Speech Dasher: Fast Writing using Speech and Gaze

Keith Vertanen and David J.C. MacKay
Cavendish Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge, CB3 0HE, UK
kv227@cam.ac.uk, mackay@mrao.cam.ac.uk

ABSTRACT
Speech Dasher allows writing using a combination of speech and a zooming interface. Users first speak what they want to write and then they navigate through the space of recognition hypotheses to correct any errors. Speech Dasher's model combines information from a speech recognizer, from the user, and from a letter-based language model. This allows fast writing of anything predicted by the recognizer while also providing seamless fallback to letter-by-letter spelling for words not in the recognizer's predictions. In a formative user study, expert users wrote at 40 (corrected) words per minute. They did this despite a recognition word error rate of 22%. Furthermore, they did this using only speech and the direction of their gaze (obtained via an eye tracker).

Author Keywords
speech recognition, eye tracking, multimodal interfaces

ACM Classification Keywords
H.5.2 User Interfaces: Voice I/O

General Terms
Experimentation, Human Factors, Performance

INTRODUCTION
Speech offers a potentially very fast way to enter text. Users have been measured dictating to a computer at 102 (uncorrected) words per minute (wpm) [3]. But speech recognition is not perfect and any recognition errors will need correction. Previous research has shown that correction time dominates and significantly slows entry rates (e.g. 14 wpm [2] and 17 wpm [3]). A common strategy is to use a non-speech modality for correction such as a keyboard, mouse or stylus [4, 5]. But such modalities may not be an option for people with limited or non-existent use of their hands.

We investigate Speech Dasher, an interface for correcting speech recognition errors by zooming through the many hypotheses proposed by the recognizer. Even if the recognizer completely misses a word, a simple letter-by-letter spelling process allows easy error correction. We focus on a design optimized for use with an eye tracker. We test our design in a preliminary user study investigating how fast expert users can write. To our knowledge, this is the first exploration of text entry using a combination of speech and gaze direction.

CHI 2010, April 10-15, 2010, Atlanta, Georgia, USA.

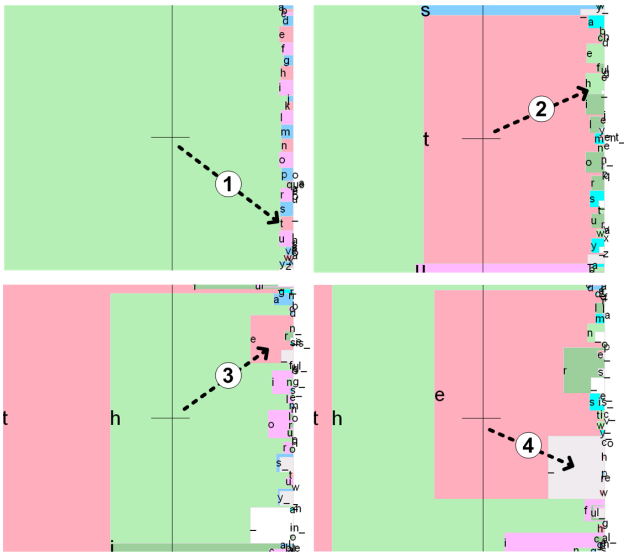

Figure 1. Using original Dasher to write "the" followed by a space. The arrows show the navigation direction needed to write the next symbol.

INTERFACE DESCRIPTION

Original Dasher
Speech Dasher builds on the text entry interface Dasher [6]. In Dasher, users write by zooming through a world of nested letter boxes (figure 1). The size of a box is proportional to that letter's probability under a language model (LM). As more letters are written, Dasher makes stronger predictions that allow common words to be written quickly. All letters appear in each box in alphabetical order. This allows anything to be written – even if the text is not well predicted.

Dasher can be controlled by any type of pointing device. By pointing to the right of the screen midpoint, the user zooms in on letters. The zooming speed is controlled by how close the pointer is to the right side. As a box passes through the midpoint, that box's letter is written. If a mistake is made, pointing to the left reverses zooming and allows text to be erased. The maximum zooming speed is a user chosen parameter. After an hour of practice, users can write at 20 wpm with a mouse [6] or 16–26 wpm using an eye tracker [7].

