber and Nelken (2007) propose an intelligent text entry method that enables users to enter abbreviated words, thereby reducing the number of letter keys that have to be pressed by the user to write words. Their system automatically decompresses the abbreviations to full words as the user is typing. Unfortunately their experiments did not indicate that their system provided any higher entry rates in comparison to typing out the full words.

Clawson et al. (2008) correct physical thumb keyboard typing errors using a machine-learning algorithm. Physical thumb keyboards have previously been shown to be very fast, up to 60 words per minute. However, at such a high entry rate the error rate exceeds 8 percent (Clarkson et al. 2005). Therefore, an effective automatic error correction mechanism would have the potential to improve efficacy.

AI techniques have also been explored to improve text entry in eye-tracking interfaces. Salvucci (1999) presents a system that recognizes words in a small vocabulary by decoding eye-tracking traces over an on-screen keyboard. This system enabled study participants to write at a mean entry rate of 15 words per minute (derived from their reported results of 822 milliseconds per character).

A completely different approach to writing using eye-trackers is Dasher (Ward and MacKay 2002). In Dasher the user navigates a cursor through graphical boxes that contain the desired characters. The boxes' sizes and positions are based on a language model. In a sense, with Dasher the user is driving to the intended destination—the desired sequence of words. The driving is efficient for common word sequences because the system provides the most likely letter sequences with the largest areas on the display. For less common words, the driving will be more involved and the user has to steer carefully toward smaller areas (less likely letters). This system enabled one of the expert users to reach an entry rate up to 25 words per minute.

Challenges

As is illustrated in the above sampling, intelligent text entry methods span many areas, some of which have transformed into their own specialized research fields, such as handwriting and speech recognition. The fact that some of the above intelligent text entry methods are commercially available for desktop computers and mobile phones shows that decades of research within the intersection of AI and HCI have to some extent succeeded.

Nevertheless, there are still many issues that need to be solved. Below are five major challenges that I believe we need to tackle: localization, error correction, editor support, feedback, and context of use.

Challenge 1: Localization

Depending on your country, the topology of your desktop keyboard layout will vary. For example, the first six top-left letter keys read QWERTZ on a German keyboard and AZERTY on a French keyboard. However, since desktop keyboards are typed one character at a time, a different keyboard layout simply means that the operating system needs to switch key map.

In contrast, any text entry method that relies on a list of words (a lexicon) will need to collect such a list for every language supported. Collecting and verifying high-quality lexicons is a labor-intensive task that may require access to native speakers of each language.

Text entry methods that require higher-order language models, for example unigram, bigram, or trigram models, also require sufficient text data (corpora) to train those models. For language models to be useful the corpora need to be representative of what users are intending to write. It is currently difficult to get access to high-quality corpora for different languages that properly model what users intend to write on their mobile phones.

Some intelligent text entry methods, such as handwriting and speech recognition, also require models of users' articulations. For example, a speech recognizer needs at the very least an acoustic model to function. This model needs to be rebuilt for different languages.

In the literature, the issue of localization tends to be underestimated, particularly when completely new text entry methods are proposed. Nevertheless, a good example of a novel text entry method proposed by researchers that quickly supported many different languages is Dasher (Ward and MacKay 2002). Currently Dasher supports more than 60 languages. Since Dasher models each language as an adaptive character-level language, it can be seeded with a relatively small corpus and still function acceptably while it gradually adapts to a user's writing.

