An Evaluation of Dasher with a High-Performance Language Model as a Gaze Communication Method

Daniel Rough
University of St Andrews
St Andrews, UK
djr53@st-andrews.ac.uk

Keith Vertanen
Montana Tech
Butte, Montana, USA
kvertanen@mtech.edu

Per Ola Kristensson
University of St Andrews
St Andrews, UK
djr53@st-andrews.ac.uk

ABSTRACT

Dasher is a promising fast assistive gaze communication method. However, previous evaluations of Dasher have been inconclusive. Either the studies have been too short, involved too few participants, suffered from sampling bias, lacked a control condition, used an inappropriate language model, or a combination of the above. To rectify this, we report results from two new evaluations of Dasher carried out using a Tobii P10 assistive eye-tracker machine. We also present a method of modifying Dasher so that it can use a state-of-the-art long-span statistical language model. Our experimental results show that compared to a baseline eye-typing method, Dasher resulted in significantly faster entry rates (12.6 wpm versus 6.0 wpm in Experiment 1, and 14.2 wpm versus 7.0 wpm in Experiment 2). These faster entry rates were possible while maintaining error rates comparable to the baseline eye-typing method. Participants' perceived physical demand, mental demand, effort and frustration were all significantly lower for Dasher. Finally, participants significantly rated Dasher as being more likeable, requiring less concentration and being more fun.

Categories and Subject Descriptors

K.4.2 [**Computers and Society**]: Social Issues—*Assistive technologies for persons with disabilities*

General Terms

Human Factors, Performance, Experimentation

Keywords

Assistive gaze communication; eye-typing; Dasher

1. INTRODUCTION

Text entry is an important daily activity among computer users. However, writing text using eye-tracking is difficult. The most common method to enter text by gaze is by eye-typing. To write text using eye-typing, the user serially gazes at the intended keys on an on-screen keyboard. For a key to become activated the user has to fixate on the key for a preset time interval, the so-called dwell-timeout. There has been considerable research into the design, development and evaluation of eye-typing interfaces (e.g. [8, 10]).

Besides eye-typing, researchers have explored alternative ways to write using gaze. For example, one method is to create hierarchical layouts which require users to enter individual letters by activating a series of keys. This enables the interface to maximize button sizes for quick, easy selection (e.g. [2, 13]).

Other attempts to improve entry rates for gaze-based communication move away from the discrete fixation and saccade movement required by eye-typing. *EyeWrite* [20] uses a gesture alphabet that allows users to draw their intended characters with basic gestures.

Unfortunately, both hierarchical layouts and *EyeWrite* result in relatively low entry rates ranging between 5–8 wpm (words per minute) [2, 13, 20].

Recently, Kristensson and Vertanen [5] proposed a new paradigm for writing by gaze that they call *dwell-free eye-typing*. Dwell-free eye-typing might enable users to write fast by removing dwell-timeouts from eye-typing. Instead, users' intended text is decoded using a pattern recognition algorithm. No complete dwell-free eye-typing system has been implemented but a human performance study indicates that such a system might enable users to write faster than 40 wpm [5].

One promising alternative gaze-based text entry method is Dasher [18]. Dasher is a predictive text entry interface in which text is entered by navigating a world of nested boxes. Each box in Dasher is labeled with a letter or other symbol (see Figure 1). The size of each box is proportional to the probability of that box's letter under a language model.

As shown in the upper-left part of Figure 1, boxes with corresponding letters are arranged vertically in alphabetical order. To write a given letter, the user zooms into the box labeled with the desired letter. As the user zooms into the box, subsequent boxes containing further letters will become gradually visible. Boxes of letters with low probabilities might at times be too small to be visible. However, the boxes are there nonetheless and the user can navigate to them by zooming to the appropriate location in the alphabetical ordering.

Zooming is controlled by any sort of pointing device (e.g. a mouse or eye-tracker). Pointing to the right of the screen midpoint causes the display to zoom in. As a box crosses the midpoint of the screen, that box's letter is considered to have been entered. Users can erase previously written text by pointing to the left of the midpoint to zoom back out. The vertical location of the pointer controls exactly which letter is entered. Zooming speed is either controlled manually or via Dasher's automatic speed control feature. The automatic speed control adjusts zooming speed based on the observed

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first p a ge. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
AVI '14, May 27 - 30, 2014, Como, Italy
Copyright 2014 ACM 978-1-4503-2775-6/14/05...$15.00.
http://dx.doi.org/10.1145/2598153.2598157

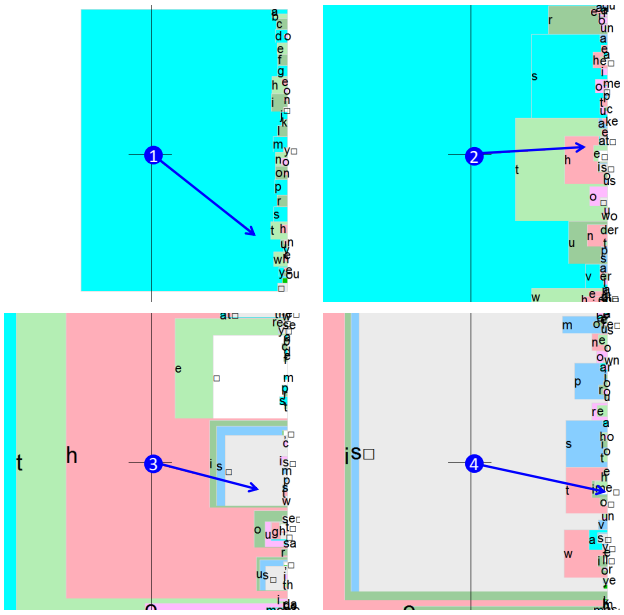

Figure 1: Example of a user writing "this time" in Dasher.

writing proficiency of the user.

