Intelligently Aiding Human-Guided Correction of Speech Recognition

Keith Vertanen and **Per Ola Kristensson**
Cavendish Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge, CB3 0HE, UK
{kv227,pok21}@cam.ac.uk

Abstract

Correcting recognition errors is often necessary in a speech interface. The process of correcting errors can not only reduce users' performance, but can also lead to frustration. While making fewer recognition errors is undoubtedly helpful, facilities for supporting user-guided correction are also critical. We explore how to better support user corrections using Parakeet – a continuous speech recognition system for text entry. Parakeet's interface is designed for easy error correction on a mobile touch-screen device. Users correct errors by selecting alternative words from a word confusion network and by typing on a predictive software keyboard. Our interface design was guided by computational experiments and used a variety of information sources to aid the correction process. In user studies, participants were able to write text efficiently despite sometimes high initial recognition error rates. Using Parakeet as an example, we discuss principles we found were important for building an effective speech correction interface.

Introduction

Intelligent text entry methods, such as speech and handwriting recognition, use AI techniques to enable users to write faster than otherwise possible, particularly on mobile devices. However, uncertain reasoning methods also risk introducing recognition errors. Indeed, efficient correction methods have been identified as one of the five primary challenges for intelligent text entry methods (Kristensson 2009). The main theme of this paper is that the efficacy of a speech recognition interface depends critically on both recognition accuracy and the design of the correction interface.

When using a speech interface, recognition errors will sometimes occur and those errors will need correction. This is particularly an issue with mobile interfaces due to the challenging acoustic environment they are often used in. In this paper we show how an intelligent user interface aids this user-guided correction process by leveraging information from the recognizer's hypothesis space. In addition, we show how information from complementary knowledge sources can aid in the correction process.

Figure 1: The Parakeet system running on a Nokia N800.

To explore how to create an efficient correction interface, we designed a mobile speech recognition system. This system, called Parakeet, enables users to correct errors using a variety of techniques (Vertanen and Kristensson 2009a). These techniques were guided by a set of design principles and computational experiments on recorded audio.

In this paper, we first describe Parakeet's design. Next we present results from several user studies. Finally, we discuss the factors we found important for building an efficient speech correction interface.

Design Principles

Avoid Cascading Errors Correcting speech recognition errors using only speech can be difficult. This is partly because errors may cascade – recognition errors in the correction phase may require further corrections, and so on (Karat et al. 1999). To avoid this scenario, we use the touch-screen as a fallback input modality for correcting errors.

Exploit the Hypothesis Space Often the best hypothesis proposed by the speech recognizer contains errors. But the user's intended text may exist in one of the many other hypotheses explored during the recognizer's search. By exposing nearby alternatives to the best hypothesis, we let the user quickly scan and select alternative candidates.

Support Fragmented Interaction Users may need to both interact with the device and engage with their environment. In such a situation, users tend to interact with their mobile device in 4–8 second bursts (Oulasvirta et al. 2005). We designed Parakeet to handle this style of interaction by avoiding timing-dependent interaction techniques.

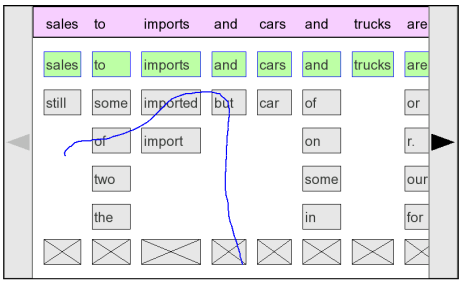

Figure 2: Parakeet's main correction interface. The recognition result is shown at the top. Likely alternative words are displayed in each column. Here the user is changing two words and deleting a third word in one crossing action.

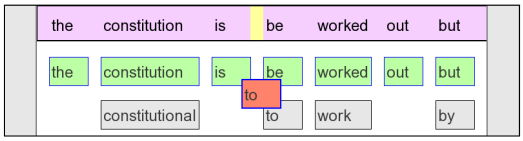

Figure 3: Inserting the word "to" between "is" and "be" by dragging it between columns.

Parakeet Interface

Parakeet runs on mobile Linux devices, such as the Nokia N800 (figure 1). To enter text, users speak into a wireless microphone. Audio is streamed to a speech recognizer running either on the device or on a nearby laptop.

After recognition, the result is shown in Parakeet's main display (figure 2). This display is based on a word confusion network (Mangu, Brill, and Stolcke 2000). A word confusion network is a time-ordered set of clusters where each cluster has competing word hypotheses along with their posterior probabilities. The confusion network is built from the lattice generated during the speech recognizer's search.