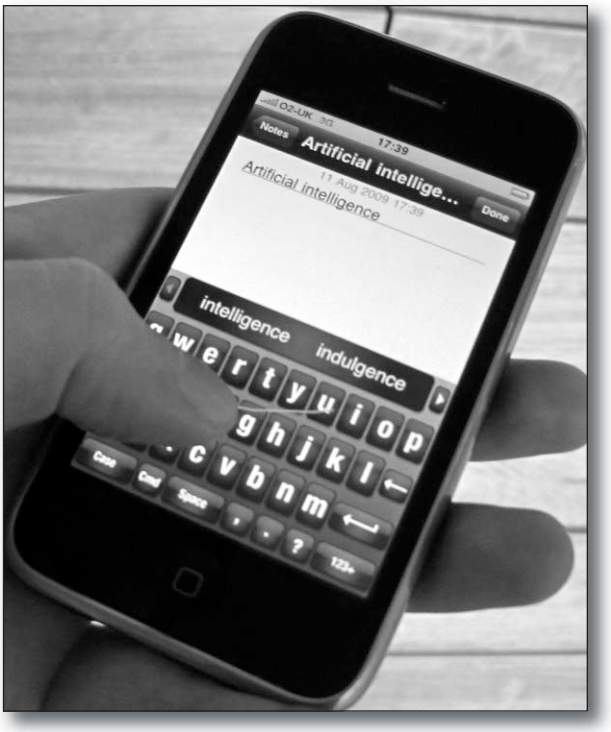

Figure 2. ShapeWriter on the iPhone.

The user is in the process of writing the word *using*.

Challenge 2: Error Correction

With or without a proper language model, in text entry, errors are unavoidable. Here we divide users' errors into two broad categories: cognitive errors and motor errors. Cognitive errors are due to the user having an improper model of an intended word. For example, a user may not be able to spell or pronounce an intended word correctly. Motor errors are due to mistakes that occur due to tremor or stress. Both classes of errors, but particularly motor errors, tend to increase as the task becomes more complex. For example, a user who is simultaneously walking and interacting with a touch-screen will find it more difficult to precisely hit targets on the screen (Crossan et al. 2005).

Another class of errors is not due to the user but to the system.[2] If the recognizer has an incorrect model of the user, then some errors are inevitable. In practice, it is difficult to design a perfect model. For example, the standard tri-gram language model of a speech-recognition system makes many assumptions of users' speech that are clearly not true, such as the assumption that all users' utterances are only dependent on the two last spoken words (that is, a trigram model). Another obvious model mismatch exists in word recognizers, such as the one used in ShapeWriter. Here the system uses a dictionary of words. If users' intended words are not in the dictionary the system cannot possi-

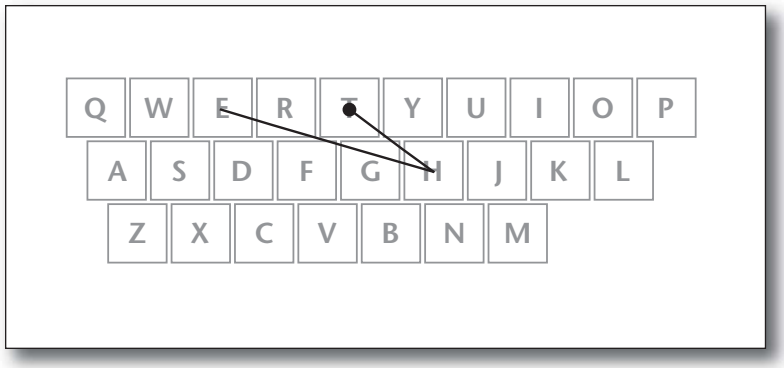

Figure 3. The Ideal Shape of the Word the in ShapeWriter.

Figure 4. Example of a Word Confusion Network Correction Interface for a Mobile Speech-Recognition System.

bly infer the correct words (so-called out-of-vocabulary errors).

Since errors are unavoidable, users need support to correct them. At the very least, some mechanism to correct errors is required. Furthermore, error correction may be carried out in the same or a different modality. Karat et al. (1999) observe that correcting speech-recognition errors using speech alone is problematic since it may lead to cascading errors when utterances intended to correct errors are themselves misrecognized. One solution is to use a correction interface that uses a different modality, such as a touch-screen. Figure 4 shows a word confusion network correction interface for a mobile speech-recognition system (Vertanen and Kristensson 2009). The best hypothesis is shown at the top row. The individual words in the best hypothesis can be changed by touching alternative words in the corresponding row positions. A word can be deleted by touching the X button in the corresponding row. The words represent the best hypothesis in the recognition result

If it is desirable to support hands-free error correction in speech recognition the problem becomes more challenging. McNair and Waibel (1994) propose a technique that enables users to correct speech-recognition errors by simply respeaking the intended words. In previous systems in the literature, users had to first select the error region in the text by speaking the misrecognized text. Thereafter, users replaced the text by respeaking their intended words. McNair and Waibel (1994) explain that their one-step method better resembles how we resolve misunderstandings in our daily human-human communication.

