Next-Generation Text Entry: Allowing Users to Flexibly Combine Speech, Typing and Gesturing

Users are reluctant to learn new text entry methods, as they demand a substantial training investment. Allowing users to flexibly combine multiple existing probabilistic text entry modalities is a way to still provide performance benefits to users.

The history of text entry spans thousands of years and ranges from etching symbols into stones and wax tablets to typing on touchscreens and speaking into microphones. With the exception of the full-sized laptop or desktop keyboard, sensor-based text entry methods are now widely used, in particular for mobile text entry. Such text entry methods infer users' intended text from noisy signals using statistical models. As we transition into an age where computing is everywhere and mobile technology has become an integral part of our everyday interactions [1] it is natural to ask how we will enable people to efficiently enter text in a variety of challenging situations, such as when users are mobile, encumbered, or generally do not have convenient access to a full-sized keyboard. As technology evolves and users expect to be able to efficiently interact with a range of new user interface technologies, ranging from smart watches to wall-sized displays to different ways of appropriating users' own bodies as interaction surfaces [4], designing efficient and compelling text entry methods becomes very challenging.

Text entry is a complex process and as a consequence a usable text entry method demands substantial user investment. This has resulted in users becoming reluctant to adopt radically different text entry methods that demand more than a few minutes of training investment. For this reason the next step in text entry is unlikely to be a fundamentally different method of entering text into a computer. However, the availability of pervasive and inexpensive high-quality sensors in combination with progress in signal processing and machine learning has resulted in a wide range of probabilistic text entry methods. An example is the typical touchscreen keyboard's automatic error correction algorithm, which calculates posterior probabilities for words by combining likelihoods from a touch model with prior probabilities from a statistical language model. Another example is speech recognition, which assigns posterior probabilities to word sequences by combining likelihoods from an acoustic model with prior probabilities from a statistical language model. Currently these and other probabilistic text entry methods focus almost exclusively on achieving high performance single-modality input, primarily by lowering error rates.

In this article I argue that an alternative avenue for research is to leverage the hypothesis spaces of probabilistic text entry methods to provide users with flexible error correction mechanisms and an ability to combine speech, typing and gesturing. Such designs can potentially provide users with high performance gains without forcing users to invest time in learning new text entry methods.

Designing for Mainstream Success

Literally hundreds of different text entry methods have been proposed in the last two decades, primarily for mobile devices. However, few text entry methods have succeeded to achieve mainstream user adoption. In mobile text entry, seven mainstream text entry methods have been adopted by users to some extent: 1) Graffiti/Unistrokes [3]; 2) multi-tap and predictive text based on a telephone keypad;

3) touchscreen keyboard; 4) physical thumb keyboard; 5) gesture keyboard [5]; 6) handwriting recognition; and 7) speech recognition.

These text entry methods share two critical traits that separate them from the vast majority of text entry methods that have failed to become adopted by users: 1) a high performance compared to other mainstream text entry methods of their generation; and 2) a high degree of similarity with existing mainstream text entry methods. Unsuccessful text entry methods often have one of these traits—but not both.

This analysis suggests that it is not sufficient to achieve wide user adoption by designing a text entry method that provides competitive entry and error rates alone—it is also critical to minimise the amount of effort, or learning, users will need to invest to become proficient with a new text entry method. This leads to a preference to keep a design similar to an existing familiar solution. The QWERTY keyboard is an example of this phenomenon, which is known as path dependency in economics [2]. As a consequence of this constraint, the text entry design space is very narrow.

To design a text entry method unfamiliar to most users, that is, a text entry method that in some way radically departs from a telephone keypad, QWERTY keyboard, handwriting or speech recognition, necessitates a need among users to invest effort to learn the new text entry method. Figure 1 exemplifies this by plotting the performance of a familiar text entry method as a horizontal solid red line. It has constant performance because users have already saturated their learning for this interface. If we now introduce an unfamiliar text entry method (dashed blue line in Figure 1) then we would expect the unfamiliar text entry method to initially exhibit a low performance that gradually improves along a learning curve. If the unfamiliar text entry method eventually provides superior performance in relation to the familiar text entry method, the two lines will eventually cross at the crossover point. At some later point the learning of the unfamiliar text entry method will saturate and provide a maximum performance benefit in relation to the familiar text entry method. For users to invest in learning a new text entry method this benefit must outweigh the time investment that is initially made.