The best recognition hypothesis is shown at the top. Each column contains likely alternatives for each recognized word. At the bottom, a series of delete buttons allow words to be removed. The user can scroll left or right by touching the arrow buttons on either side of the screen. Users make corrections by using a number of different actions:

- **Tapping** – An alternate word can be chosen by simply tapping on it.
- **Crossing** – Multiple words can be corrected in a single continuous crossing gesture (figure 2).
- **Copying** – Words can be dragged between different columns or inserted between columns (figure 3).
- **Replacing with variant** – By double-tapping a word, a morphological variant can be chosen (figure 4).
- **Typing** – Arbitrary corrections can be made using a predictive software keyboard (figure 5). As a user types, word completion predictions are offered.

Experimental Driven Design

Our design was guided by computational experiments. We first recognized utterances from a standard test set, creating a confusion network for each utterance. We then simulated an "oracle" user who made optimal use of a given interface design to correct as many errors as possible. While in Parakeet all errors can be corrected via the software keyboard, the oracle user was assumed not to use the keyboard.

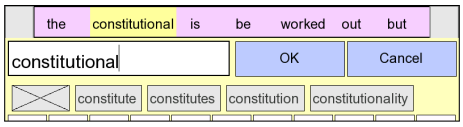

Figure 4: After touching "constitutional", the user is brought to the software keyboard. The morphological variants for "constitutional" are shown in a row.

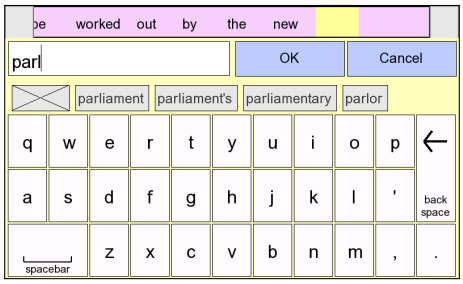

Figure 5: The predictive software keyboard. The user has typed "parl". The most likely word completions are displayed above the keyboard as the user types.

Increasing the number of words in each cluster allowed more errors to be corrected (figure 6). Since the majority of gains were made by adding the first few alternatives, we used a cluster size of five. Minimizing the cluster size allowed larger and easier to touch buttons. We added a delete button to every cluster as this substantially reduced errors (*Del* line, figure 6). Allowing the copying of words between clusters within two words of each other provided a small gain (*Copy+Del* line, figure 6).

When a word is double-tapped in the confusion network, the keyboard interface opens. We predict likely alternatives based on the tapped word. For example, if the user tapped "constitutional", we propose morphological variants, such as "constitute" (figure 4). Providing more predictions provided greater error reductions (figure 7). Since five predictions provided the majority of the gains, we used a single

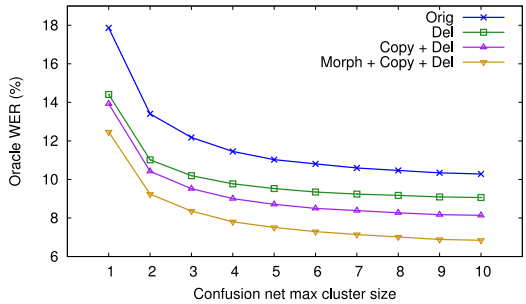

Figure 6: Oracle word error rate (WER) as a function of cluster size in the confusion network. The top line is using the confusion network with no modifications.

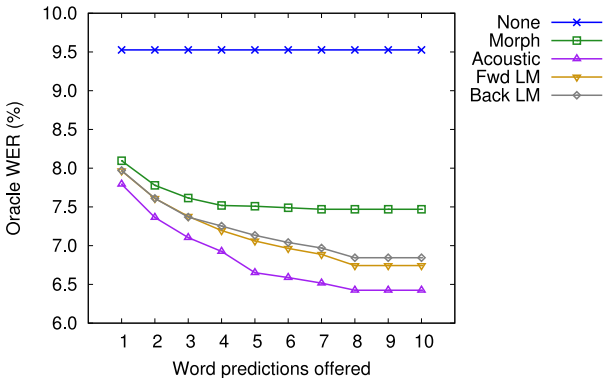

Figure 7: Oracle word error rate (WER) as a function of how many word predictions were given by the keyboard.

prediction row above the keyboard (figure 4).