Dasher's text entry performance using an eye-tracking interface is unclear. Ward and MacKay [19] tested two expert and two novice users' Dasher performance for 60 minutes. They found that one expert user reached 22 wpm, the other expert user reached 25 wpm, and the two novice users reached entry rates of around 12 and 14 wpm (Figure 1 in [19]). The study is problematic because no control condition was used for the novice users, and the sample was biased as the two expert users were the authors of the study. The study was also too small to be generalizable.

Another study of Dasher was carried out by Tuisku et al. [12]. They recruited 12 Finnish-speaking participants who wrote phrases from the MacKenzie and Soukoreff phrase set [7] translated into Finnish. After 150 minutes of practice, participants reached an average entry rate of 17.3 wpm. However, no control condition was used (such as an eye-typing interface). A further complication is that Dasher's Finnish language model was trained on a Finnish novel called *Pereat Mundus* by Leena Krohn, which is not representative of the type of text users are likely to write using Dasher.

Finally, another study tested three participants' text entry performance with Dasher, with one of the participants being the inventor of Dasher [16]. After 120 minutes of practice, the first and second participants reached 17 wpm, and the inventor of Dasher reached 22 wpm after 90 minutes of practice (Figure 6 in [16]).

In summary, previous studies of Dasher are flawed in a number of ways. The studies have been too short, involved too few participants, suffered from sampling bias, lacked a control condition, used an inappropriate language model, or a combination of the above.

In this paper we contribute the results of two studies of Dasher. In both our studies we configure Dasher to use a state-of-the-art long-span language model. We show our language model provides superior predictive performance to Dasher's built-in language model. In the first experiment we compare a standard eye-typing interface with a 1-second dwell-timeout with Dasher in a multisession experiment involving nine participants. In the second experiment we compare an eye-typing interface in which participants can adjust the dwell-timeout with Dasher in a multisession experiment involving 12 different participants. Both studies show that Dasher is significantly faster than eye-typing with no significant difference in error rate.

2. EXTENDING DASHER

Writing in Dasher relies on making character predictions based on the user's previously written text. If the user's desired next character is well-predicted, it will appear in a big box and the user will be able to locate it easily and navigate to it quickly. On the other hand, poorly predicted characters may initially be too small to be seen. The user must decide on the correct area to point at based solely on knowledge of the vertical alphabetical ordering of characters. Even once pointed in the right direction, it takes time to reach the poorly predicted character's small box.

Dasher makes its predictions using the text compression algorithm prediction by partial match (PPM) [1]. Similar to n-gram language models, PPM makes predictions based on counting how often certain characters appear in some training text after different prior contexts. In PPM terminology, the model *order* refers to how many prior characters are used to predict the next character. An order-1 model uses a single previous character of context, e.g. $P(e|h)$, an order-2 model uses two previous characters of context, e.g. $P(e|th)$, and so on. PPM blends the probability estimates from different model orders to obtain robust predictions. Dasher uses the PPM-D blending strategy and by default uses models up to order-5.

When Dasher starts, its PPM model is trained from scratch using a file containing a collection of text. The exact file depends on the user's configured language and alphabet. For example, for British English with an alphabet of the lowercase letters, A–Z, and limited punctuation, the file `training_englishLC_GB.txt` is used to train PPM. This training file contains 55K words and 317K characters. Additionally, Dasher may train on any past Dasher writing by the user.

During usage, Dasher adapts its PPM model, allowing the model to better predict the user's writing style, vocabulary, etc. This is accomplished by incrementing counts in the PPM trie data structure every time the user writes a new character. The changes from this adaptation can be kept between Dasher writing sessions by appending the user's writing to the training text Dasher loads at startup.

2.1 Improved Language Model for Dasher

We found the existing Dasher language modeling infrastructure was insufficient to provide state-of-the-art language modeling performance. To achieve the best predictions, we anticipated the need to condition on more than five characters of context. However, to estimate longer-span language models requires substantially more training data than 300K characters. We wanted to train on billions of characters, something that would be too time consuming to do every time Dasher starts up. Additionally, Dasher's current PPM implementation lacks any model pruning - attempting to train on large amounts of data would result in running out of memory.

To address these problems, we modified Dasher to load an n-gram language model pre-trained using the SRILM toolkit [11]. This toolkit supports training on very large amounts of data, and allows pruning of models to help control memory requirements.

We created our character-based language model using data collected from the microblogging site, Twitter. Prior research has demonstrated that Twitter data models Augmentative and Alternative Communication (AAC) text much better than other publicly available large data sources, such as Wikipedia or newswire text [14]. We sampled a small percentage of worldwide tweets from December 2010 to June 2012 using Twitter's free streaming API.

We first eliminated any tweets marked as retweets and any tweets not identified as English using a language identification module [6]. Additionally, we only kept tweets with a source attribute identified as having been written from a mobile client (e.g. "Twitter for iPhone", "Twitter for Android", etc). We also eliminated any repeated tweets from the same Twitter user. After these filtering steps, we had 291M tweets.

We split each tweet into one or more sentences. We only kept sentences where all words (after removing punctuation such as commas) were in a list of 330K English words. We obtained our word list by concatenating a number of human-edited dictionaries (Wiktionary, Webster's dictionary, CMU pronouncing dictionary, and GNU aspell). Our final training set consisted of 94M sentences (3.1B characters).

We report the predictive power of our language models using the per-character *perplexity*. Perplexity is the degree of uncertainty the model has about the next prediction given the previous context; lower perplexity is better. For example, the per-character perplexity of a completely random sequence of numeric digits would be 10.

Our language model used a vocabulary of the letters A–Z plus the following symbols: space, apostrophe, comma, period, exclamation mark, and question mark. We tested a variety of model orders on a development set and found the majority of perplexity gains were realized by a 10-gram model (i.e. the probability of the next character depended on the previous nine characters).

We trained our model with SRILM using Witten-Bell smoothing and no count cutoffs. Our resulting 10-gram model had a compressed disk size of 1.6 GB. To reduce the model's size, we entropy-pruned our model. This resulted in a compressed disk size of 39 MB.

