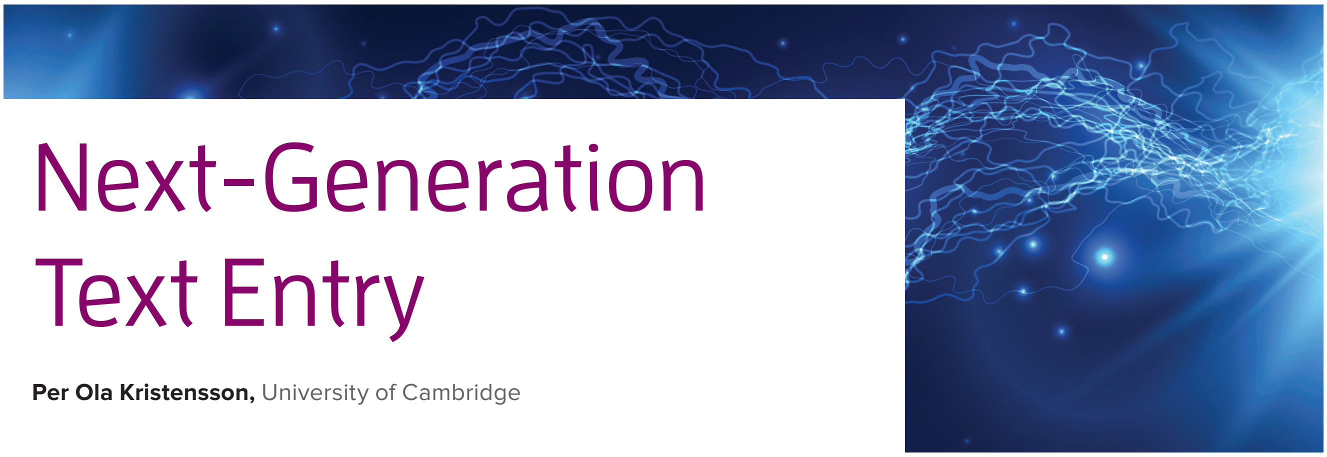


Next-Generation Text Entry

Per Ola Kristensson, University of Cambridge

Letting users seamlessly switch among typing, speaking, and gesturing will facilitate communication via emerging smart devices in a wide variety of situations.

With computing ubiquitous and mobile technology an integral part of our everyday interactions,[1] text entry has become increasingly problematic for interface designers. Users need to be able to enter text when they're on the go, encumbered, lack access to a full-size keyboard, and in other challenging situations. They expect to be able to efficiently interact with a wide variety of evolving interface technologies, including smart watches and other wearables, wall-sized and tabletop displays, and even their own bodies as interaction surfaces.[2]

Text entry is a complex process that can demand a substantial user investment in time to learn. Consequently, people are reluctant to adopt radically different text-entry methods that require more than a few minutes of training. However, the availability of inexpensive high-quality sensors combined with recent 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 word probabilities 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. Here, I argue that a superior alternative is to leverage the hypothesis spaces of multiple probabilistic text-entry methods to provide users with flexible error-correction mechanisms and an ability to integrate speech, typing, and gesturing. Such an approach can provide higher performance without forcing users to invest considerable time in learning new text-entry techniques.

DESIGNING FOR MAINSTREAM SUCCESS

Hundreds of text-entry methods have been proposed in the last two decades, primarily for mobile devices. However, few have achieved mainstream user adoption. In the mobile context, the seven most popular methods are

- single-stroke letter alphabets,[3]
- multitap and predictive text entry on a keypad,
- a touchscreen keyboard,
- a physical thumb keyboard,
- a gesture keyboard,[4]
- handwriting recognition, and
- speech recognition.

All of these text-entry methods share two critical traits: good performance and high similarity to other mainstream methods. Unsuccessful approaches often have one of these

EDITOR ANTTI OULASVIRTA
Aalto University; antti.oulasvirta@aalto.fi

traits, but not both. Thus, providing competitive entry and error rates alone is insufficient to achieve widespread adoption; it's also critical to minimize the user effort required to become proficient. This implies that any future solution must be similar to a familiar one—a phenomenon known in economics as *path dependency*.[5] As a result of this constraint, the text-entry design space is very narrow.

As Figure 1 shows, users must invest significant time to learn a text-entry method that substantially departs from a familiar one such as a QWERTY keyboard or speech recognition. The familiar method's performance is flat because user learning is already saturated; in contrast, the unfamiliar method's performance follows a learning curve. Initially, the new method's performance is lower, but if it eventually proves superior, it will surpass the familiar method's performance at the *crossover point*. Ultimately, user learning of the new method will saturate and provide a maximum performance benefit. For a new text-entry method to be considered worth learning, this benefit must be worth the initial time investment.