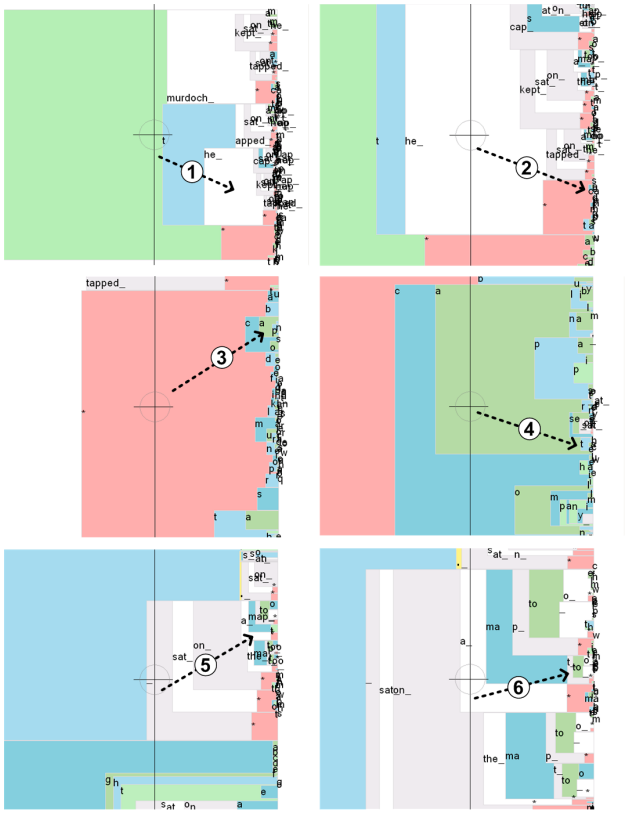

Figure 2. Writing "the cat sat on a mat" in Speech Dasher.

Speech Dasher

We augmented Dasher by adding information from a speech recognizer. In Speech Dasher, users navigate as in original Dasher, but can quickly zoom through entire words. We call the most likely word hypotheses the *primary predictions*. If the primary predictions are incorrect, *secondary predictions* are obtained by going to an "escape" box (a red box with an asterisk). Anything can be written in an escape box using letter-by-letter predictive spelling. The escape boxes utilize both a letter-based LM and the less probable speech results.

Usage Example

Here is a usage example. First, the user turns on the microphone, says "the cat sat on a mat", then turns off the microphone. After recognition completes, the display is updated to show not only the best hypothesis "murdoch kept sat on a map to", but also other likely hypotheses (figure 2, step 1). Navigation now begins in the display with the primary predictions being "murdoch", "tapped" and "the". The user zooms in on the box for "the" (steps 1–2). At this point, the user wants to write "cat" but all the primary predictions are incorrect. By zooming into the red escape box, "cat" is spelled letter-by-letter (steps 3–4). The user now writes "sat on a mat" using words in the primary predictions (steps 5–6).

Interface Elements

The Speech Dasher experimental interface is shown in figure 3. A large rectangular dwell button is located in the lower-left corner. The button changes its label depending on the user's current state. The button is "clicked" if 85% of

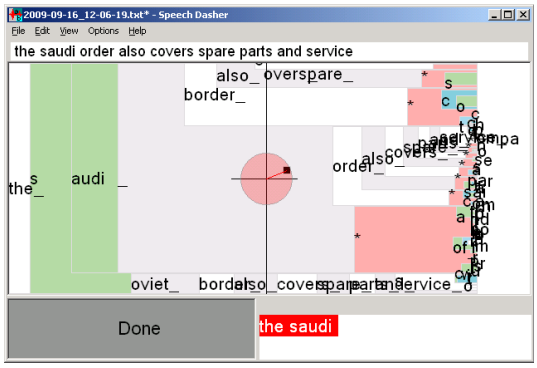

Figure 3. The Speech Dasher interface during error correction.