2.2 Language Model Comparisons

Our pre-trained language model allowed us to condition on a much longer context (nine characters instead of five) and to train on much more data (3.1B characters instead of 300K). However, we also did give up the adaptive feature present in Dasher's original PPM implementation. We wanted to investigate how our static language model performed compared to Dasher's default PPM language model as well as PPM without adaptation. To measure this, we used each model to predict the characters in a sequence of 1347 sentences drawn from the Mobile Enron data set [15].

Figure 2 shows the average per-character perplexity of adaptive PPM, non-adaptive PPM, and our MobileTwitter language model. The curves show the moving average of the last 100 sentences out of the sequence of 1347 sentences. As can be seen in Figure 2, our MobileTwitter model outperformed the non-adaptive PPM model for the entire sequence. It also outperformed the adaptive PPM model for the first 900 sentences. After this, adaptive PPM and our model performed similarly. Overall, the perplexity on the 1347 sentences was 5.6 for non-adaptive PPM, 4.6 for adaptive PPM, and 4.2 for our pruned MobileTwitter model. For comparison, our unpruned 1.6 GB model has a perplexity of 3.8. In our experiments, participants never wrote more than 385 sentences with Dasher.

3. EXPERIMENT 1

We carried out a within-subjects experiment with a single independent variable (Text Entry Method) with two levels (Dasher and Eye-Typing). Participants were presented with memorable sentences and asked to write them using each interface as quickly and as accurately as possible.

3.1 Method

We recruited nine participants from our university campus via convenience sampling. Their ages ranged between 19–27. Six were male and three were female. None of the participants had used an eye-tracking interface before. Participants were compensated with a £30 Amazon voucher for volunteering to participate. All participants studied at the university. Two of the participants studied computer science, the other participants studied history, physics, chemistry, international relations, mathematics or art history. All participants were fluent in English.

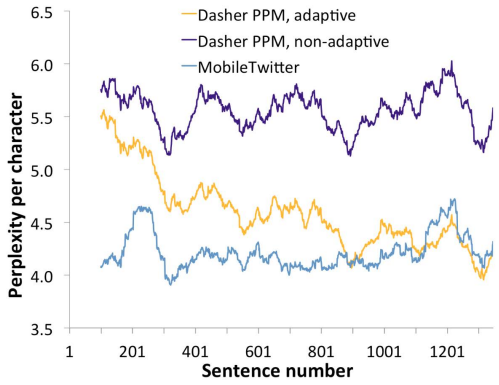

Figure 2: Comparison of the per-character perplexity as we simulated the writing of a sequence of 1347 sentences from the Mobile Enron data set. The curves are a moving average with a window size of 100 sentences.

Figure 3: A participant seated in our laboratory in front of the Tobii P10 eye-tracker machine used in the experiments.

As stimuli we used the Mobile Enron data set [15]. This set consists of sentences from genuine mobile emails that have been validated to be memorable. It has also been shown [4] that this set of sentences results in comparable performances to the earlier and less externally valid phrase set by MacKenzie and Soukoreff [7].

We used a Tobii Technology AB P10 assistive work station with an integrated eye-tracker running Windows XP (see Figure 3). The screen size was 30 × 23 cm and the screen resolution was 1024 × 768 pixels. The eye-tracker had a sampling rate of 40Hz and an accuracy of 0.5°.

Both Dasher and Eye-Typing were configured to run in full-screen mode on the workstation to minimize potential distraction.

Dasher was configured with the language model we described previously. In this first experiment, we also enabled Dasher's automatic speed control setting, which sets the zoom speed as a function of an individual participant's proficiency in using the interface. This is the default setting in Dasher.

As a control condition, we implemented a standard eye-typing keyboard for our Eye-Typing condition. Our dwell-based eye-typing keyboard was modeled after the assistive P10 workstation machine's

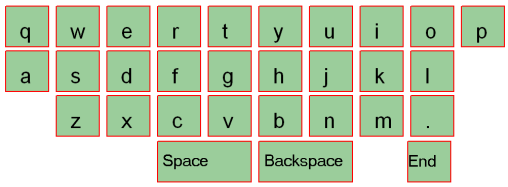

Figure 4: The eye-typing keyboard.

default eye-typing keyboard. It had the exact same keyboard layout (QWERTY) and geometry, in terms of the key sizes, spaces between the keys, and the key positions on the screen. The system in the Eye-Typing condition also behaved identically to the built-in eye-typing keyboard, showing a visual dwell-timeout indication as a graphical clock. The dwell-timeout was set to 1000 ms, the default setting on the assistive P10 workstation machine and a common default setting in eye-typing experiments [10]. Our software was written in C++ and we used the Tobii SDK to interface with the eye-tracker and to implement the fixation detection code.

The keyboard (Figure 4) was 288.6 × 103 mm (985 × 344 pixels). Individual letter keys were 24.9 × 24 mm (85 × 80 pixels). The spaces between the columns were 4.4 mm (15 pixels). The spaces between the rows were 2.4 mm (8 pixels). The width of the space and backspace keys were 49.8 mm (170 pixels).

3.1.1 Procedure

This experiment consisted of one practice session and nine testing sessions for each condition. Each session was separated by a break. Sessions were spread out over five consecutive days.

In the practice session, the Dasher and Eye-Typing interfaces were explained to the participants. To minimize the risk of introducing a novelty bias, participants were not told that eye-typing was an established paradigm. Instead, both the Dasher and Eye-Typing conditions were presented as two different methods researchers were investigating to help users with motor disabilities write using an eye-tracker.

Thereafter, participants calibrated the eye-tracker using the Tobii P10 built-in calibration tool. Calibration was repeated until it was successful and the Tobii P10 eye-tracker was able to accurately track the participant's gaze. Any participant who could not be successfully calibrated at this stage was removed from the study. This happened with three participants, who were compensated for their time.