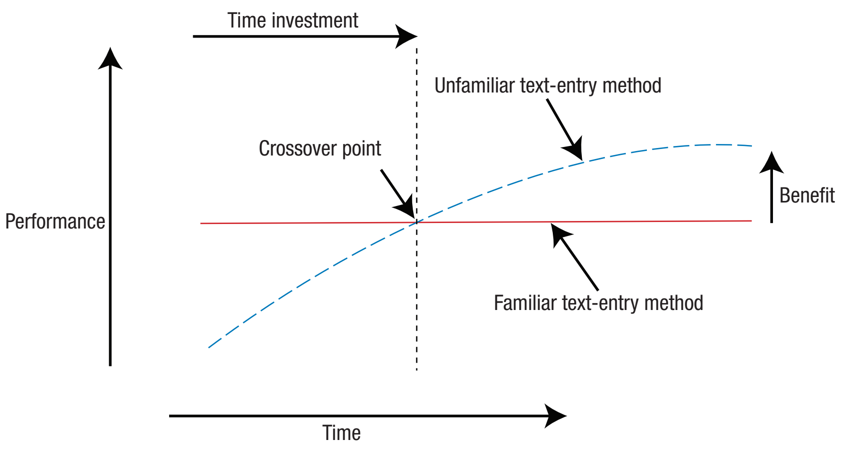


Figure 1. For a new text-entry method to be worth learning, the eventual maximum performance benefit to users must be worth the initial time investment.

However, in some cases, users are unlikely to reach the crossover point. Consider, for example, text entry for a tiny device such as a smart watch. ZoomBoard[6] is an existing technology that lets users type on the device using a QWERTY touchscreen, albeit with two interactions instead of one: the first touch zooms in on a region of the keyboard, and the second selects the intended key. The two closed-loop touch interactions make performance slow, limiting text entry to about 10 words per minute (wpm), but the method is simple and easy to learn. Suppose a new text-entry method, such as a chorded keyboard variant with optimal letter assignments, promises better performance but demands 40 hours of typing experience just to achieve a comparable 10 wpm. Assuming the user casually types for 5 minutes per day, it would take 480 days to reach the crossover point. Such a time investment is unlikely to be worth the eventual performance benefit.

A radically novel text-entry method is thus unlikely to succeed, as the cost of learning how to enter text is too great. However, substantial progress can still be achieved by maximizing the potential of widely used probabilistic text-entry methods.

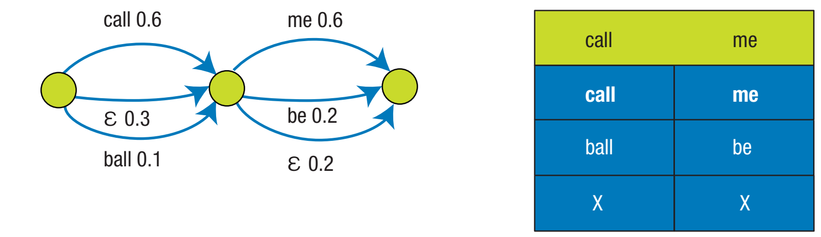
HELPING USERS CORRECT ERRORS

Except for the full-size QWERTY keyboard, current mainstream text-entry methods are probabilistic—they assign posterior probability distributions to letter or word sequences. Such methods can be implemented even on a regular QWERTY keyboard by, for instance, modeling the physical keys as a coarse pixel grid and typing errors as deviations on this grid.

When a probabilistic text-entry method infers a user's intentions, it searches a vast *hypothesis space* of all possible letter or word sequences in a language. Of course, 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 can be used to reduce frustration and improve error-correction performance.

Designers can exploit this hypothesis space by converting it into a *word-confusion network*—a time-ordered series of connected word-confusion clusters. Each cluster contains a set of word hypotheses, and the probabilities of these hypotheses sum to one. The left side of Figure 2 shows a word-confusion network with two clusters. Epsilon transitions capture the possibility that no word was intended, and if this transition is chosen, no word is generated.

The right side of Figure 2 shows how a word-confusion network can be arranged in a user interface to let users explore the hypothesis space.[7] The top row shows the text currently output by the system. The cells below are buttons that represent the most likely word hypotheses from the 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



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

Figure 2. Example word-confusion network consisting of two word-confusion clusters (left), and a user interface representation of the network (right).