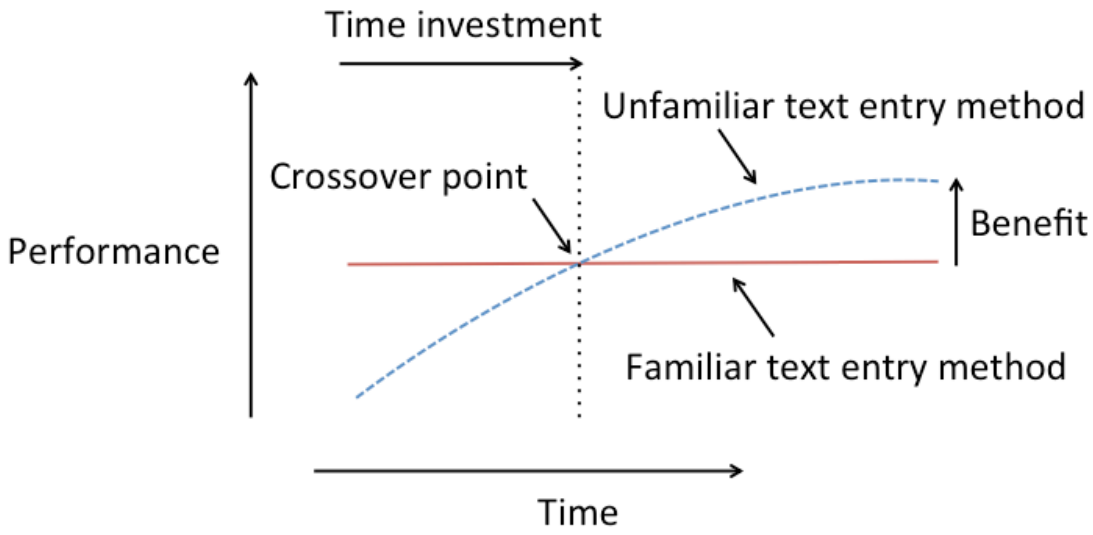


Figure 1: An illustration of the crossover point.

However, in some instances users are unlikely to even reach the crossover point. Consider typing on a tiny device, such as a smart watch. An existing method to type on such a device is ZoomBoard [8] which allows users to type on a familiar touchscreen QWERTY, albeit using two touch interactions instead of one: the first touch zooms into a region of the keyboard and the second touch selects the intended key. It is a simple method that is easy-to-learn, however it requires two closed-loop touch interactions, which makes it slow and performance is limited to about 10 words per minute (wpm). Now assume a user desires to type on a smart watch and has decided to use a new optimised unfamiliar text entry method that is eventually going to provide a performance benefit but demands 40 hours of everyday typing experience for it to become usable at 10 wpm (an example of such a text entry method could for instance be a chording keyboard variant with optimal letter assignments). If we assume the user is casually typing for an average of five minutes per day then it will then take the user 480 days to reach crossover point. In this case the time investment is unlikely to be worth the eventual performance benefit.

The above analysis suggests a radically different text entry method is unlikely to be successful as the cost of relearning how to enter text is too great. However, it is still possible to achieve substantial progress by maximising the potential of widely used existing probabilistic text entry methods.

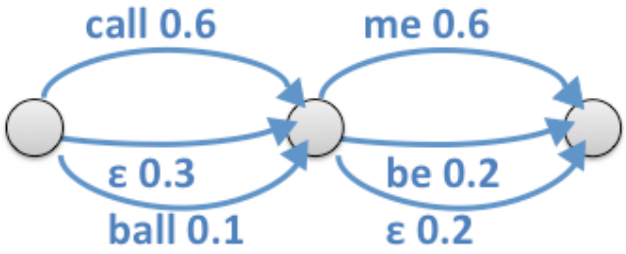
Helping Users Correct Errors

With the exception of the full-sized QWERTY laptop and desktop keyboard, current mainstream text entry methods are probabilistic—they assign posterior probability distributions to letters or words. Even an unmodified regular full-sized QWERTY keyboard can easily be turned into a probabilistic text entry method, by for instance viewing the physical keys as a course pixel grid and modelling typing errors as deviations on this pixel grid. When a probabilistic text entry method infers users' intended text it searches a vast hypothesis space of potentially all possible letter or word sequences in a language. This means there is a (possibly infinite) set of hypotheses of what the user intended to write. Of course, in the end the user is typically only interested in the best hypothesis. However, errors are unavoidable in text entry and when this happens the next best hypotheses in the hypothesis space can be used to reduce frustration and improve error correction performance.