Participants were then shown how to use Dasher and Eye-Typing. Participants completed a few test sentences in each method to ensure they understood how both methods worked.

Nine testing sessions followed the practice session. In each testing session, participants used Dasher and Eye-Typing to write sentences randomly drawn from the set of test sentences. The order of Dasher and Eye-Typing was balanced across the sessions and the starting condition was balanced across participants insofar as possible. Participants wrote for 10 minutes using one interface before taking a break. After the break, they wrote for 10 minutes using the other interface.

Participants were shown a stimulus sentence in a pop-up dialog box and asked to remember it. The sentence was also spoken to them by playing back pre-recorded sound files. If the participant forgot a sentence during the writing process, they could press the physical SPACE key to replay the audio file. When the participant felt ready they pressed the physical SPACE key to begin writing the sentence. Participants were instructed to write sentences "as quickly and as accurately as possible". To terminate a sentence in Dasher the participant gazed at the centre of the screen and waited for a set dwell-timeout. To terminate a sentence in Eye-Typing, the participant gazed at an END key on the keyboard and waited for a set dwell-timeout.

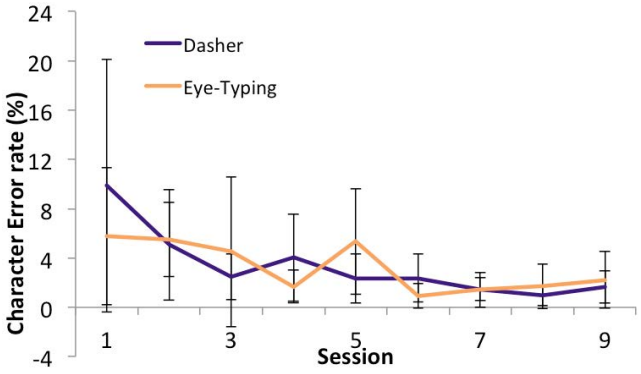

Figure 5: Mean character error rate and 95% confidence interval as a function of session for Experiment 1.

3.2 Results

In total we collected 27 hours of data and participants wrote a total of 2472 sentences. No transformations were applied to our data before statistical significance testing. All statistical significance tests used an initial significance level of $\alpha = 0.05$.

3.2.1 Error Rate

Error rate was measured as Character Error Rate (CER). CER is the minimum edit distance between the stimulus sentence and the user's response, divided by the number of characters in the stimulus sentence.

The character error rate as a function of session is shown in Figure 5. The character error rate in the first session was 9.9% CER for Dasher and 5.8% CER for Eye-Typing. This character error rate was reduced in the last session to 1.7% CER for Dasher and 2.2% CER for Eye-Typing.

Character error rates were analyzed using repeated measures analysis of variance. The difference in character error rate between Dasher and Eye-Typing was not significant ($F_{1,8} = 0.11$, $\eta_p^2 = 0.001$, $p = 0.918$), nor was the difference across Session ($F_{1,8} = 3.516$, $\eta_p^2 = 0.305$, $p = 0.098$). There was also no significant interaction between Input Method and Session ($F_{8,64} = 0.748$, $\eta_p^2 = 0.086$, $p = 0.412$).

3.2.2 Entry Rate

Entry rate was measured in words per minute, with a word defined as five consecutive characters including spaces. The time interval used to compute entry rate was the time interval from the participant pressing a physical SPACE key to the time participants entered the last letter in the stimulus sentence. Timing data was measured using the CPU's high-resolution time stamp counter, which provides microsecond precision.

Entry rate as a function of session is shown in Figure 6. Dasher resulted in approximately twice as fast an entry rate as Eye-Typing. The entry rate in the first session was 6.6 wpm for Dasher and 4.6 wpm for Eye-Typing. Entry rates increased with practice. In the last session the entry rate had increased to 12.4 wpm for Dasher and 6.0 wpm for Eye-Typing.

Entry rates were analyzed using repeated measures analysis of variance. The difference in entry rate between Dasher and Eye-

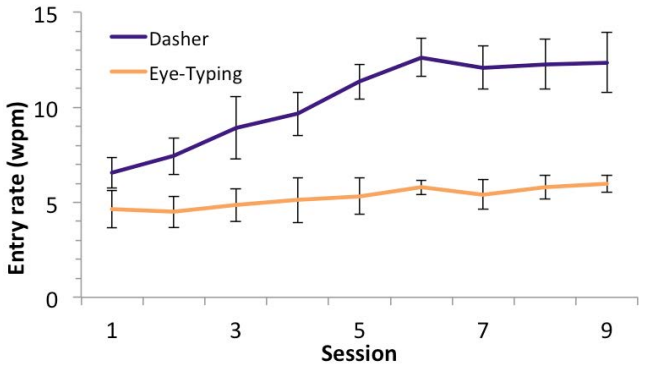

Figure 6: Mean entry rate (wpm) and 95% confidence interval as a function of session for Experiment 1.

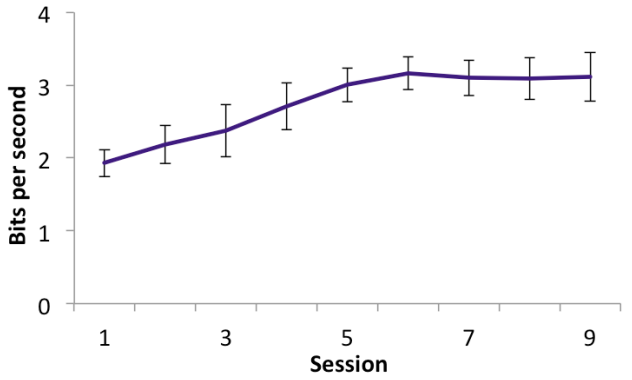

Figure 7: Mean Dasher speed (bps) and 95% confidence interval as a function of session for Experiment 1.

