Parakeet: A Continuous Speech Recognition System for Mobile Touch-Screen Devices

Keith Vertanen and Per Ola Kristensson
Cavendish Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge, UK
{kv277, pok21}@cam.ac.uk

ABSTRACT
We present Parakeet, a system for continuous speech recognition on mobile touch-screen devices. The design of Parakeet was guided by computational experiments and validated by a user study. Participants had an average text entry rate of 18 words-per-minute (WPM) while seated indoors and 13 WPM while walking outdoors. In an expert pilot study, we found that speech recognition has the potential to be a highly competitive mobile text entry method, particularly in an actual mobile setting where users are walking around while entering text.

Author Keywords
Continuous speech recognition, mobile text entry, text input, touch-screen interface, error correction, speech input, word confusion network, predictive keyboard

ACM Classification Keywords
H5.2. User Interfaces: Voice I/O

INTRODUCTION

The advantages of speech recognition as a mobile text entry method seem obvious and highly attractive. Speaking is a naturally acquired skill which does not demand much practice from users. Speech can also be very fast. Users can speak up to 200 WPM [22].

However, speech recognition remains to prove itself as a competitive mobile text entry method. At least in the past, speech recognition performance suffered from poor accuracy [12]. The privacy implications of using speech as the sole modality for text entry are also obvious. Further, it has been argued that using speech as a text entry method also carries cognitive costs that may limit speech recognition entry rates in practice [23].

IUI'09, February 8–11, 2009, Sanibel Island, Florida, USA.

Figure 1. The Nokia N800 device and a Bluetooth headset. The Parakeet continuous speech recognition system is shown running on the device.

From an engineering perspective, speech recognition demands a device with more memory and processing power than required by traditional mobile text entry methods, such as predictive text or character recognition.

Despite these challenges, there has been recent interest in creating continuous speech recognition engines for embedded and mobile devices [4]. Two projects in this direction are PocketSphinx [11] and PocketSUMMIT [10].

These developments pave the way for the study of speech recognition as a mobile text entry method. Two important research questions arise. First, what constitutes a good design for a speech recognition system on a modern mobile device? Second, how efficient and practical is speech recognition as a mobile text entry method?

In this paper, we describe Parakeet: a system designed for efficient mobile text entry using speech recognition. We have built a complete mobile speech recognition system that recognizes American and British English. Users enter text into their mobile Linux device (such as the Nokia N800) by speaking into a Bluetooth headset microphone (Figure 1).

The design and development of Parakeet followed several design cycles. Our design was to a large extent guided by computational experiments on recorded speech audio. These experiments helped us from two perspectives. First,

they helped us visualize the tradeoffs between different design choices, and thereby guided our creative design process. Second, they aided us in finding the optimal parameter settings for our graphical user interface.

This work provides insight into the design issues related to speech-based mobile text entry. We also present the first study of speech-based text entry on an *actual* mobile device in an *actual* mobile setting.

The rest of this paper is structured as follows. First, we discuss the principles which guided our design. Second, we describe our interface and detail the experiments which helped shape the design. Third, we describe the details of our mobile speech recognition system. Fourth, we present a user study validating our design. Fifth, we present results from an expert pilot study demonstrating the promising potential our system has as a mobile text entry solution. Finally, we discuss limitations and implications of our findings, point to future work, and conclude.

DESIGN PRINCIPLES

Avoid Cascading Errors

Speech recognition is imperfect and recognition errors are ultimately unavoidable. Thus, error correction is a crucial part of any speech recognition interface. As previous research has shown (e.g. [12]), users have difficultly correcting recognition errors using only speech. This is partly because errors may cascade – recognition in the correction phase may fail, requiring further corrections, and so on [12]. A possible solution is to use a different modality than speech for correction. For example, Suhm et al. [25] investigated a speech system which enabled corrections via pen gestures. For an in-depth review of multimodal speech interfaces, we refer the reader to Oviatt et al. [20].

To avoid cascading errors, we decided to create a multimodal speech interface. In our system, recognition is first performed on a user's utterance. The recognition result is then corrected in a manner which is simple, direct and transparent to the user. The correction process does not rely on any further error-prone recognition technologies.

Exploit the Speech Recognition Hypothesis Space

In a typical speech recognition interface, such as Dragon NaturallySpeaking, only the best recognition hypothesis is initially shown to the user. In order to explore alternatives to this best hypothesis, the user must take explicit action such as highlighting some text and issuing a command. As observed by Kristensson and Zhai [16], this style of error correction interface introduces a degree of uncertainty. The user has to hope his or her action will expose the desired correction. If it does not, the user has wasted effort. Such interfaces may lead to user frustration, which is a common problem in intelligent user interfaces in general.

Therefore, we designed our correction interface so it avoids forcing users to blindly search the hypothesis space. We wanted the user to be able to see immediately, and without explicit action, whether the desired correction was available. If the desired correction was not easily available, the user could then undertake a more costly corrective measure (but a measure guaranteed to be successful).

Efficient and Practical Interaction by Touch

Mobile devices increasingly come with stylus or touch sensitive screens. These modalities enable us to implement a full direct manipulation user interface. Direct manipulation enables us to design a user interface where users can view and act on the rich set of recognition hypotheses we wanted to display.

We preferred touch-interaction to stylus for three reasons. First, touch doesn't require the user to utilize a stylus while on the go. Second, one-handed usage is impossible with a stylus, and users tend to prefer mobile interfaces that can be operated with one hand [13]. Third, a well-designed touch-screen interface can also be used with a stylus, but the converse does not necessarily hold. By creating a good touch-screen design, we are also creating an interface suitable for stylus use.