Depending on the circumstances, efficient error correction techniques can be critical for the success of a system. As an example, we carried out a formative study where users dictated text while they walked around outdoors. Our users were able to enter text at 15 words per minute with a corrected word-level error rate of about 2 percent, despite the fact that the word-level error rate of the speech recognizer was 26 percent (Vertanen and Kristensson 2009).

In conclusion, we are still searching for efficient error correction interfaces. Well implemented, they improve entry rates and reduce frustration among users.

Challenge 3: Editor Support

Effective error correction methods ultimately depend on good editor support. Modern operating systems are designed for text entry by means of a desktop keyboard. Whenever a user types an individual key the keyboard's scan code is translated into a keyboard-independent key code that is stored in a queue and sequentially dispatched to the window the user is currently working on. If the user makes a mistake the user presses the Delete and Backspace keys, which are processed and dispatched in the same manner as all other key presses.

The aforementioned process is one directional and there is no reliable way for a text entry method to retrieve text that has previously been dispatched to the user's working window. When text entry is operating on a unit more complex than a single character, two issues arise.

First, a number of intelligent text entry methods calculate the probability of a recognized word based on previous words already written by the user. If there is no reliable way to detect these previous words in the user's document this cripples practical implementations of these methods.

Second, an intelligent text entry method may output several word candidates. However, only the highest ranked word candidate is dispatched to the user's document. Since recognition is error prone, it is convenient to enable users to revisit previously written words and let them explore the alternative choices. Such interaction requires support by operating systems. Unfortunately, this support is lacking. As a result, text entry systems have to

resort to manually monitoring and modeling all user interface messages (Watanabe, Okada, and Ifukube 1998). Such mechanisms are error prone and may not be possible to implement on all operating systems.

This issue is not likely to disappear until intelligent text entry methods, such as speech recognition, become mainstream text entry methods on both desktop and mobile computers. Better toolkits and input method frameworks would mitigate the current situation.

Challenge 4: Feedback

Another aspect that requires attention is user feedback. The immediate feedback users receive from a recognizer is the recognized text presented to them by the system. The quality of recognition is most likely a highly influential feedback mechanism in itself. Curiously, the effect of immediate versus withheld feedback on users' performance with, and perception of, a text entry method does not appear to have been formally studied. Recognition-based text entry methods have a degree of tolerance toward imprecise "sloppy" input. This can result in a positive feedback loop that causes users to write increasingly sloppily as they observe how much the recognizer tolerates. Such an effect may be positive, as it drives users to find their own optimal operating point (Kristensson and Denby 2009).[3] Of course, at a certain point the input will be too noisy for any recognizer, and the system will fail with a recognition error. Figure 5 exemplifies how we tried to preempt this situation by visualizing to users how close their input gestures are to the recognized ideal templates in ShapeWriter (Kristensson and Zhai 2004).

Annotating Words Based on Confidence.
In most modern word processing applications a spelling checker annotates words not found in a dictionary. The popularity of spelling checkers indicates that users perceive them as effective when using a text entry method, such as the desktop keyboard. With a character-per-character text entry method, it is easy to make mistakes on the character level. Such errors can be reliably detected by a spelling checker.

In contrast, intelligent text entry methods often work by recognizing and outputting entire individual words into a document. This effectively eliminates spelling errors and the need to detect them. However, it does not necessarily mean that words outputted by the recognizer are words intended by the user. Since these unintended words are undetectable by spelling checkers, researchers have tried to use other sources of information to detect them. Aside from a grammar checker, another source of information is the recognizer's confidence scores. If these confidence scores are sufficiently accurate, they can be used as a basis to annotate words. An early example of the use of confidence visualization, possibly the first, is described by Schmandt (1981). He presents a speech-recognition system that displayed recognized key words in different shades depending on the recognizer's confidence scores.