Typing was statistically significant ($F_{1,8} = 129.448$, $\eta_p^2 = 0.942$, $p < 0.001$). We also found a significant main effect for Session ($F_{1,8} = 28.987$, $\eta_p^2 = 0.784$, $p = 0.001$), which means participants improved with practice. There was a significant interaction between Input Method and Session ($F_{8,64} = 13.959$, $\eta_p^2 = 0.636$, $p = 0.006$). Pair-wise Bonferroni-Holm corrected *post-hoc* tests revealed that all pair-wise differences between Dasher and Eye-Typing across sessions were significant, with ($p < 0.003$) for all sessions, except the first ($p = 0.031$).

Recall that Dasher was configured to use automatic speed control. Figure 7 plots the Dasher zoom speed in bits per second (bps) for all participants as a function of session. As is evident in the figure, Dasher's speed in bps strongly correlated with the entry rate shown in Figure 6.

3.2.3 *Perceived Task Load*

Perceived task load was measured using the full version of the NASA-TLX perceived task load index. Since these ratings result in residuals that are non-normal, we performed the statistical significance tests using Friedman's non-parametric test.

We found no statistically significant differences for Temporal Demand, Performance Load or Overall Task Load (Table 1). Participants' perceived physical demand, mental demand, effort and frustration were significantly lower for Dasher.

3.2.4 *Subjective Ratings*

We also asked participants to rate five statements on a 1–7 Likert scale (1 = Strongly Disagree, 7 = Strongly Agree). The results are

Table 1: Significance test results for perceived task loads in the experiment. Significant differences are boldfaced. All tests in the table had a single degree of freedom

	First Session		*Last Session*	
	χ^2	p	χ^2	p
Physical Demand	0.143	0.705	**4.500**	**0.034**
Mental Demand	0.111	0.739	**7.000**	**0.008**
Temporal Demand	1.000	0.317	0.000	1.000
Performance Load	0.000	1.000	0.111	0.739
Effort	1.000	0.317	**4.500**	**0.034**
Frustration	**9.000**	**0.003**	**9.000**	**0.003**
Overall Task Load	1.000	0.317	2.778	0.096

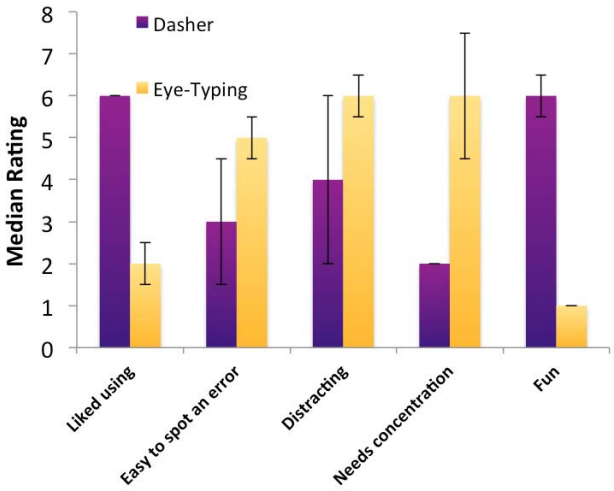

Figure 8: Median subjective ratings and interquartile ranges on a 1–7 Likert scale for Experiment 1.

shown in Figure 8. There was no significant difference in how users rated the ability to easily spot errors ($\chi^2 = 2.778, df = 1, p = 0.096$). There was also no significant difference in how users rated how distracting each interface was. Participants liked using Dasher significantly more than Eye-Typing ($\chi^2 = 9.000, df = 1, p = 0.003$), and found it more fun to use ($\chi^2 = 9.000, df = 1, p = 0.003$).

3.2.5 *Open Comments*

Participants were invited to write down any open comments they had about the experiment. Invariably, there were negative comments about Eye-Typing. One participant wrote that "the physical concentration required to write took a lot of effort". Conversely, participants expressed enjoyment with Dasher, with one participant stating that it "makes the keyboard seem so slow" and more than one participant stating it was "like a game".

4. EXPERIMENT 2

In Experiment 1, the ability to adjust the Eye-Typing dwell-timeout was not implemented. In this experiment we wanted to investigate whether enabling participants to adjust the dwell-timeout would substantially increase the entry rate for Eye-Typing. We also wanted to test more participants and therefore recruited 12 participants instead of nine as in the previous experiment. To enable participants to fully learn both interfaces, we extended session times from 10 to 15 minutes. Further, to avoid potentially confusing par-

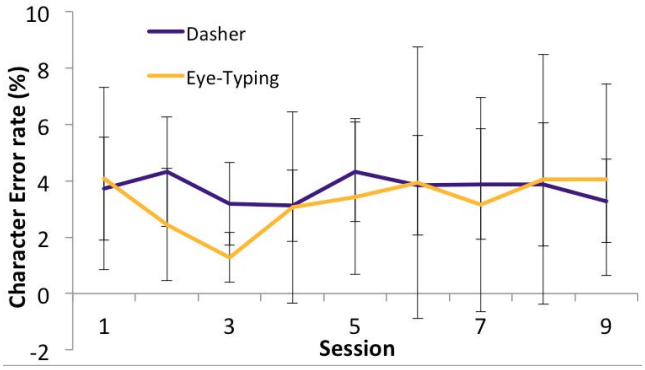

Figure 9: Mean character error rate (%) and 95% confidence interval as a function of session for Experiment 2.

ticipants, in Experiment 2 participants first completed all their testing sessions with one interface before moving onto the other interface. The starting condition was balanced across participants.

4.1 Method

For this experiment, we recruited 12 participants from our university campus via convenience sampling. 11 of the participants' ages ranged between 18-27, with one participant aged 59. Eight were male and four were female. Three of the participants had used an eye-tracker before, but none had used it for text entry. Participants were compensated with a £40 Amazon voucher for volunteering. All the participants studied at the university. Nine of the participants studied computer science, the other participants either studied either history or physics. The material was the same set as used in Experiment 1.