Support Fragmented Interaction

When users are entering text while walking around, they need to divide their attention between interacting with the mobile device and dealing with their surroundings. Oulasvirta et al. [19] found that users interacting with a mobile web browser while walking in a demanding setting attended to the device in 4-8 second bursts. This finding has two implications. First, our system needs to enable users to quickly process and respond to the interface. Second, our system needs to be designed so that it is easy for users to pick up from where they left off after an interruption.

We therefore designed Parakeet to minimize attention demands. For example, after recognition is completed, we flash the entire display screen and beep. This simple feedback frees the user to attend to their surrounding almost entirely while waiting for recognition to complete.

We also wanted to minimize the physical actions required. In a mobile setting, a large sequence of precise touch actions is likely to go wrong. We designed towards an interface presenting more visual information (perhaps requiring several bursts of attention) but requiring fewer motor actions. For example, the user might enter a few letters of a word and then scan a list of predictions which allow completion of the entire word with a single touch.

INTERFACE DESCRIPTION

The main part of our correction interface displays the recognizer's best hypothesis along a single line at the top (Figure 2). If the best hypothesis cannot fit on the screen, the user can scroll using the buttons on the left and right sides. In addition to the best hypothesis, likely alternatives for each word in the best hypothesis are also displayed. For example, in Figure 2 the word "imports" has two other competing words ("imported" and "import").

This display is based on a word confusion network [9]. A word confusion network is a time-ordered set of clusters where each cluster contains competing word hypotheses along with their posterior probabilities. The word confusion network is built from the lattice generated during the speech recognizer's search.

Ogata and Goto [18] also used a confusion network as a basis for a speech correction interface. In relation to their work, our interface incorporates several novel aspects:

- **Word candidate order -** Ogata and Goto [18] ordered all word candidates (including delete buttons) strictly by probability. We changed this so all delete buttons were in the bottom row. This was done for consistency and also to allow contiguous errors to be deleted in a single swipe.
- **Copying -** We allow copying of words between clusters. This feature is described in detail later.
- **Keyboard fallback -** We provide a fallback correction mechanism based on a predictive software keyboard.
- **Mobile design -** Our interface is designed to work on a small mobile device using touch-interaction, rather than on a desktop using a mouse. This requires careful attention to the size of user interface elements and the addition of a scrolling feature.

Words in each cluster are displayed in order of their probability. While we could have adjusted button color or size based on these probabilities, our past work on confidence visualization suggests such a feature was unlikely to benefit users [28].

By displaying more than the 1-best result, our interface allows the user to quickly scan other words that the recognizer thought were likely. This is done without requiring the user to explicitly ask for alternatives as in status quo interfaces, such as Dragon NaturallySpeaking.

Available Actions in the Word Confusion Network

Substituting Words

To substitute a word, the user can use several methods. The most direct method is to simply touch an alternate word in the confusion network. This causes the selected word to change color and updates the word displayed in the top row. Sometimes several desired substitutions are in adjacent columns. In this case, the user can cross each desired word to perform multiple substitutions with one gesture.

Editing Words

The user's desired word may not be one of the displayed alternatives. By touching a word in the top row, or by double-tapping any word, the user is brought to a separate edit screen. Here they can use a software keyboard to either edit the selected word or enter an entirely new word.

Deleting Words

To delete words, the user touches the delete button (a box with a diagonal *X*, cf. Figure 2). If the user wants to delete several words at once, the user can slide across adjacent delete buttons. In speech recognition, often the recognizer gets off track and outputs several erroneous words in a row. In such instances, it is particularly useful to delete several words at once by crossing contiguous delete buttons. To easily allow such contiguous deletes, we aligned all delete buttons at the bottom.

Inserting Words

To insert a word, the user can touch the area between two columns. This brings up a keyboard interface which allows the user to choose from a set of word candidates or write a new word (Figure 7) A second option is to touch a preceding word (thereby opening the keyboard interface) and type a space and then the new word.

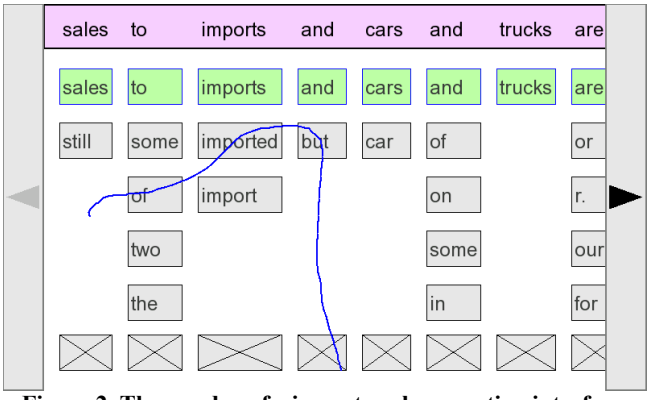

Figure 2. The word confusion network correction interface. The user is crossing several words and delete in one action.

Sometimes the user's desired word may appear in a different column from where the word is required. A third way to insert a word is to copy a word from another column. After touching a word for a moment, the background of the word changes and the word can be copied to another cluster (Figure 3). During the copy, the current target cluster is highlighted in yellow in the top row of the display.

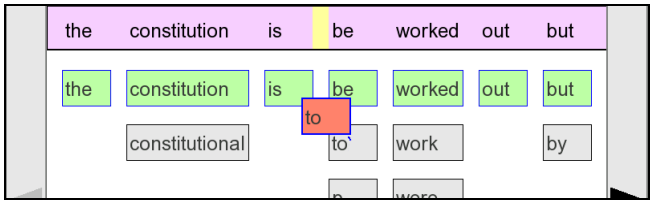

Figure 3. The user is copying the word "to" by dragging it from its original column to the desired location.