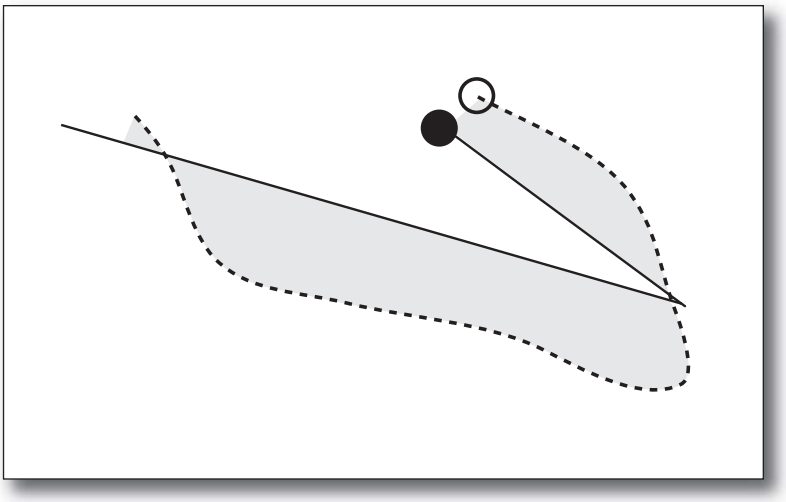

Figure 5. Ideal Shape in Comparison to User's Input.

The ideal shape of the word *the* (indicated by a solid polyline) is visualized in comparison to the user's input gesture (indicated by a dotted spline) in ShapeWriter. For clarity the keyboard is omitted in this illustration.

It is plausible that annotating low-confidence words would help users detect more misrecognized words.[4] To investigate this hypothesis, we built a speech-recognition user interface that annotated recognized words based on the speech recognizer's confidence values (figure 6). Our experiment showed that annotating words based on their confidence did not result in a measurable difference in the number of errors users were able to detect and correct. Examining the material further, we found that misrecognized words were indeed found significantly more often when they were annotated in comparison to a baseline where no words were annotated at all. However, unannotated misrecognized words in the annotation condition were also discovered less often than in the baseline. In other words, users tended to look only at the annotated words. Unless confidence scores are highly accurate, this means that users will not be able to identify that many misrecognized words, since these will often fail to be correctly annotated by the system (Vertanen and Kristensson 2008).

Galletta et al. (2005) reached a similar conclusion when they investigated the efficacy of spelling checkers. They found that when spelling checkers correctly annotated misspelled words users indeed benefited. However, they also found that users tended to trust spelling checkers too much. This meant that when the spelling checker made a mistake users tended to believe the spelling checker and made incorrect changes prompted by it. In

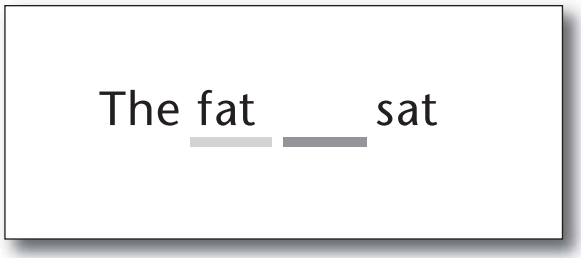

Figure 6. Example of Confidence Visualization.

The user's utterance has been recognized and the inferred words are shown to the user. The words are annotated based on the recognizer's confidence. The intensity of the gray underlining is proportional to the recognizer's belief that the underlined word is misrecognized. The large underlined gap represents the recognizer's belief that a word is likely missing in the recognition result.

addition, if the spelling checker was unable to detect a spelling error, users tended to not detect the error themselves.

In conclusion, until confidence scores are more accurate they are unlikely to aid users. We have shown that if confidence scores are accurate enough they do indeed benefit users (Vertanen and Kristensson 2008). However, to reach such levels it may be necessary to incorporate external knowledge outside the speech recognizer. Since speech recognizers already use statistical knowledge of the preceding word context, an alternative source might for example be commonsense reasoning. Commonsense reasoning has earlier been explored as a means to disambiguate between speech-recognition hypotheses (Lieberman et al. 2005).