The apparatus was identical to Experiment 1. In this experiment, the automatic speed control on Dasher was disabled. Instead, a spinner widget was placed in a toolbar at the bottom of the screen, allowing participants to manually adjust their zoom speed. In the Eye-Typing condition, the spinner allowed participants to adjust the dwell-timeout.

4.1.1 Procedure

The experiment consisted of one practice session and nine testing sessions for each condition. Each testing session lasted 15 minutes, with breaks in between. Participants came for eight one-hour periods, where they would complete up to three sessions. These periods were spread out over a minimum of four days, and a maximum of 14 days. During all testing sessions, participants were allowed to adjust the speed of the Dasher interface zoom. For the Eye-Typing testing sessions, participants were allowed to adjust the dwell-timeout after the fifth session.

4.2 Results

We collected 54 hours of data and participants wrote a total of 5067 sentences. Statistical analyses were performed identically to Experiment 1.

4.2.1 Error Rate

The character error rate as a function of session is shown in Figure 9. The character error rate in the first session was 3.7% CER for Dasher and 4.1% CER for Eye-Typing. This character error rate was reduced slightly in the last session to 3.3% CER for Dasher and 4.0% CER for Eye-Typing. The lowest character error rate of all sessions was 3.1% for Dasher and 1.3% for Eye-Typing.

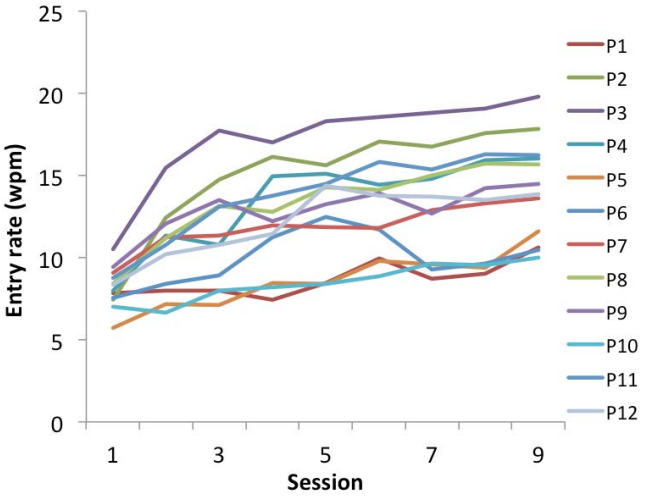

Figure 10: Dasher entry rate for all participants as a function of session for Experiment 2.

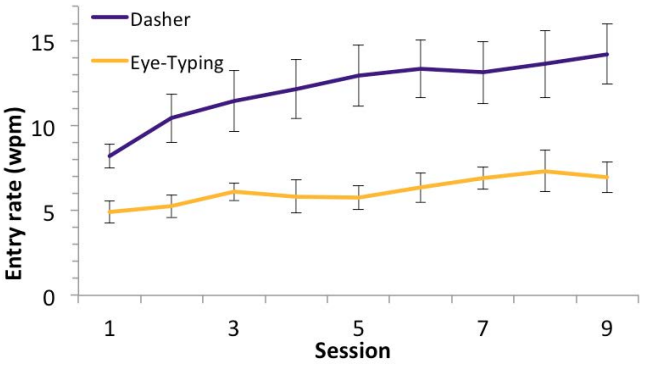

Figure 11: Mean entry rate (wpm) and 95% confidence interval as a function of session for Experiment 2.

Character error rate was analyzed using repeated measures analysis of variance. The difference in character error rate between Dasher and Eye-Typing was not significant ($F_{1,11} = 0.153$, $\eta_p^2 = 0.014$, $p = 0.703$). The difference between sessions was also not significant ($F_{1,8} = 1.229$, $\eta_p^2 = 0.101$, $p = 0.291$). Finally, there was no significant interaction between input method and session ($F_{8,88} = 0.905$, $\eta_p^2 = 0.076$, $p = 0.362$).

4.2.2 Entry Rate

Entry rate was measured identically to Experiment 1, and is shown as a function of session in Figure 11. Again, Dasher resulted in approximately twice the entry rate compared to Eye-Typing by the end of all testing sessions. The entry rate in the first session was 8.2 wpm for Dasher and 4.9 wpm for Eye-Typing. In the last session, the entry rate had increased to 14.2 wpm for Dasher and 7.0 wpm for Eye-Typing.

As in Experiment 1, the average Dasher zoom speed in bps per session is plotted in Figure 12. There is a strong similarity between this and the entry rate means shown in Figure 11.

We saw no direct correlation between these variables. Whereas before, the zoom speed would generally increase as the participant's entry rate increased, the manual speed adjustment meant that these variables were independent of one another.

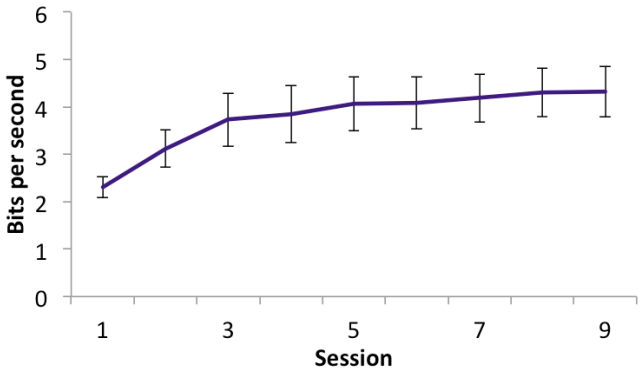

Figure 12: Mean Dasher speed (bps) and 95% confidence interval as a function of session for Experiment 2.