Correcting by Crossing

Similar to Kurihara et al. [14], we allowed users to correct errors by making continuous crossing gestures. Figure 2 shows one fluid gesture changing "to" to "of", "imports" to "imported", and finally deleting "and". This crossing-based interaction is possible because in each column, only one item at a time can be selected. Therefore, the detection algorithm only needs to keep track of the last word or delete button crossed in each column. For example, in Figure 2 the

user has crossed "but" and the delete button in column 4. In this case, the detection algorithm will select the delete button as it was the last thing crossed. Users can start crossing anywhere in the display and in any direction.

The theoretical performance of crossing interfaces is of the same mathematical form as Fitts' law [1,7]. At the same index of difficulty [7], crossing is more efficient or on par with pointing performance [1]. It is also likely that fluid crossing actions for corrections can be cognitively "chunked" into more meaningful actions [3]. Users may prefer to think about corrections of an *utterance* rather than corrections of a (possibly unrelated) *set of words*.

Finding Useful Actions

We used computational experiments to guide our decisions on what interface actions to include. We also used the experiments to decide how many word alternatives to use in each cluster. The experiments were done by generating word confusion networks for three standard acoustic test sets (WSJ1 si_dt_05, WSJ0 si_et_05, WSJ0 si_dt_05, 1253 utterances). We assumed an "oracle" user. The oracle made optimal use of a given confusion network and set of interface actions to correct as many errors as possible.

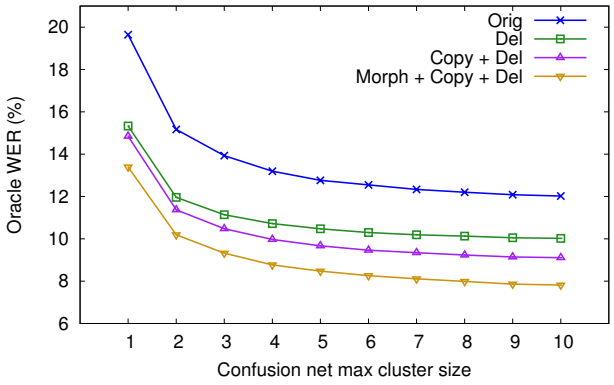

Figure 4. The errors remaining (oracle WER) after correction using a confusion net with a given cluster size. The top line is the performance of the original confusion net. Other lines show performance as correction features were added.

As shown in Figure 4, increasing the cluster size allowed more errors to be corrected. The majority of gains were observed by adding the first few alternatives. This guided our decision to use a small cluster size of five. By minimizing the cluster size, we were able to use larger buttons that are easier to touch. Adding a delete button to every cluster was shown to substantially reduce errors ("Del" line, Figure 4). This makes sense as every insertion error made by the recognizer can be corrected.

We tested allowing copying words from clusters within two words of each other ("Copy+Del" line, Figure 4). This provided a small gain. Bigger gains were possible when we allowed copying across longer distances, but we anticipated users would be unlikely to copy words over long distances.

Finally, we tested allowing words to be easily replaced with one of their morphological variants (e.g. replacing "import" with "imported" or "imports"). This provided further error reductions ("Morph+Copy+Del" line, Figure 4).

Word Substitution Prediction

When a user double-taps a word in the confusion network, the keyboard interface opens. In order to try and minimize the need to type a word, we decided to try and predict likely alternative words based on the word the user touched. For example, if the user touched the word "constitutional", the interface might propose morphological variants of the word, such as: "constitute", "constitutes", etc. (Figure 5).

Before we settled on displaying morphological variants, we considered several alternatives. One alternative is to predict words that are acoustically close. Another alternative is to propose word candidates based on the preceding word (using a forward language model), or based on the following word (using a backward language model). For details on how we generated these alternatives, see [26].

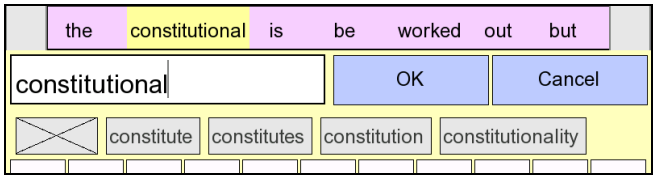

Figure 5. The user touched the word "constitutional" in the confusion net. The morphological variants for "constitutional" are shown in a prediction row above the software keyboard.

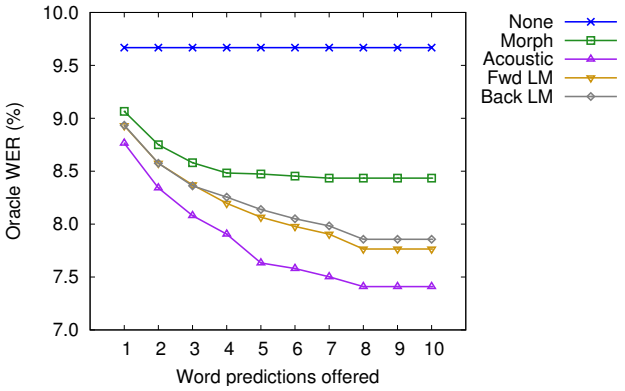

Figure 6. The errors remaining (oracle WER) after correction using an interface offering a given number of predictions. The top line is the baseline (no predictions). Other lines show performance of different types of predictions.

We again simulated a perfect "oracle" user on our set of test word confusion networks. As shown in Figure 6, all forms of predictions reduced oracle word error. Providing more predictions further reduced error, but the majority of the gains were observed by five predictions. This guided our decision to use a single row of word predictions.