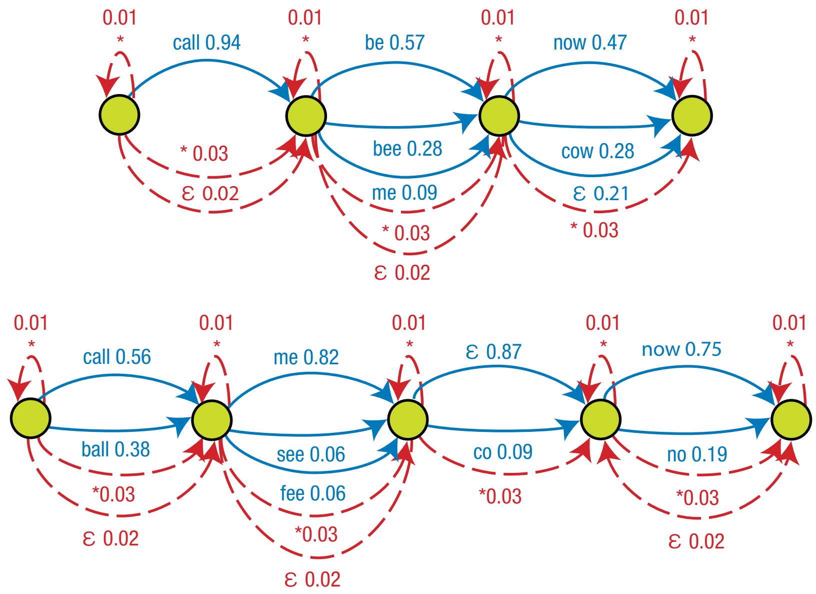


Figure 3. Merging probabilistic text-entry modalities to achieve smarter error correction. In this case, the user intended to type or speak "call me now," but the system inadvertently output "call be now." The merge algorithm takes two word-confusion networks as inputs from different text-entry modalities (blue solid lines) and then, using some of the probability mass in each word-confusion cluster, softens the networks by adding epsilon and wildcard transitions (red dashed lines). Epsilon transitions let the algorithm proceed to the next cluster without generating a word. Wildcard transitions indicate that the algorithm can either stay in the same cluster and generate any word (self-loops), or proceed to the next cluster and generate a wildcard word that matches any word. Thereafter, the algorithm searches for the highest joint probability path through both networks using a token-passing model.

word was intended at this point; if the user presses this button, the system deletes the corresponding output word. Note that all word hypotheses are ordered by probability, except for epsilon transitions, which are always assigned to the bottom row for consistency. Other orderings are certainly possible.

A word-confusion network presented as a user interface can improve error correction by letting users easily access the next-best word candidates in the hypothesis space. It also provides additional flexibility in the use of multiple modalities. For instance, a speech recognizer might generate the original hypothesis space and corresponding word-confusion network, while a touch-based user interface could be used to revise the text.

SMARTER ERROR CORRECTION

When correcting a text-entry error, users currently must either tap on the backspace key or select the mistaken text to signal to the system that an error-correction action is about to occur and to indicate the replacement text's location. However, the rich hypothesis space of probabilistic text entry can be used to design more flexible user interfaces.

For example, suppose a user intends to enter "call me now" using a speech recognizer, touchscreen keyboard, or gesture keyboard but the system instead outputs "call be now." To address this type of error, a colleague and I developed an algorithm that, in this instance, lets users change "be" into "me" simply by typing or speaking "me," without any explicit reference to the error location.[8] Optionally, users can provide more context to improve accuracy—in the above example, the user could type or speak "call me," "me now," or "call me now."

The algorithm realizes this functionality by merging word-confusion networks from different probabilistic text-entry modalities, as Figure 3 shows. In this case, it takes two networks as inputs and then, using some of the probability mass in each word-confusion cluster, softens the networks by adding epsilon and wildcard transitions. Epsilon transitions let the algorithm proceed to the next cluster without generating a word. Wildcard transitions indicate that the algorithm can either stay in the same cluster and generate any word (self-loops), or proceed to the next cluster and generate a wildcard word that matches any word. Thereafter, the algorithm searches for the highest joint probability path through both networks using a token-passing model. The search space is infinite, but a search is still viable via beam pruning. The algorithm's free parameters, such as the epsilon- and wildcard-transition probabilities, are tuned on training data.