Challenge 5: Context of Use

The last challenge is to design for the heterogeneous contexts that users of mobile devices are exposed to on a daily basis. The by far most common experimental situation is to let the participant sit down in a quiet office and enter a set of given phrases or sentences. This experimental setup has completely dominated mobile text entry research, with very few exceptions. However, as modern mobile phones and their text entry methods become increasingly sophisticated, users are probably going to expect that they will be able to enter text at reasonable speeds, even on a shaky bus or when walking down a street. Suddenly a high entry rate and a low error rate is not enough. A text entry method will be required to also understand and support users' needs in particular situations. For example, if the user is walking toward the departure gate, a speech recognizer may be the most effective text entry method. On the other hand, if the user is attending a committee meeting a completely silent text entry method is desired. A context-aware system could model such situations and automatically switch input modality.

Oulasvirta et al. (2005) carried out one of the first comprehensive studies of users interacting with their mobile devices while walking around outdoors. They discovered that users cope with their environment in fragmented bursts, constantly shifting attention between their device and their surroundings. This finding inspired the design of our static error correction interface in our mobile speech-recognition system Parakeet (Vertanen and Kristensson 2009).

Proper modeling of different contexts of use raises many currently unanswered questions. We do not have sufficient knowledge of how users use, and would like to use, mobile text entry methods today. Nor do we know which modalities they prefer in different situations and contexts. A seamless multimodal text entry system needs to be able to accurately sense and reason about its surrounding based on models of both the environment and the user's preferences. It will require a user interface that enables users to easily monitor, understand, and control its behavior. Such multimodal adaptive text entry systems provide users with more flexible options than what is currently offered.

Walking the Last Mile

It is worth noting that a clever algorithm does not guarantee a successful intelligent text entry method. Even though it may be possible to show theoretical quantitative advantages with certain approaches, there is always a risk that the user interface is cumbersome, or that the algorithm is perceived as brittle by users.

An example is our work on designing the elastic stylus keyboard that was presented earlier in this article (Kristensson and Zhai 2005). The design of this system followed an iterative design process. Two user studies were conducted using an early incarnation called the linear correction system. The primary difference between the linear correction system and the final design was that the former could not cope with insertion and deletion errors (the user omitting a tap, or introducing a spurious extra tap). The efficacy of the linear matching algorithm was evaluated in a user study. Theoretically, it should enable users to write more quickly than a traditional system since it provides a certain degree of error tolerance. However, in the user study the system achieved almost the same entry rate as a baseline system that performed no corrections at all. When analyzing the material further, we discovered that while the system did automatically correct the majority of users' errors, it also caused a couple of errors that most likely made users trust the system less. It is well established that interactive systems that do not behave as expected undermine users' trust in them (Tseng and Fogg 1999). Thus, this deficiency resulted in a system

that was perceived as brittle by users. For example, if a single tap was accidentally omitted a completely different word would be outputted, which puzzled users.[5] Since users did not trust the system they chose to be overly careful. As a result, the text entry rate did not increase as expected.

In contrast, the final elastic stylus keyboard system took into account the fact that users may omit or accidentally add spurious taps. As a result, the final system eliminated almost all errors detected in the first user study. This indicates that marginal improved performance of the AI algorithm can dramatically determine the end-user benefits of the intelligent text entry method.[6] It also shows that theoretical benefits remain theoretical until they are empirically verified. In the end, engineering and evaluation have to go hand in hand if we hope to create better systems.

Conclusions

During the last decades, several intelligent text entry methods have been conceived, improved upon, and evaluated by researchers from both the AI and HCI communities. Unlike perhaps most other research on interactive systems, the text entry research field has maintained close connections to AI and engineering. A possible explanation is that text entry methods have always had clear objective functions insofar that a good text entry method has a high entry rate and a low error rate.

However, these objective functions have been optimized to a point where further progress is most likely incremental. In this article, I have presented five challenges that set a broader view on intelligent text entry. It is not enough to design a clever algorithm and show empirical benefits. The motivation behind text entry research is to ensure we have practical writing systems available for everyone, everywhere. Thus, successful text entry methods need to work in the real world. Researchers need to walk the last mile and ensure their methods have a purpose in our society. In this process, several neglected, but often critical, challenges arise that may very well be deciding factors in the success of a text entry method. By addressing these challenges, intelligent text entry methods will have greater impact in our society.