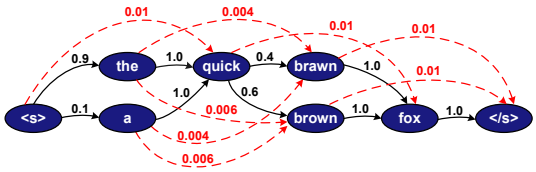

Figure 4. Lattice after new edges were added to cover all one-word insertion errors. The new edges are shown as dashed red lines.

gaze locations in a one second period are inside the button. As the user dwells on the button, its color changes from gray to red. When clicked, the dwell button flashes, beeps, and is deactivated until the user looks away from the button.

In the center of the interface is a circular slow-down region. This region allows users to temporarily stop navigation in order to rest, read text, or move to the dwell button. When the user looks inside the circle, speed is progressively slowed. The circle becomes more red as the speed is dampened. After 1.25 seconds, navigation is stopped. Looking outside the region causes navigation speed to progressively increase.

The top text box shows what the user is being asked to write. The lower-right text box initially shows the best recognition result. If this result is correct, the DONE button can accept the result. Otherwise navigation is started and the text is replaced with the user's writing. When using gaze, checking the result text can disrupt Dasher navigation. To address this, we played every word written using text-to-speech (TTS).

HOW IT WORKS

Recognition and Utilization of Lattice

For speech recognition, we use PocketSphinx [1]. Depending on the user, we use a US or a UK English acoustic model. We use a word trigram LM trained on newswire text. After a user speaks a sentence, the recognition lattice is obtained from PocketSphinx. The lattice is a directed acyclic graph containing the various hypotheses explored during the recognition. We prune the lattice to remove unlikely hypotheses. Next, the acoustic and LM likelihoods on the lattice edges are converted to posterior probabilities. Finally, edges are added to skip over words (figure 4). These skip edges add extra hypotheses covering all one-word insertion errors. The penalty of each added edge was set to a constant α_{ins} multiplied by the penalties of the skipped edges.

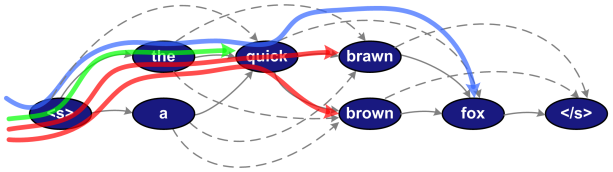

Figure 5. The user has written "the_quiet_". A substitution at "quick" allows the two red paths to reach "brawn" and "brown". A substitution at "quick" also allows the blue path to reach "fox" via the insertion edge between "quick" and "fox". A deletion after "the" allows the green path to reach "quick". Currently we would predict "b", "f" and "q".

Computing the Symbol Probabilities

Each box in Dasher requires a probability distribution over all symbols in the alphabet (including a word boundary symbol denoted by underscore). This is done by first finding the set of lattice paths consistent with the symbol history. For the moment, we will assume that at least one such path exists. So for example, given the lattice in figure 4, if the symbol history is "the_quick_br", there is one path to "brawn" and one path to "brown". Given these paths, the model predicts that the next symbol would be either "a" or "o". A letter's probability is based on the total penalties incurred by its corresponding lattice path.

Formally, let $\boldsymbol{S}$ be the symbol alphabet and $s_1, ..., s_t$ be the t symbols written thus far. We denote a path by ρ and the penalty value of a path by $V(\rho)$. Let $\boldsymbol{C}(s_1, ..., s_{t+1})$ be the collection of paths that are consistent with the symbol history $s_1, ..., s_{t+1}$. The probability of the next symbol is:

$$P(s_{t+1} \mid \text{lat}, s_1, ..., s_t) = \frac{\sum_{\rho \in \boldsymbol{C}(s_1,...,s_{t+1})} V(\rho)}{\sum_{s_x \in \boldsymbol{S}} \sum_{\rho \in \boldsymbol{C}(s_1,...,s_t,s_x)} V(\rho)}.$$

Backoff and Pruning

A sequence of symbols may not be in the lattice. For example, in figure 4 no paths exist for "the_quie". This results in the model predicting zero probability for all symbols in the alphabet. This prevents words not in the lattice from appearing in the primary predictions. Such out-of-lattice words can be written via the secondary predictions (described shortly). For now, assume an out-of-lattice word has been written. After completing the word, Speech Dasher tries to get the user back on track somewhere in the lattice. This is done by assuming the recognizer has made a deletion or substitution error somewhere in the lattice. A new search is initiated with paths allowed to make one error (see example in figure 5). During the new search, paths incur a penalty for using an error (α_{del} for a deletion error, α_{sub} for an substitution error). Using the paths allowed to make an error, the probability distribution over symbols is calculated. If no paths are found using one error, two errors are used, and so on.