COMBINING TYPING, SPEECH, AND GESTURING

Any probabilistic text-entry method that outputs word-confusion networks can be used with our algorithm. For example, a speech recognizer could be the source of the original text, and a touchscreen keyboard could be used for error correction. Because the merge process is asynchronous and doesn't rely on specific timing data, it's also possible for a user to simultaneously speak and type the same sentence and have the algorithm leverage both hypothesis spaces to maximize the probability of correctly inferring the intended text. Our experiments show that merging speech and gesture-keyboard text entry can reduce the word error rate by 53 percent relative, and using only the gesture keyboard to correct misrecognized spoken words can reduce the word error rate by 44 percent relative.[8]

Allowing users to seamlessly switch between different text-entry methods is useful because different 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 entered by a user (in practice, there might be tens of thousands of such samples).

The distribution of the performance envelopes reveals the tradeoffs in speed and accuracy between the text-entry methods. For example, speech recognition is fast when there are no errors but results in dramatically lower performance when users must correct errors (red solid circles). In contrast, a touchscreen keyboard offers lower peak text-entry rates but degrades less with errors (dashed blue circles). This suggests that text-entry methods with different performance characteristics can complement one another depending on the ease of error correction and the importance of high text-entry rates, among other things. The precise objectives are likely to change depending on a user's particular context. Leveraging the hypothesis spaces of multiple probabilistic text-entry methods makes it possible to design interfaces that let users choose their own operating points on the fly.

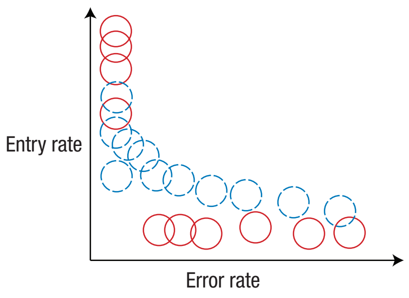


Figure 4. Hypothetical performance envelopes for two text-entry methods. Each circle represents an entry rate–error rate pair for a particular stimulus sentence entered by a user. The distribution of the performance envelopes reveals the tradeoffs in speed and accuracy between the two methods.

Allowing users to combine multiple probabilistic text-entry modalities and error-correction strategies is likely to become everyday practice for two reasons. First, no method is appropriate in all situations. Second, any approach that dramatically reimagines how users enter text is unlikely to achieve widespread adoption, in large part due to the often unfavorable tradeoff between the initial training investment and the eventual performance gain.

Letting users seamlessly switch among typing, speaking, and gesturing will facilitate communication via emerging smart devices in a wide variety of situations. For example, when temporarily encumbered or otherwise unable to type, a person could readily switch from keyboard text entry to speech entry. With users increasingly relying on inherently uncertain probabilistic text-entry methods, such flexibility is going to become even more important in the future. C

REFERENCES

1. G.D. Abowd, "What Next, Ubicomp? Celebrating an Intellectual Disappearing Act," *Proc. 2012 ACM Conf. Ubiquitous Computing* (UbiComp 12), 2012, pp. 31–40.
2. C. Harrison, D. Tan, and D. Morris, "Skinput: Appropriating the Body as an Input Surface," *Proc. 28th ACM Conf. Human Factors in Computing Systems* (CHI 10), 2010, pp. 453–462.
3. D. Goldberg and C. Richardson, "Touch-Typing with a Stylus," *Proc. INTERACT 1993 and CHI 1993 Conf. Human Factors in Computing Systems* (CHI 93), 1993, pp. 80–87.
4. P.O. Kristensson and S. Zhai, "SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-Based Computers," *Proc. 17th Ann. ACM Symp. User Interface Software and Technology* (UIST 04), 2004, pp. 43–52.
5. P.A. David, "Clio and the Economics of QWERTY," *American Economic Rev.*, vol. 75, no. 2, 1985, pp. 332–337.
6. S. Oney et al., "ZoomBoard: A Diminutive QWERTY Soft Keyboard Using Iterative Zooming for Ultra-Small Devices," *Proc. 31st ACM Conf. Human Factors in Computing Systems* (CHI 13), 2013, pp. 2799–2802.
7. J. Ogata and M. Goto, "Speech Repair: Quick Error Correction Just by Using Selection Operation for Speech Input Interfaces," *Proc. 6th Ann. Conf. Int'l Speech Communication Assoc.* (Interspeech 2005), 2005, pp. 133–136.
8. P.O. Kristensson and K. Vertanen, "Asynchronous Multimodal Text Entry Using Speech and Gesture Keyboards," *Proc. 12th Ann. Conf. Int'l Speech Communication Assoc.* (Interspeech 2011), 2011, pp. 581–584.

PER OLA KRISTENSSON, coinventor of the gesture keyboard, is a university lecturer and leads the Intelligent Interactive Systems group in the Department of Engineering at the University of Cambridge, UK. Contact him at pok21@cam.ac.uk.