Acknowledgments

I thank Keith Vertanen and David MacKay for their assistance. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 220793.

Notes

1. The use of a keyboard layout optimized for efficient one-finger typing can increase the asymptotic performance of expert users, but it creates a learning hurdle for beginning users. The usability side-effects theme article in this issue discusses the special importance of avoiding excessive learning demands during early use. Offering a familiar, though less efficient, keyboard with a new text entry technique is consistent with this point.

2. The remainder of this section provides concrete illustrations of many of the general points made in the section on errors in the theme article on usability side effects.

3. The attempt of users to find their own "ideal operating point" can be seen as an example of a type of learning discussed in the usability side-effects theme article: learning which of various alternative ways of using the system works best for the user in question (and perhaps in particular situations). The novelty of this learning problem in the case of ShapeWriter is typical, in that it is due to the novelty of the intelligent technology being introduced. The visualization shown in the figure is an ingenious way of speeding up the learning by providing pertinent feedback.

4. The need to learn to make appropriate use of the confidence values of an interactive intelligent system is another example of the difficulty that users face in figuring out how to take advantage of what is offered by an intelligent system. It illustrates the worst case, discussed in the section on learning in the usability side-effects theme article, in which even extensive experience may not enable a user to discover the best method. Note that the researchers cited here had to perform a controlled study to determine the consequences of relying on confidence values; normal users could hardly be expected to reach reliable conclusions on their own.

5. This response of users to unexpected system behavior illustrates a point made in the section on comprehensibility in the usability side-effects theme article: a user's operation of a system can be influenced by the user's ability to understand and predict the system's responses to his or her actions. In this example, lack of predictability and comprehensibility led to overly cautious user behavior that did not fully exploit the potential of the system's intelligent interpretation algorithms.

6. This example illustrates a point made in the section on errors in the usability side-effects theme article: instead of globally maximizing accuracy according to some standard metric, researchers and designers should find out what particular errors are likely to occur and what consequences are associated with them, so as to be able to ensure that the system's performance parameters and correction mechanisms handle the most important cases effectively.

References

Clarkson, E.; Clawson, J.; Lyons, K.; and Starner, T. 2005. An Empirical Study of Typing Rates on Mini-QWERTY Keyboards. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, 1288–1291. New York: Association for Computing Machinery.

Clawson, J.; Lyons, K.; Rudnick, A.; Iannucci, R. A.; and Starner, T. 2008. Automatic Whiteout++: Correcting Mini-QWERTY Typing Errors Using Keypress Timing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 573–582. New York: Association for Computing Machinery.

Crossan, A.; Murray-Smith, R.; Brewster, S.; Kelly, J.; and

Musizza, B. 2005. Gait Phase Effects in Mobile Interaction. In *Extended Abstracts on Human Factors in Computing Systems,* 1312–1315. New York: Association for Computing Machinery.

Darragh, J.; Witten, I. H.; and James, M. L. 1990. The Reactive Keyboard: A Predictive Typing Aid. *IEEE Computer* 23(11): 41–49.

David, P. A. 1985. Clio and the Economics of QWERTY. *American Economic Review* 75(2): 332–337.

Fitts, P. M. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology* 47(6): 381–391.

Galletta, D. F.; Durcikova, A.; Everard, A.; and Jones, B. M. 2005. Does Spell-Checking Software Need a Warning Label? *Communications of the ACM* 48(7): 82–86.

Goodman, J.; Venolia, G.; Steury, K.; and Parker, C. 2002. Language Models for Soft Keyboards. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence,* 419–424. Menlo Park: AAAI Press.

Gould, S. J. 1992. The Panda's Thumb of Technology. In *Bully for Brontosaurus*, 59–75. London: Penguin Books.

Janson, T. 2004. *Latin—Kulturen, Historien, Språket.* Stockholm: Wahlström and Widstrand.

Karat, C.-M.; Halverson, C.; Horn, D.; and Karat, J. 1999. Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 568–575. New York: Association for Computing Machinery.