We found in early prototypes that showing too many choices made navigation difficult because all but the most probable choices were small in the Dasher display. We changed our design so only the most likely hypotheses appear in the primary predictions. This is done by setting symbols with a probability less than α_{min} to zero and then renormalizing the probabilities. This pruning causes the primary predictions to be highly peaked at only the most probable word hypotheses.

Secondary Predictions

Some words may be missing from the lattice or may have been too improbable to appear in the primary predictions. These words can be written in an escape box that appears only at word boundaries. The escape box interpolates between four models:

- Lattice paths – A search for paths is conducted but with no minimum probability threshold. Additionally, highly probable paths outside the escape box are excluded.
- Uniform lattice paths – A search for paths is done, but the penalty of each path is set uniformly. This allows low scoring paths to still have appreciable probability.
- Language model – A letter-based LM predicts the next symbol based on the previous symbol history.
- Shortened language model – A letter-based LM predicts the next symbol based on a previous symbol history that is truncated at the first word boundary symbol.

We interpolated evenly between the models. We found this provided a consistent user experience inside the escape box while still allowing utilization of less probable lattice paths.

FORMATIVE USER STUDY

Our user study had three goals. First, to test and refine our interface for use with gaze. Second, to investigate Speech Dasher performance for users experiencing different levels of recognition accuracy. Third, to get an estimate of expert performance using both Dasher and Speech Dasher.

Participants and Setup

We conducted a longitudinal study with three users. We chose users we anticipated would have different levels of recognition accuracy due to their accent. Participant US1 was American, UK1 was British, and DE1 was German. Note that UK1 (the second author) had significant experience both with eye tracking and with original Dasher. We include UK1 to show the potential performance for users who are very experienced with Dasher navigation.

Users completed 6–8 training sessions followed by 3 test sessions. The training sessions served to get users to expert performance. It also allowed us to find optimal settings for the eye tracker and to tweak the size and position of interface elements. After the training sessions, all aspects of the interface were fixed. Training and test sessions followed the same procedure. First, users wrote for 15 minutes using Dasher or Speech Dasher. After a break, they wrote for 15 minutes in the other condition. The order of conditions was swapped between sessions. We calibrated the Tobii P10 eye tracker at the start of each session. We used the parameter values: $\alpha_{\text{ins}} = 0.01$, $\alpha_{\text{sub}} = 0.08$, $\alpha_{\text{del}} = 0.01$, $\alpha_{\text{min}} = 0.2$.

Task and Method

Users wrote newswire sentences with 8–12 words. The sentences were shown in the top text box and also played via TTS. At any point, the sentence could be played again by pressing the space bar. Users received sentences in random order. Dasher's maximum zooming speed was adjusted using a slider that users operated with a normal mouse. We encouraged users to adjust the speed throughout the study.

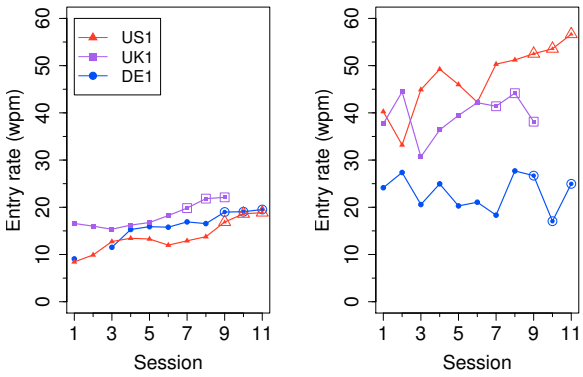

Figure 6. Entry rate for each user in Dasher (left) and Speech Dasher (right). The final test sessions are shown with a double symbol.