While acoustically based predictions performed the best, they are also highly unintuitive to users. As an example, "aback" is acoustically similar to "attack". It would be

difficult for a user to know what word predictions would be offered when touching a word. Language model predictions also suffer from being unintuitive. They depend on surrounding context rather than the actual word the user wants to change. For these reasons, we decided to use morphological variants. It is straightforward to explain to users that if they want a variant of a word differing in say ending, possessiveness, or grammatical number, they should touch the word and check the prediction row.

PREDICTIVE SOFTWARE KEYBOARD

Sometimes a user's desired word does not appear anywhere in the interface. To allow entry of arbitrary words, we added a predictive software keyboard.

Software Keyboard Design

Previous research suggests that pointing performance is severely degraded when users are walking [5]. We therefore tried to make the keys as big as possible (Figure 7).

Another explicit design decision was to make the keyboard literal – each key press is output immediately (as with an ordinary desktop keyboard). There are some proposed keyboards that can improve typing precision by inferring the intended letter keys rather than making a literal interpretation, e.g. [8,15]. However, these solutions are based on machine learning algorithms and could introduce further recognition errors should their inference be wrong. Since we want to avoid cascading errors, we opted for a traditional keyboard to provide fail-safe text entry.

Typing Prediction

We complemented the keyboard with typing prediction [6]. The keyboard suggests likely words based on what the user has typed so far (Figure 7). It finds predictions by searching a prefix-tree of 64K words. The prediction display is populated with the most likely words (given a unigram language model) which match the prefix. Matching predictions are displayed in alphabetical order. As previously described, when users first open the keyboard, the prediction row is populated with morphological word variants. These are replaced with prefix-based predictions as soon as the user begins typing.

Typing prediction results were displayed on the screen 350 ms after the last key press. This delay was introduced for two reasons. First, the prediction lookup and screen redraw introduces lag which could interfere with users' typing. Second, a dynamically changing graphical interface might distract users from their typing task.

MOBILE SPEECH RECOGNIZER

Our speech recognizer was based on CMU Sphinx, using the PocketSphinx [11] decoder. In this section, we give details of the recognition-related components of our system.

Audio Capture

We captured audio on the N800 using a Blue Parrot B150 Bluetooth headset. We chose this headset as it has a close-talking boom microphone. Audio sampled at 8 kHz was obtained using the GStreamer framework. Audio was streamed to the recognizer as soon as the microphone was enabled. This allowed decoding to begin even before the user had finished speaking. As the user's entire audio was not available at the start of recognition, we used Cepstral mean normalization based on a prior window of audio.

Acoustic Model

For fast performance, we opted for a semi-continuous acoustic model. Our acoustic model was trained following the recipe from [27]. We used a HMM topology of 5-states with skip transitions, 256 codebook Gaussians, cross-word triphones, and 39 CMU phones (plus silence). Audio was parameterized into 13 Mel frequency cepstral coefficients plus their deltas and delta-deltas.

Our US-English model was trained on 211 hours of WSJ training data, down-sampled to 8 kHz. We used 8000 tied-states and the CMU pronunciation dictionary.

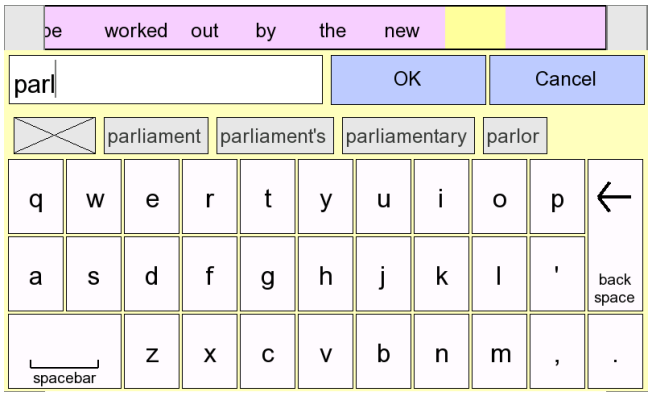

Figure 7. The predictive software keyboard. The user has typed "parl" and the most likely ways to complete the word are displayed in a row below the keyboard.

Our UK-English model was trained on 16 hours of WSJCAM0 training data, down-sampled to 8 kHz. We used 4000 tied-states and the BEEP pronunciation dictionary. We mapped BEEP's phone set to CMU's and added missing words from the CMU dictionary. Gender dependent UK-English models were created using MLLR adaptation of the means followed by MAP adaptation of the means, mixture weights and transition matrices.

For both the US and UK models, speaker-dependent adaptation used MLLR adaptation of the means followed by MAP adaptation of the means and mixture weights.

Language Model

We trained a trigram language model using: newswire text from the CSR-III corpus (222M words), Knesser-Ney Smoothing, and the WSJ 5K word list (without verbalized punctuation). Since our test sentences were taken from the CSR set-aside directory, we purposely chose the WSJ 5K vocabulary in order to introduce a small amount of out-of-vocabulary (OOV) errors. We thought it was important to

validate our design in the face of OOVs as they are typically unavoidable in real-world recognition tasks.

The language model was one of the dominating factors in the memory footprint of our system. Normally, to control language model size, n-grams with a count below a threshold are discarded. We instead used no count cutoffs and performed entropy-pruning [24]. We found entropy-pruning produced compact and well-performing models.

Lattice Processing

We obtained an initial recognition lattice from PocketSphinx [11]. We performed a forward/backward reduction on the lattice to reduce redundant nodes [29]. The bigram lattice was then expanded and rescored using a compact trigram expansion algorithm [29]. The trigram lattice was pruned, removing lattice paths outside a beam width of the posterior probability of the best path. A word confusion network was then created using a clustering algorithm [17]. The confusion network was pruned to limit the maximum size of each cluster and to eliminate clusters where the "no word" hypothesis dominated.