Kristensson, P. O. 2007. Discrete and Continuous Shape Writing for Text Entry and Control. Ph.D. Dissertation, Linköping University, Sweden.

Kristensson, P. O., and Denby, L. C. 2009. Text Entry Performance of State of the Art Unconstrained Handwriting Recognition: A Longitudinal User Study. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 567–570. New York: Association for Computing Machinery.

Kristensson, P. O., and Zhai, S. 2004. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-Based Computers. In *Proceedings of the Seventeenth ACM Symposium on User Interface Software and Technology*, 43–52. New York: Association for Computing Machinery.

Kristensson, P. O., and Zhai, S. 2005. Relaxing Stylus Typing Precision by Geometric Pattern Matching. In *Proceedings of the Tenth ACM International Conference on Intelligent User Interfaces*, 151–158. New York: Association for Computing Machinery.

Lieberman, H.; Faaborgand, A.; Daher, W.; and Espinosa, J. 2005. How to Wreck a Nice Beach You Sing Calm Incense. In *Proceedings of the Tenth international Conference on Intelligent User Interfaces*, 278–280. New York: Association for Computing Machinery.

MacKenzie, I. S., and Soukoreff, R. W. 2002. Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human-Computer Interaction* 17(2): 147–198.

Masui, T. 1998. An Efficient Text Input Method for Pen-Based Computers. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 328–335. New York: Association for Computing Machinery.

McNair, A. E., and Waibel, A. 1994. Improving Recognizer Acceptance through Robust, Natural Speech Repair. In *Proceedings of the Third International Conference on Spoken Language Processing*, 1299–1302. Grenoble: ISCA.

Melin, O. W. 1927. *Stenografiens Historia, Första Delen* (The History of Shorthand). Stockholm: P. A. Norstedt and Söner.

Oulasvirta, A.; Tamminen, S.; Roto, V.; and Kuorelahti, J. 2005. Interaction in 4-Second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 919–927. New York: Association for Computing Machinery.

Plamondon, R., and Srihari, S. N. 2000. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(1): 63–84.

Rosenbaum, D. A. 1991. *Human Motor Control*. San Diego: Academic Press.

Salvucci, D. 1999. Inferring Intent in Eye-Based Interfaces: Tracing Eye Movements with Process Models. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 254–261. New York: Association for Computing Machinery.

Schmandt, C. 1981. The Intelligent Ear: A Graphical Interface to Digital Audio. In *Proceedings of the IEEE International Conference on Cybernetics and Society*, 393–397. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27: 379–423, 623–656.

Shieber, S. M., and Nelken, R. 2007. Abbreviated Text Input Using Language Modeling. *Natural Language Engineering* 13(2): 165–183.

Tappert, C. C.; Suen, C. Y.; and Wakahara, T. 1990. The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(8): 787–808.

Tseng, S., and Fogg, B. J. 1999. Credibility and Computing Technology. *Communications of the ACM* 42(5): 39–44.

Vertanen, K., and Kristensson, P. O. 2009. Parakeet: A Continuous Speech Recognition System for Mobile Touch-Screen Devices. In *Proceedings of the Fourteenth ACM International Conference on Intelligent User Interfaces*, 237–246. New York: Association for Computing Machinery.

Vertanen, K., and Kristensson, P. O. 2008. On the Benefits of Confidence Visualization in Speech Recognition. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1497–1500. New York: Association for Computing Machinery.

Ward, D. J., and MacKay, D. J. C. 2002. Fast Hands-Free Writing by Gaze Direction. *Nature* 418: 838.

Watanabe, T.; Okada, S.; and Ifukube, T. 1998. Development of a GUI Screen Reader for Blind Persons. *Systems and Computers in Japan* 29(13): 18–27.

Zhai, S., and Kristensson, P. O. 2003. Shorthand Writing on Stylus Keyboard. In *Proceedings of the ACM Conference on Human Factors in Computing Systems,* 97–104. New York: Association for Computing Machinery.

Per Ola Kristensson is a junior research fellow at the University of Cambridge and a cofounder of ShapeWriter, Inc. He received his Ph.D. in computer science from Linköping University in 2007.