One method to exploit this hypothesis space in design is to convert the hypothesis space to a word confusion network. A word confusion network is a time-ordered series of connected word confusion clusters. Each cluster contains a set of word hypotheses and the probabilities of these word hypotheses sum to one. Figure 2 (left) shows a word confusion network with two word confusion clusters. Epsilon-transitions capture the possibility that no word was intended and if this transition is chosen then no word is generated.

Figure 2 (right) shows how a word confusion network can be arranged in a user interface to allow users to explore the hypothesis space (first suggested by Ogata and Goto [7]). The first row shows the text that is currently outputted by the system. The buttons below represent the most likely word hypotheses from the word confusion network for every time step. Pushing a button changes the corresponding output text. An *X*-button represents an epsilon-transition, which is the hypothesis that no word was intended at this point. Consequently, if the *X*-button is pressed the corresponding

output word at that point is deleted. Note that Figure 2 (right) orders all word hypotheses by probabilities except for epsilon-transitions, which are always assigned to the bottom row for consistency. Other arrangements are certainly possible.



call	me
call	**me**
ball	be
X	X

Figure 2: A word confusion network consisting of two word confusion clusters (left). A user interface representation of the word confusion network (right).

A word confusion network presented as a user interface can improve error correction by allowing users easy access to the next best alternative word candidates in the hypothesis space. A second advantage is that it becomes possible to provide users with additional flexibility in how they use multiple modalities. For instance, a speech recogniser might generate the original hypothesis space and the corresponding word confusion network while a second modality, such as touch-based user interface, can be used to revise the text.

Smarter Ways to Correct Errors

Users currently need to either backspace or select text in order to explicitly provide two crucial pieces of information to the system: 1) a signal to the system that a correction or revision of the text is about to take place; 2) the location of the replacement text. The rich hypothesis space of a probabilistic text entry method can however be used to design more flexible interfaces for users.

For example, assume the user has intended to write “call me now” and this is misrecognised as “call be now”. We have developed an algorithm that allows users to correct the misrecognised word “be” into “me” by simply having the user writing the word “me” without any explicit reference to the error location [6]. The system automatically identifies the erroneous set of words and replaces them with the corrected words. The system is flexible and allows users to provide more context than just the erroneous words. For instance, in the above example the user can also correct the word “be” by providing further context, such as writing “call me”, “me now” or “call me now”. Providing further context is optional but can improve accuracy.