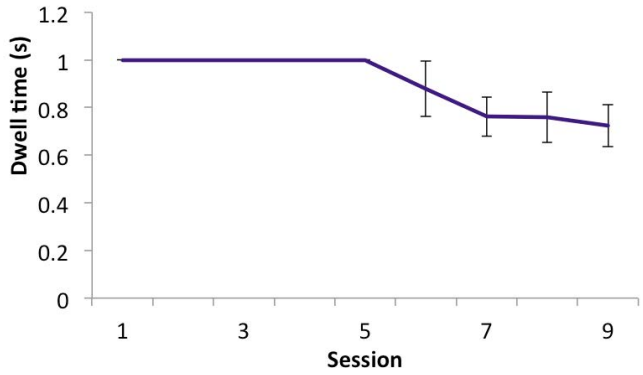

Figure 13: Mean Eye-Typing dwell-timeout (s) and 95% confidence interval as a function of session for Experiment 2.

Figure 13 shows the mean Eye-Typing dwell-timeout in seconds for each session. This shows how participants adjusted the dwell-timeout in the last four sessions when they were allowed to do so. Despite this, the entry rate of the Eye-Typing condition remained similar throughout all sessions, suggesting that the ability to adjust the dwell-timeout did not have an effect on participants' entry rates. It should also be noted that the curve in this figure flattens out by the final session, demonstrating that participants did not choose to go any faster.

Entry rate was analyzed using repeated measures analysis of variance. The difference in entry rate between Dasher and Eye-Typing was statistically significant ($F_{1,11} = 71.219$, $\eta_p^2 = 0.866$, $p < 0.001$). We found a significant main effect for Session ($F_{1,8} = 40.27$, $\eta_p^2 = 0.785$, $p < 0.001$), which means participants improved with practice. There was also a statistically significant interaction between Input Method and Session ($F_{8,88} = 7.397$, $\eta_p^2 = 0.402$, $p = 0.02$). Pair-wise Bonferroni-Holm corrected *post-hoc* tests revealed that all pair-wise differences between Dasher and Eye-Typing across sessions were significant ($p < 0.001$).

4.2.3 *Subjective Ratings*

As in Experiment 1, we asked participants to rate the same five statements on the 1–7 Likert Scale. In this experiment, we included a question on how much they liked being able to manually adjust the speed of each interface, which was only relevant in this experiment. The results are shown in Figure 14. Contrary to the previous experiment, there was no significant difference in how participants

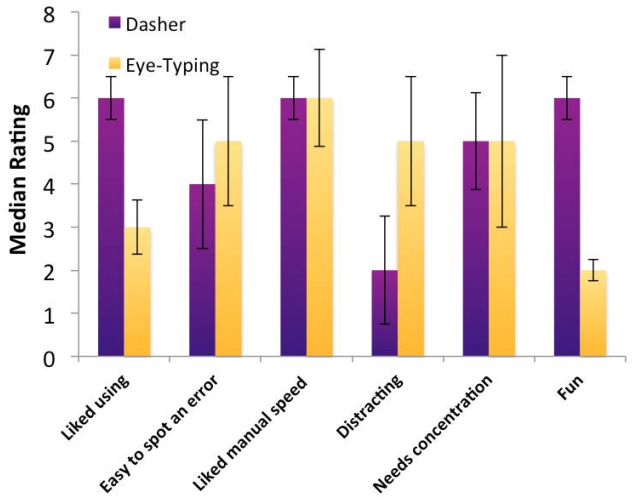

Figure 14: Median subjective ratings and interquartile ranges on a 1–7 Likert scale for Experiment 2.

rated the intensity of concentration required ($\chi^2 = 0.091, df = 1, p = 0.763$). Despite the difference in medians, there was also no significant difference in how distracted participants were in checking what they had written ($\chi^2 = 0.818, df = 1, p = 0.366$).

4.2.4 *Open Comments*

Again, there was a clear bias towards Dasher in participants' feedback. In terms of usability, one said "I found myself straining when using the onscreen keyboard", which was mirrored by another participant, who said "my eyes were a lot more strained and sore with the keyboard". However, the same participant found Dasher "quite relaxing and enjoyable to use". Negative comments towards Dasher were related to participants finding it difficult to correct errors. One participant said "getting back a level once you had entered a letter was difficult". Another participant said "it was not always easy to find where to go 'back' to".

5. DISCUSSION

It must be noted that it is only possible to make reliable entry and error rate comparisons between techniques within the same experiment. Different experimental setups sample different participants, use different apparatus and stimuli, and use slightly different procedures. Furthermore, some previous evaluations of eye-typing and Dasher have been conducted with Finnish participants writing Finnish text (e.g. [8, 9, 12]). The structure of the Finnish language is very different from English and a previous research paper has reported that Finnish participants' English text entry was 16% slower than Finnish [3]. This confound is exacerbated by the fact that several prior studies [8, 12] did not use a control condition.

Our eye-typing entry rate result of 7 wpm is comparable to those in previous work, which are mostly within the range of 6-12 wpm. Wobbrock et al. [20] found an average entry rate of 7 wpm with an uncorrected error rate of 4.62%, highly similar to our results. Majaranta et al. [9] conducted three eye-typing experiments, two with a long dwell-timeout of 900 ms and one with a shorter dwell-timeout of 450 ms. The first two studies had a grand mean entry rate of 7 wpm. The final study with the short dwell-timeout had a grand mean entry rate of 10 wpm and an error rate of 1.2%, although this study had recruited experienced participants who had been exposed to either the first or second study. Hansen et al. [2]

reported a 6 wpm mean using a hierarchical keyboard. Špakov et al. [17] achieved a higher average entry rate of 12 wpm with a 2.3% error rate and an automatic dwell-timeout adjustment algorithm. However, their participants were experienced with eye-typing. In our Experiment 2, only two participants attempted to reduce their dwell-timeout below 500 ms.