Speaker Adaptation Environment

Since we knew mobile recognition would be challenging, we wanted to improve accuracy by adapting the acoustic model to the user's voice. Adaptation is done by having users read a set of sentences with known transcriptions. With a desktop dictation package, users typically record adaptation data while in front of their desktop computer.

Adapt data	Test data	WER ± 95% CI
none	indoor	15.87 ± 2.16
indoor	indoor	12.61 ± 1.93
outdoor	indoor	12.72 ± 1.94
none	outdoor	21.56 ± 2.53
indoor	outdoor	18.39 ± 2.39
outdoor	outdoor	18.45 ± 2.37

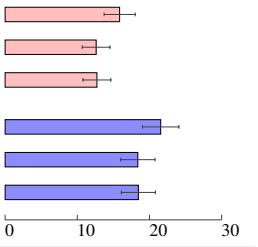

Table 1. The effect of adaptation environment on speech recognition word error rate (WER). The bars show means and 95% confidence intervals (from a bootstrap estimate [2]).

When taking speech recognition mobile, a question arises whether it is worth recording adaptation data in a mobile rather than a desktop setting. Price and colleagues [21] investigated this by having half of their participants record adaptation data while walking on a noisy treadmill, while the other half recorded data while seated in an office (with simulated treadmill noise playing). While not statistically significant, they found that users who had performed adaptation while on the treadmill had a lower word error rate (WER) both when tested on the treadmill and when seated. They suggest that adapting in a more demanding condition might improve overall recognition performance (though they provide no insight into why this might be).

We conducted a small within-subject experiment to test the effects of adaptation environment. We had four US-English speakers record two identical sets of adaptation data on the N800. One was recorded while walking outside and the other while seated inside. For an adaptation set, we used 40 phonetically diverse sentences from the WSJ corpus. Speakers also recorded 125 test sentences from the CSR set-aside directory. Two speakers did the outdoor recording first, while the other two did the indoor recordings first. Recognition experiments were conducted afterwards on a desktop computer.

As shown in Table 1, performing adaptation either indoors or outdoors improved accuracy, reducing word errors by about 20% relative compared to doing no adaptation. We found recognition was much harder outdoors than indoors, increasing the word error rate by 45% relative. The error rates of the indoor and outdoor adapted models were very similar regardless of the test set. Since we found no particular advantage to recording adaptation data outdoors, we decided to record all adaptation data indoors.

USER STUDY

To see how well our system worked in practice, we conducted an initial user study. Our primary aim was to validate our design. In the study, users spoke and corrected sentences while seated indoor and while walking outdoors.

Participants

We recruited four participants (3 males, 1 female) from the university campus. 3 participants used the UK acoustic model, 1 participant used the US model. Their ages ranged from 22 to 39. Participants had no significant prior experience using speech or touch interfaces.

Method and Setup

Participants used a Nokia N800 Internet Tablet (Figure 1). The physical dimensions of the device (length × width × thickness) were 75 × 144 × 13mm. The screen had a resolution of 800 × 480 pixels and a size of 90 × 55 mm.

For stimuli text, we used sentences from the set-aside directory of the CSR-III corpus. These sentences were excluded from language model training. We chose sentences with between 8 and 16 words. Using our 5K WSJ language model, 2.4% of words were out-of-vocabulary and the average per-word perplexity was 18.

Participants took part in a single two-hour session. Participants first trained the speech recognizer for 10 minutes. They then received a 5 minute introduction to the interface, followed by 10 minutes of practice. Participants proceeded to either the seated indoor condition or the walking outdoor condition. In the outdoor condition, participants walked circles around our building on a safe pedestrian path and under constant supervision. Each condition lasted about 30 minutes.

In both conditions, participants were presented with stimuli sentences in a large font. When ready, participants pressed

a MIC ON button, read the sentence, then pressed a MIC OFF button. After a recognition delay, the device beeped and the participant hit a CORRECT button to enter the correction interface. After correcting a sentence, the participant pressed a DONE button to move to the next sentence. Participants were told to proceed "quickly and accurately".

Results

On average, participants completed 41 sentences indoors (sd = 7) and 27 sentences outdoors (sd = 4). Figure 8 shows the GPS track of one participant's outdoor trial.

We found the outdoor condition presented some challenges for our system. Our trials took place during a period of windy weather with an average wind speed of 13 knots, gusting to 28 knots. This made recognition more difficult on the windward sections of the course. In addition, one trial took place on a sunny day and that participant had difficulty seeing the N800's screen on parts of the course.

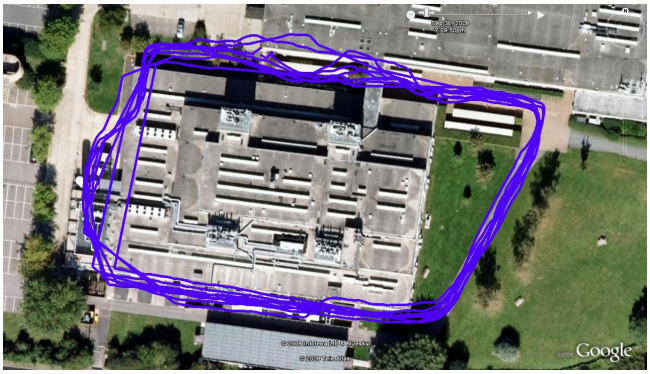

Figure 8. One participant's GPS track line around the Cavendish Laboratory. The image is from Google Earth.

Error Rate

Word error rate (WER) was calculated as the word edit distance between the stimuli sentence and the final sentence divided by the number of words in the stimuli sentence.