In Dasher, using a CORRECT button made the navigation display appear. As the user wrote, letters were output in the lower-right text box. Once the sentence was complete, a DONE button moved to the next sentence.

In Speech Dasher, an utterance was recorded using a MIC ON and a MIC OFF button. After a recognition delay (2.0 s ± 1.1 s), the interface beeped and the navigation display appeared. The best recognition hypothesis was shown in the lower-right text box. If the recognition was completely correct, a DONE button moved to the next sentence. Otherwise, correction proceeded via navigation. After correction, a DONE button moved to the next sentence.

Error Rate Results

Error rate was measured using the word error rate (WER). WER is the word edit distance between the target sentence and what the user wrote divided by the number of words. In the test sessions, the initial recognition results had an error rate of 22%. The WER varied significantly between users with a WER of 7.8% for US1, 12.4% for UK1, and 46.7% for DE1. In both conditions, we measured the error rate of the user's final text. Users left few errors uncorrected. The final WER was 1.3% in Dasher and 1.8% in Speech Dasher.

Entry Rate Results

Entry rate was calculated in words per minute (wpm). In Dasher, we used the time between the end of the dwell on CORRECT and the start of the dwell on DONE. In Speech Dasher, we used the time between the end of the dwell on MIC ON and the start of the dwell on DONE. This time included recording, recognition delays and correction time.

In the test sessions, the average entry rate was 20 wpm in Dasher and 40 wpm in Speech Dasher. In Speech Dasher, users showed a wide range of entry rates, presumably due to their differing recognition error rates. US1 was the fastest at 54 wpm, UK1 was next at 42 wpm, and DE1 was at 23 wpm. On sentences with at least one recognition error, users still wrote at 30 wpm in Speech Dasher. Dasher entry rates improved as the study progressed (figure 6). For US1 and UK1, Speech Dasher was faster than Dasher. As expected, the lower the WER, the faster the entry rate (figure 7).

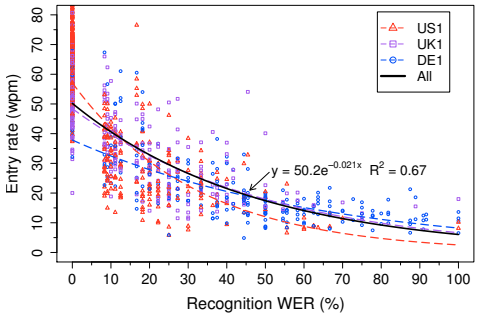

Figure 7. Users' sentences by entry rate and recognition WER.

CONCLUSIONS

We described Speech Dasher, an interface allowing efficient correction of speech recognition errors. We showed how Speech Dasher combined information from the recognizer, from the user, and from a letter-based language model to create an easy-to-use and consistent correction experience.

While our user study was small and used able-bodied users, our preliminary results indicate that speech and gaze may be a promising text entry method for people who cannot use other input modalities. After four hours of practice, users were able to write at 40 wpm despite a recognition WER of 22%. Even on sentences that had a least one recognition error, users were still able to write at 30 wpm. Using Dasher without speech, users were slower, writing at 20 wpm. This shows that Speech Dasher successfully leveraged recognition information to greatly improve users' writing efficiency.

ACKNOWLEDGMENTS

We thank Tobii Technology for use of the eye tracker. We thank Per Ola Kristensson for helpful discussions.

REFERENCES

1. D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky. PocketSphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proc. of ICASSP*, 185–188, 2006.
2. C.-M. Karat, C. Halverson, D. Horn, and J. Karat. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proc. of CHI*, 568–575, 1999.
3. K. Larson and D. Mowatt. Speech error correction: The story of the alternates list. *International Journal of Speech Technology*, 183–194, 2003.
4. S. Oviatt. Taming recognition errors with a multimodal interface. *Comm. of the ACM*, 43(9):45–51, 2000.
5. B. Suhm, B. Myers, and A. Waibel. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8(1):60–98, 2001.
6. D. J. Ward, A. F. Blackwell, and D. J. C. MacKay. Dasher - a data entry interface using continuous gestures and language models. In *Proc. of UIST*, 129–137, 2000.
7. D. J. Ward and D. J. C. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838, 2002.