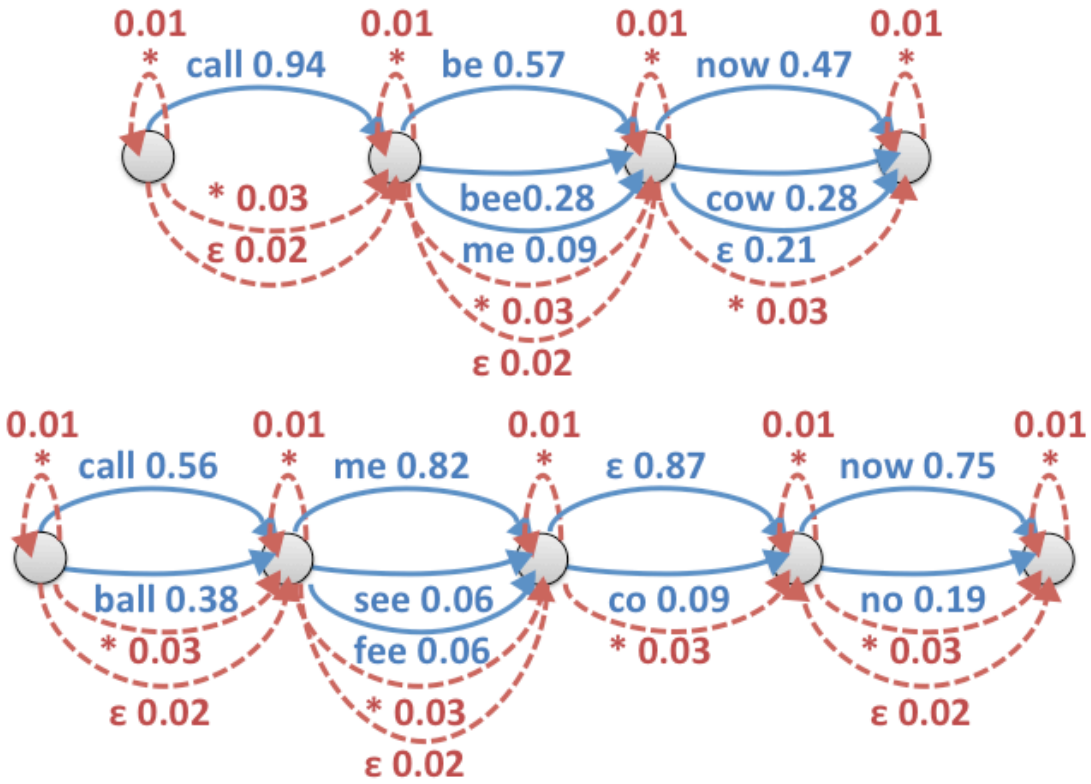


Figure 3: A statistical model for merging two probabilistic text entry modalities.

It is possible to realise this functionality by merging two word confusion networks [6]. The algorithm takes as an input two word confusion networks (blue solid transitions in Figure 3). It then takes some of the probability mass in each cluster and softens the networks by adding epsilon- and wildcard-transitions (red dashed transitions in Figure 3). Epsilon-transitions allow the algorithm to proceed to the next word confusion cluster in the network without generating a word. Wildcard-transitions means that the algorithm can either stay in the same cluster and generate any word (self-loops in Figure 3), or proceed to the next cluster and generate a wildcard word. A wildcard word matches any word. Thereafter the algorithm searches for the highest joint probability path through both networks using a token-passing model [6]. The search space is technically infinite but a search is still viable using beam pruning [6]. The free parameters of the model, such as the epsilon- and wildcard-transition probabilities, are tuned on training data (for details see [6]).

Combining Speech, Typing and Gesturing

Any probabilistic text entry method that can output word confusion networks can be used with the above merge model. For example, a speech recogniser can be the source of the original utterance and a touchscreen keyboard can be used for a subsequent error correction. Since the merge process is asynchronous and does not rely on specific timing data it is also possible to allow a user to simultaneously speak and type the same sentence and have the algorithm leverage both hypothesis spaces in order to maximise the probability that the user's intended text is correctly inferred. Our experiments show that we can reduce the word error rate by 53% relative by merging speech and gesture keyboard entry [6]. When the user is only using the gesture keyboard to correct the erroneous words in the speech modality, the word error rate can be reduced by 44% relative [6].

Allowing users to seamlessly switch between different text entry methods is useful because different text entry methods tend to have different performance envelopes. Figure 4 shows an example of two performance envelopes where each circle represents an entry rate-error rate pair for a particular stimulus sentence written by a user (in practice, there might be tens of thousands of such samples). In general, the faster users write, the more errors occur. This is an example of the speed-accuracy trade-off of human performance. The shapes of the performance envelopes reveal the performance characteristics, and trade-offs, between the text entry methods. For example, a text entry method such as speech recognition might be fast when there are no errors but result in a severely lowered performance when users are forced to correct errors (red solid circles in Figure 4). In contrast, a text entry method such as a touchscreen keyboard may not offer as high peak entry rates but will on the other hand provide a more graceful degradation in the presence of errors (dashed blue circles in Figure 4). This suggests that different text entry with different performance characteristics can complement each other depending on, among other things, how easy it is correct errors and how important it is to reach high entry rates. These objectives are likely to change depending on a user's particular text entry situation. By leveraging the hypothesis space of probabilistic text entry methods it is possible to design interfaces that allow users to choose these operating points on the fly on their own.

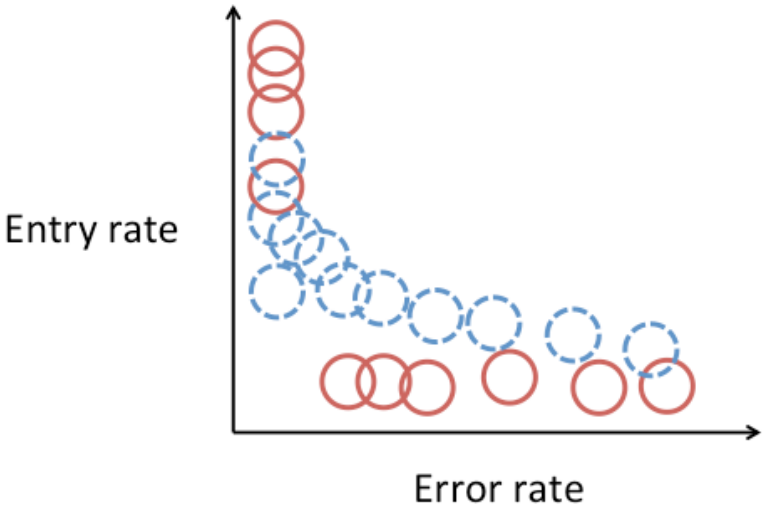


Figure 4: A comparison of two hypothetical performance envelopes for two text entry methods.