Table 2 shows the mean error rates obtained indoors and outdoors. The *before correction* error rate is the error rate of the speech recognizer's output. The *after correction* error rate is the error rate after participants corrected the recognition result. As shown in Table 2, the recognizer's error rate was considerably higher outdoors than indoors. In both conditions, participants corrected almost all errors.

Entry Rate

Entry rate was measured in words-per-minute (WPM). We used the standard convention defining a word as five consecutive characters. The time duration to enter a sentence was calculated as the time between pressing MIC ON and pressing DONE. Table 3 shows the mean entry rates. As expected, users were faster indoors than outdoors.

Correction Method Usage

Participants could correct errors either using the word confusion network or the software keyboard. If participants forget the sentence, they also could invoke a help screen which displayed the stimuli sentence again.

Indoors, participants spent 62% of their time in the word confusion network, 32% in the predictive software keyboard, and 6% in help. Outdoors, participants spent 56% of their time in the word confusion network, 33% in the predictive software keyboard, and 11% in help.

Condition	Text	WER ± 95% CI
indoor	before correction	16.17 ± 4.50
outdoor	before correction	25.63 ± 3.13
indoor	after correction	1.22 ± 1.04
outdoor	after correction	2.23 ± 1.68

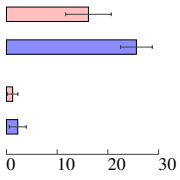

Table 2. Novice users' mean word error rates (WER) and 95% confidence intervals.

Condition	WPM ± 95% CI
indoor	18.36 ± 1.80
outdoor	12.83 ± 0.55

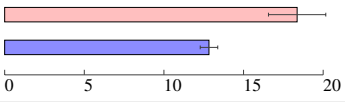

Table 3. Novice users' mean entry rates in words-per-minute (WPM) and 95% confidence intervals.

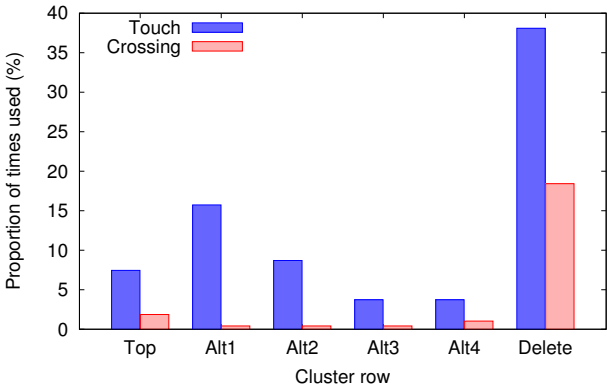

Figure 9. How often words in each row in the confusion network were selected and the method used (touch or crossing). Top is the 1-best result, Alt1-Alt4 are alternative words (Alt1 is the top alternative), Delete is the delete button.

Word Confusion Network Usage

Figure 9 shows the overall usage of the word confusion network. The most commonly selected row in the confusion network was delete. When substituting words, selections decrease in frequency as a function of how far away the words were from the 1-best result (Alt1-Alt4 in Figure 9). This validated our computational results which showed the first few alternatives were the most useful for corrections.

Users most often selected single buttons via touch. When they did select multiple buttons via a crossing gesture, they primarily selected delete buttons. This showed aligning delete buttons in a single row was a useful feature.

Out of 273 tasks, 82 had a completely correct recognition result. Users completed 80 of these tasks without making any unnecessary actions (such as touching a word or using the keyboard). In 27 of the 273 tasks, the sentence could be completely corrected using only the confusion network. In these instances, users corrected 26 of the sentences (96%) using only the confusion network. This shows users took advantage of the confusion network whenever possible.

Of 416 selection actions in the confusion network, users touched a single word 374 times (90%) and crossed multiple words 42 times (10%). When crossing, users selected 2.6 buttons on average. The copy words feature (Figure 3) was only used three times.

Scrolling

Only 2% of recognition results fit on one screen. The average width of a result was 1083 pixels (sd = 266). The display width (minus scroll buttons) was 700 pixels. So on average, users needed to scroll once to check their sentence.

Software Keyboard Usage

Of the 270 times users invoked the keyboard, they chose a morphological prediction 18 times (7%). While indoors, 17% of key presses were the backspace key. While outdoors, 25% of key presses were the backspace key.

In total, participants wrote 265 words with the keyboard. When typing those words, participants used the typing prediction 54% of the time. On average, participants typed about 3 letters (mean = 3.3) before selecting a prediction. When participants did not use prediction, the desired word had been displayed by the system 70% of the time. In these cases, we found on average the user only needed to type a few additional letters (mean = 1.6) to complete their word. This is likely why they ignored the word prediction.

EXPERT PILOT STUDY

To illustrate the potential of speech as a viable mobile text entry method, we measured one expert user (one of the authors) over seven sessions. The goal of the expert pilot was to demonstrate how fast our technique could potentially go (as done in other text entry studies, such as [15]).

Each expert session had an indoor and outdoor condition as previously described. However, instead of a fixed time limit, the expert instead completed a fixed number of sentences. The expert had several years of experience developing and using speech recognition systems. The expert used the US acoustic model.

Results

In total, the expert completed 313 sentences indoors and 309 outdoors. Table 4 shows the recognition word error rates and final corrected error rates obtained by our expert.

Despite relatively long recognition delays (mean = 18 s, sd = 7 s), our expert's text entry rates were surprisingly good (Table 5). If we removed the time the expert spent waiting for recognition, his writing speed over doubled. While having no delay is not realistic, as devices and recognizers become faster, a good amount of this gain could be realized.