Before we settled on displaying morphological variants, we considered two other possibilities: acoustically similar words and words probable under an n-gram language model. Acoustically close words have similar, but not identical, phone sequences in a pronunciation dictionary. The language model predictions were based on the preceding word (using a forward language model), or on the following word (using a backward language model).

While acoustic predictions performed best, they are unintuitive to users. As an example, "aback" is acoustically similar to "attack". It would be difficult for a user to know touching "aback" would yield "attack". Language model predictions are also unintuitive as they depend on surrounding context rather than the actual word tapped. For these reasons, we decided to use morphological variants despite their lower performance in our computational experiments.

User Study – Newswire

In our first study, four users spoke and corrected newswire sentences while seated indoors and while walking outdoors (Vertanen and Kristensson 2009a). Recognition was performed on the mobile device. Our key findings were:

- **Error rates** – Users had a word error rate (WER) of 16% indoors and 25% outdoors. After user correction the WER was 1% indoors and 2% outdoors.
- **Entry rates** – Users' mean text entry rate was 18 words per minute (wpm) indoors and 13 wpm outdoors. These rates included somewhat long recognition delays (mean = 22 s). Without these delays, entry rates would have almost doubled. This indicates the potential speech has for efficient mobile text entry. As a reference point, users entering text with T9 predictive text while seated in an office reached a mean entry rate of 16 wpm after 30 sessions (Wobbrock, Chau, and Myers 2007).
- **Use of confusion network** – Users performed corrections via the word confusion network interface when possible. When errors could be completely corrected using the confusion network, users did so 96% of the time. When substituting words, most selections were for the first few alternative words. This validated our computational results that showed the first few alternatives were the most useful.

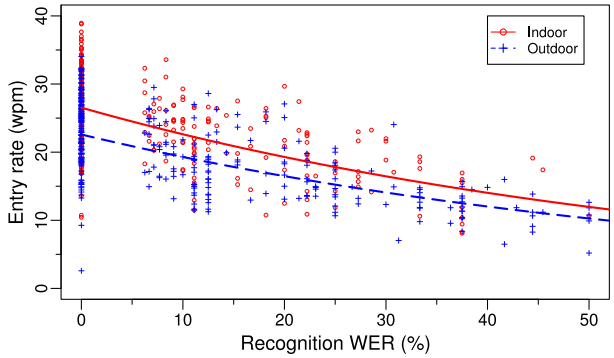

Figure 8: Expert's utterances by entry rate in words per minute (wpm) and word error rate (WER).

- **Touch versus crossing** – 90% of selections used a touch action, 10% used a crossing action. Crossing actions were mostly used to select consecutive delete boxes.
- **Copying** – Copying a word between columns was not a popular feature and was only used 3 times.
- **Predictive keyboard** – When users typed on the keyboard, they used word completion 54% of the time. Users typed 3 letters on average before selecting a prediction.

Expert results We also had an expert (the first author) complete a large number of indoor and outdoor trials. Our expert user achieved high entry rates for a wide-variety of word error rates (figure 8). The expert wrote at 24 wpm while indoors and 20 wpm outdoors. Without the recognition delays, the expert's entry rates would have been 53 wpm indoors and 45 wpm outdoors.

User Study – Voice Search

In our second study, four users spoke and corrected web search queries while walking indoors (Vertanen and Kristensson 2009b). Voice search is challenging because queries tend to be short, involve a diverse vocabulary, and are often performed in a noisy environment. These factors make such queries hard to recognize. We therefore hypothesized that Parakeet could help make voice search practical. For this study, we did recognition on a laptop that was wirelessly connected to the N800 device. However, to users it appeared as if the mobile device performed the recognition.

Due to the short nature of search queries, we report error and entry rates on a per character basis. Users' performance is shown in figure 9. The mean entry rate was 1.7 characters per second (cps). As a reference point, the mean entry rate of mobile Google queries (typically entered via a telephone keypad or a QWERTY keyboard) was 0.42 cps (Kamvar and Baluja 2007). Users spent 47% of their correction time in the confusion network interface and the remainder in the software keyboard.