Tuisku et al. [12] found that after 10 sessions of 15 minutes, participants were able to write at an average of 17 wpm. Their error rate of 4.04% is comparable to our character error rate of 3.3%. The most likely reason for this difference is that the participants in this study wrote in Finnish, using a language model trained on a novel. The structure of the Finnish language is entirely different to English, so it is unclear whether these results generalize. It is impossible to tell whether the differences in language and training text led to an increase or decrease in entry rate. Since Tuisku et al. [12] did not use a control condition it is also unclear whether these participants would also have higher entry rates in an eye-typing condition. Vertanen and MacKay [16] used Dasher as a baseline condition in their experiment. Two of the participants reached 17 wpm and the inventor of Dasher reached 22 wpm (Figure 6 in [16]). In our study we also found that some participants could reach entry rates in the range 17–20 wpm (cf. Figure 10). However, as we sampled a broader set of users, Figure 10 also makes it clear that some users are not as fast.

The first paper on using Dasher for eye-tracking [19] claims that users could write up to 25 wpm within an hour of practice. However, this result refers to a single expert user of the interface. Although not stated in the results, Figure 1 in Ward and MacKay [19] reveals that the other two participants in their study reached an entry rate between 10-15 wpm after an hour of practice. As the average entry rate of our participants after an equivalent time was 12 wpm, the entry rates obtained in our study are in line with the non-expert users in the study by Ward and MacKay [19].

While allowing participants to manually adjust their speed resulted in higher entry rates for both interfaces, it raised the issue of participants' awareness of their control of the interface. One participant never increased the speed beyond a tentative 2.65 bps, despite consistently low character error rates. Conversely, another participant increased the speed to beyond 6 bps and was only able to reach a below-average entry rate. Generally, however, participants were able to find their "sweet spot" by the end of the sessions and would produce steady entry rates.

6. CONCLUSIONS

In this paper we have made the following contributions. First, we have presented our method of modifying Dasher to use a state-of-the-art long-span language model trained on appropriate training texts for eye-typing. Second, we have compared Dasher's performance against a standard eye-typing interface and an eye-typing interface using adjustable dwell-timeouts. Both experiments showed that Dasher had a significantly higher entry rate than eye-typing (14 wpm versus 7 wpm). We also found that character error rates were not significantly different to that of eye-typing.

We analyzed rating-based feedback provided by our participants, and found that Dasher was significantly more likeable and less frustrating to use than eye-typing. These results were also reflected in the participants' open comments. Based on the results from these two experiments, we recommend eye-tracking users adopt Dasher for their text entry needs.

7. REFERENCES

[1] Bell, T. C., Cleary, J. G., and Witten, I. H. *Text Compression*. Prentice Hall, Englewood, New Jersey, USA, 1990.

[2] Hansen, J. P., Tørning, K., Johansen, A. S., Itoh, K., and Aoki, H. Gaze typing compared with input by head and hand. In *Proc. ETRA 2004*, ACM Press (2004), 131–138.

[3] Isokoski, P., and Linden, T. Effect of foreign language on text transcription performance: Finns writing english. In *Proc. NordiCHI 2004*, NordiCHI '04, ACM (New York, NY, USA, 2004), 109–112.

[4] Kristensson, P. O., and Vertanen, K. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proc. IUI 2012*, ACM Press (2012), 29–32.

[5] Kristensson, P. O., and Vertanen, K. The potential of dwell-free eye-typing for fast assistive gaze communication. In *Proc. ETRA 2012*, ACM Press (2012), 241–244.

[6] Lui, M., and Baldwin, T. Cross-domain feature selection for language identification. In *Proc. IJCNLP 2011* (2011), 553–561.

[7] MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *Extended Abstracts of the 21st ACM Conference on Human Factors in Computing Systems*, ACM Press (2003), 754–755.

[8] Majaranta, P., Ahola, U.-K., and Špakov, O. Fast gaze typing with an adjustable dwell time. In *Proc. CHI 2009*, ACM Press (2009), 357–360.

[9] Majaranta, P., MacKenzie, S., Aula, A., and Räihä, K.-J. Effects of feedback and dwell time on eye typing speed and accuracy. *Universal Access in the Information Society 5*, 2 (2006), 199–208.

[10] Majaranta, P., and Räihä, K.-J. Twenty years of eye typing: systems and design issues. In *Proc. ETRA 2002*, ACM Press (2002), 15–22.

[11] Stolcke, A. SRILM – an extensible language modeling toolkit. In *Proc. ICSLP 2002*, ISCA (2002), 901–904.

[12] Tuisku, O., Majaranta, P., Isokoski, P., and Räihä, K.-J. Now dasher! dash away!: longitudinal study of fast text entry by eye gaze. In *Proc. ETRA 2008*, ACM Press (2008), 19–26.

[13] Urbina, M. H., and Huckauf, A. Alternatives to single character entry and dwell time selection on eye typing. In *Proc. ETRA 2010*, ACM Press (2010), 315–322.

[14] Vertanen, K., and Kristensson, P. O. The imagination of crowds: conversational AAC language modeling using crowdsourcing and large data sources. In *Proc. EMNLP 2011*, ACL (2011), 700–711.

[15] Vertanen, K., and Kristensson, P. O. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proc. MobileHCI 2011*, ACM Press (2011), 295–298.

[16] Vertanen, K., and MacKay, D. J. Speech Dasher: fast writing using speech and gaze. In *Proc. CHI 2010*, ACM Press (2010), 595–598.

[17] Špakov, O., and Miniotas, D. On-line adjustment of dwell time for target selection by gaze. In *Proc. NordiCHI 2004*, ACM Press (2004), 203–206.

[18] Ward, D. J., Blackwell, A. F., and MacKay, D. J. C. Dasher - a data entry interface using continuous gestures and language models. In *Proc. UIST 2000*, ACM Press (2000), 129–137.

[19] Ward, D. J., and MacKay, D. J. C. Fast hands-free writing by gaze direction. *Nature 418*, 6900 (2002), 838.

[20] Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., and Duchowski, A. T. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proc. ETRA 2008*, ACM Press (2008), 11–18.