Condition	Text	WER ± 95% CI
indoor	before correction	8.46 ± 1.60
outdoor	before correction	14.83 ± 2.17
indoor	after correction	0.94 ± 0.37
outdoor	after correction	1.52 ± 0.99

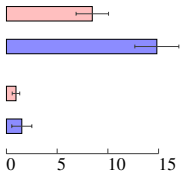

Table 4. Expert mean word error rates (WER). The 95% confidence intervals reflect the individual's variance.

Condition	Rec delay	WPM ± 95% CI
indoor	actual	24.43 ± 0.70
outdoor	actual	19.60 ± 0.72
indoor	none	53.18 ± 1.94
outdoor	none	44.79 ± 2.05

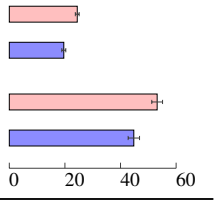

Table 5. Expert mean entry rates. The 95% confidence intervals reflect the individual's variance. The bottom two rows show performance assuming no recognition delay.

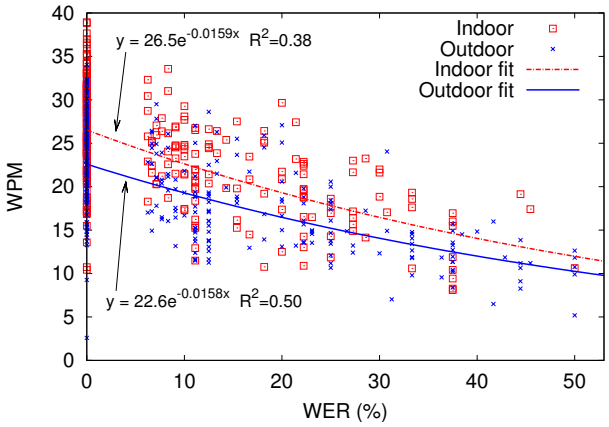

Figure 10. Plot of the entry rate in words-per-minute (WPM) and word error rate (WER) of sentences written by the expert.

As expected, walking outdoors slowed entry, but it did so only by about 20% relative. About half of the expert's sentences (51%) were recognized with no errors. Unsurprisingly, these were completed the fastest (27 WPM indoors, 24 WPM outdoors). The trend lines in Figure 10 show how entry rate slowed as recognition errors increased. As errors increased, the decrease in entry rate was not as dramatic as one might expect. For example, in the 10% word error range, entry rates dropped by only about 15% in both conditions.

DISCUSSION

Limitations

We used a 5K vocabulary and low-perplexity sentences. While we would have liked to test users on more difficult

text, we wanted a "useable" level of recognition errors. In our opinion, there is no point testing speech correction interfaces at really high word error rates. If there are too many errors, it would likely be better not to use speech in the first place. While commercial recognizers like Dragon achieve fast and accurate recognition on difficult text on a desktop, the same is not currently possible using a research recognizer on a mobile device. Recognizers and devices will undoubtedly improve, but in the meantime, we believe we can explore the design and usability of speech interfaces by giving users simpler recognition tasks.

Due to limited computational resources on our device, users experienced long recognition delays (mean = 22 s, sd = 14 s). Some (pathological) utterances took up to a minute to recognize. As devices become faster, delays will be reduced and entry rates should improve. Still, it is worth noting that our novice's (corrected) entry rate of 13 WPM walking outdoors (including these long recognition delays) was still about as fast as the entry rates users obtained while seated indoors using T9 predictive text after several sessions [30].

Design Implications

Review Screen

Many sentences were recognized completely correct. Despite this, users were forced to enter the word confusion network screen and scroll through the entire result to ensure it was correct. It may be advantageous to first allow a simple single screen review of the entire recognition result.

Easy Fallback

We noticed that for some utterances, the recognition result had so some many errors as to make correction an exercise in erasing everything and typing the entire sentence. Our interface should better support this circumstance, allowing users to fallback to keyboard only entry when necessary.

High Contrast

While outdoors, users sometimes found it hard to read the screen because of glare. The user interface could benefit from a redesign that puts more emphasis on high contrast.

More Efficient Use of Screen Real Estate

We found users sometimes had trouble with target selection. Particularly in the word confusion network, buttons could benefit from being larger. Given that users made relatively few selections in the lowest rows in the word confusion network (cf. Figure 9), we may want to remove a row or two to provide space for larger buttons.

Improved Speech Recognition

Recognition delays after users hit the MIC OFF button accounted for 50% of entry times. As mobile devices get faster, these delays will be reduced significantly. This will have a very large impact on the practical entry rates achievable by continuous speech recognition on a mobile device. Improvements in recognition accuracy will also clearly help improve text entry throughput.

CONCLUSIONS

In this paper we presented Parakeet – a touch-screen system for continuous speech recognition on mobile devices. To our knowledge, we are the first to explore a practical mobile continuous speech recognition system for text entry. Our design of Parakeet took advantage of empirical and qualitative findings in the HCI literature. In addition, wherever possible, we adopted an engineering-driven design process where we optimized our user interface based on our system's predicted behavior on empirical data.

In Parakeet, we introduced several novel user interface enhancements. The final design of Parakeet was validated by a user study. We had participants use our system both seated indoors and while walking outdoors. To our knowledge, no speech recognition text entry system has been tested with users actually walking around. Among other things, the study confirmed that word confusion networks were a useful correction interface for users. When the intended sentence was in the confusion network, users were able to find and select it 96% of the time. We also found that participants used the crossing feature about 10% of the time, showing that crossing was a useful complementary feature. Last, we gave practical design recommendations based on lessons learned in our study.

Our expert pilot study demonstrated that speech may be a competitive mobile text entry method, particularly in an actual mobile setting where users are moving around. Our immediate future work is to run a full-scale experiment validating this hypothesis.