Discussion

The need to use intelligent reasoning procedures to enable accurate speech recognition is obvious. Indeed, the speech recognition community has improved accuracy by adopting a variety of AI techniques, such as dynamic time warping,

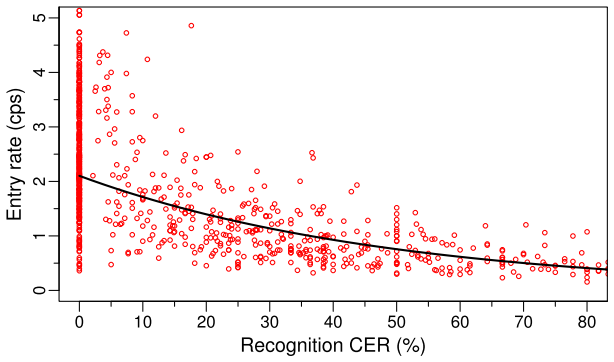

Figure 9: The entry rate in characters per second (cps) and recognition character error rate (CER) for each query entered during our voice search user study.

hidden Markov models, and discriminative training. What is not as obvious is that there are fundamentally two tacts for achieving high entry rates. The first is to improve recognition accuracy. The second is to reduce correction time. While a lot of research has focused on the first, very little has focused on the second. In this paper we demonstrate the dramatic effect a well-designed correction interface can have on entry rates, even if the recognition error rate is high. So how do we design efficient speech correction interfaces?

First, we demonstrate the need to run human factors experiments. Without such a user-centered approach, it is tempting to devote too much effort optimizing a single metric of system performance. For example, eking out a small reduction in the 1-best WER may not be the best route to substantial improvements in entry rates or user satisfaction. Instead, there may be other design issues that are more important. For example, we learned a higher contrast color scheme was needed to improve readability in sunlight. We also found that buttons in our confusion network were too small and, since users rarely used low probability words, we should present fewer words but with bigger buttons.

Second, knowing how well an interface does at one particular recognition operating point is not that informative since recognition accuracy varies depending on the current state of recognition technology, as well as on the specifics of the recognition task. By looking at the performance envelope (e.g. figure 8 and 9), we learn how the interface performs at a variety of recognition error rates. It also allows different interfaces to be more easily compared even if those interfaces have different mean recognition error rates.

Third, computational experiments can guide interface design by making explicit the advantages and disadvantages of different correction techniques and design decisions. This experimental-driven design helped us choose which correction techniques to include in Parakeet. However, it is important to not blindly follow these experiments. In the design of Parakeet's prediction interface, we found that acoustically-based predictions performed best in computational experiments. However, we decided to use morphologically-based predictions as they are more intuitive to users.

Fourth, efficient correction interfaces should leverage complementary knowledge sources. In Parakeet, the speech recognizer provides the primary knowledge source for correction. But when this source fails, we use other complementary knowledge sources to assist correction. For example, we use morphological alternatives proposed by the Snowball stemmer (Porter 2001). Additionally, the word predictions in the software keyboard use a language model different from the one used by the speech recognizer.

Conclusions

When using speech to enter text, users' entry rates depend importantly on the efficiency of the correction interface. Our intelligent correction interface is based on a word confusion network. Our design was guided by a set of design principles and computational experiments on recorded audio. Our experiments reveal that users can enter text and search queries relatively quickly, even while walking. Our data also shows that users take advantage of the correction interface whenever possible. Our findings demonstrate that in addition to reducing the recognition error rate, we should also focus on creating more intelligent correction interfaces.

Acknowledgments

This research was funded in part 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

Kamvar, M., and Baluja, S. 2007. Deciphering trends in mobile search. *IEEE Computer* 40(8):58–62.

Karat, C.-M.; Halverson, C.; Horn, D.; and Karat, J. 1999. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proc. CHI '99*, 568–575.

Kristensson, P. O. 2009. Five challenges for intelligent text entry methods. *AI Magazine* 30(4):85–94.

Mangu, L.; Brill, E.; and Stolcke, A. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language* 14(4):373–400.

Oulasvirta, A.; Tamminen, S.; Roto, V.; and Kuorelahti, J. 2005. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. In *Proc. CHI '05*, 919–928. ACM.

Porter, M. 2001. Snowball: A language for stemming algorithms. `http://snowball.tartarus.org/texts/introduction.html`.

Vertanen, K., and Kristensson, P. O. 2009a. Parakeet: A continuous speech recognition system for mobile touch-screen devices. In *Proc. IUI '09*, 237–246. ACM.

Vertanen, K., and Kristensson, P. O. 2009b. Recognition and correction of voice web search queries. In *Proc. Interspeech '09*, 1863–1866.

Wobbrock, J. O.; Chau, D. H.; and Myers, B. A. 2007. An alternative to push, press, and tap-tap-tap: Gesturing on an isometric joystick for mobile phone text entry. In *Proc. CHI '07*, 667–676. ACM.