Conclusions

Allowing users to flexibly combine multiple probabilistic text entry modalities is likely to become the future of everyday text entry for two reasons. First, no text entry method is appropriate in all situations. Speech can be very fast but suffers from severe performance degradation in the presence of errors. In addition, there are also privacy implications. Touchscreen keyboards and gesture keyboards do not have any privacy implications and can leverage direct manipulation techniques, including touch interaction techniques illustrated in this article, to allow users to quickly correct errors. However, they are not as fast as speech, which suggests both of these text entry methods are individually best suited for particular situations.

Second, a text entry method that dramatically reimagines the way users write is unlikely to achieve wide user adoption. As we saw in the analysis earlier in this article, an unfamiliar text entry method requires substantial user investment in order to reach the crossover point and to provide a performance benefit. So far no unfamiliar text entry method has succeeded in achieving widespread adoption, possibly because users believe the trade-off between training investment vs. eventual performance benefit is unfavourable.

However, by allowing users to flexibly combine several probabilistic text entry methods, we can still provide users with substantial performance gains by allowing users to seamlessly switch modalities and correction strategies in order to optimise their own personal desired operating point for a particular situation. For example, a user who is momentarily encumbered might temporarily want to switch from mobile keyboard entry to speech. As we move into a future with computing becoming an ever increasingly part of the fabric of everyday life, such flexibility allows users to seamlessly switch from speaking to typing or gesturing words on the spot and communicate via a range of smart devices in a wide variety of situations. For instance, progress in wearable technology is likely to enable users to seamlessly use keyboards in thin air via for instance depth sensing technology. In such a future, in which users are increasingly relying on inherently uncertain probabilistic text entry methods, providing flexibility across modalities is going to become even more important.

Biography

Dr Per Ola Kristensson is a University Lecturer in the Department of Engineering at the University of Cambridge, United Kingdom, where he leads the Intelligent Interactive Systems group. He is a co-inventor of the gesture keyboard (commercialised as ShapeWriter, Swype, T9 Trace, etc. and available by default on Android phones as 'gesture typing') and a co-founder of ShapeWriter, Inc., which was acquired by Nuance Communications in 2010. Recent external recognitions include being listed as an *MIT Technology Review* Innovator Under 35 (TR35) in 2013, the Royal Society of Edinburgh Sir Thomas Makdougall Brisbane Medal (2013) and the ACM User Interface Software and Technology (UIST) Lasting Impact Award (2014).

References

1. Abowd, G.D. 2012. What next, Ubicomp? Celebrating an intellectual disappearing act. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp 2012)*. ACM Press: 31-40.

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

3. Goldberg, D. and Richardson, C. 1993. Touch-typing with a stylus. In *Proceedings of the INTERACT 1993 and CHI 1993 Conference on Human Factors in Computing Systems (CHI 1993)*. ACM Press: 80-87.

4. Harrison, C., Tan, D. and Morris, D. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010).* ACM Press: 453-462.

5. Kristensson, P.O. and Zhai, S. 2004. SHARK^2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 2004)*. ACM Press: 43-52.

6. Kristensson, P.O. and Vertanen, K. 2011. Asynchronous multimodal text entry using speech and gesture keyboards. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech 2011)*. ISCA: 581-584.

7. Ogata, J. and Goto, M. 2005. Speech repair: quick error correction just by using selection operation for speech input interfaces. . In *Proceedings of the 6th Annual Conference of the International Speech Communication Association (Interspeech 2005)*. ISCA: 133-136.

8. Oney, S., Harrison, C., Ogan, A. and Wiese, J. 2013. ZoomBoard: a diminutive QWERTY soft keyboard using iterative zooming for ultra-small devices. In *Proceedings of the 31st ACM Conference on Human Factors in Computing Systems (CHI 2013).* ACM Press: 2799-2802.