ACKNOWLEDGMENTS

We would like to express our gratitude to the study participants. We also thank David MacKay for his many constructive comments and David Huggins-Daines for support and advice on PocketSphinx. This research was in part funded by a donation from Nokia. The following applies to P.O.K. only: The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 220793.

REFERENCES

1. Accot, J. and Zhai, S. More than dotting the i's – foundations for crossing-based interfaces. *Proc. CHI 2002*, ACM Press (2002), 73-80.
2. Bisani, M. and Ney, H. Bootstrap estimates for confidence intervals in ASR performance evaluation. *Proc. ICASSP 2004*, IEEE Press (2004), 409-412.
3. Buxton, W. Chunking and phrasing and the design of human-computer dialogues. *Proc. IFIP World Computer Congress 1986*. IFIP (1986), 475-480.
4. Cohen, J. Embedded speech recognition applications in mobile phones: status, trends and challenges. *Proc. ICASSP 2008*, IEEE Press (2008), 5352-5355.

5. Crossan, A., Murray-Smith, R., Brewster, S., Kelly, J. and Musizza, B. Gait phase effects in mobile interaction. *Ext. Abstracts CHI 2005*, ACM Press (2005), 1312-1315.
6. Darragh, J.J., Witten, I.H. and James, M.L. The reactive keyboard: a predictive typing aid. *IEEE Computer 23*, 11 (1990), 41-49.
7. Fitts, P. The information capacity in the human motor system in controlling the amplitude in movement. *J. Experimental Psychology 47* (1954), 381-391.
8. Goodman, J., Venolia, G., Steury, K. and Parker, C. Language modeling for soft keyboards. *Proc. AAAI 2002*, AAAI Press (2002), 419-424.
9. Hakkani-Tür, D., Béchet, F., Riccardi, G. and Tur, G. Beyond ASR 1-best: using word confusion networks in spoken language understanding. *J. Computer Speech and Language 20*, 4 (2006), 495-514.
10. Hetherington, I.L. PocketSUMMIT: small footprint continuous speech recognition. *Proc. ICSLP 2007*, ISCA (2007), 1465-1468.
11. Huggins-Daines, D., Kumar, M., Chan, A., Black, A.W., Ravishankar, M. and Rudnicky, A.I. PocketSphinx: a free real-time continuous speech recognition system for hand-held devices. *Proc. ICASSP 2006*, IEEE Press (2006), 185-188.
12. Karat, C.M., Halverson, C., Horn, D. and Karat, J. Patterns of entry and correction in large vocabulary speech recognition systems. *Proc. CHI 1999*, ACM Press (1999), 568-575.
13. Karlson, A.K., Bederson, B.B. and Contreras-Vidal, J.L. Understanding one-handed use of mobile devices. In Lumsden, J. (Ed.) *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*. Idea Group (2008), 86-100.
14. Kurihara, K., Goto, M., Ogata, J. and Igarashi, T. Speech Pen: Predictive Handwriting Based on Ambient Multimodal Recognition. *Proc. CHI 2006*, ACM Press (2006), 851-860.
15. Kristensson, P.O. and Zhai, S. Relaxing stylus typing precision by geometric pattern matching. *Proc. IUI 2005*, ACM Press (2005), 151-158.
16. Kristensson, P.O. and Zhai, S. Improving word-recognizers using an interactive lexicon with active and passive words. *Proc. IUI 2008*, ACM Press (2008), 353-356.
17. Mangu, L., Brill E. and Stolcke A. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *J. Computer Speech and Language 14*, 4 (2000), 373-400.
18. Ogata, J. and Goto, M. Speech repair: quick error correction just by using selection operation for speech input interfaces. *Proc. ICSLP 2005*, ISCA (2005), 133-136.
19. Oulasvirta, A., Tamminen, S., Roto, V. and Kuorelahti, J. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. *Proc. CHI 2005*, ACM Press (2005), 919-927.
20. Oviatt, S. Cohen, P., Wu, L., Vergo, J., Duncan, L, Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J. and Ferro, D. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Human-Computer Interaction 15* (2000), 263-322.
21. Price, K.J., Lin, M., Feng, J., Goldman, R., Sears, A. and Jacko, J. Motion does matter: an examination of speech-based text entry on the move. *Universal Access in the Information Society 4* (2006), 246-257.
22. Rosenbaum, D.A. *Human Motor Control*. Academic Press (1991).
23. Shneiderman, B. The limits of speech recognition. *Communications of the ACM 43,* 9 (2000), 63-65.
24. Stolcke, A. Entropy-based Pruning of Backoff Language Models. *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, DARPA (1998), 270-284.
25. Suhm, B., Myers, B. and Waibel, A. Multimodal error correction for speech user interfaces. *ACM TOCHI 8*, 1 (2001), 60-98.
26. Vertanen, K. Efficient computer interfaces using continuous gestures, language models, and speech M.Phil. thesis. University of Cambridge, United Kingdom (2004).
27. Vertanen, K. Baseline WSJ acoustic models for HTK and Sphinx: training recipes and recognition experiments. Technical report, University of Cambridge, United Kingdom (2006).
28. Vertanen, K. and Kristensson, P.O. On the benefits of confidence visualization in speech recognition. *Proc. CHI 2008*, ACM Press (2008), 1497-1500.
29. Weng, F., Stolcke, A. and Sankar, A. Efficient lattice representation and generation. *Proc. ICSLP 1999*, ICSA (1999), 1251-1254.
30. Wobbrock, J.O., Chau, D.H. and Myers, B.A. An alternative to push, press, and tap-tap-tap: gesturing on an isometric joystick for mobile phone text entry. *Proc. CHI 2007*, ACM Press (2007), 667-676.