Asynchronous Multimodal Text Entry using Speech and Gesture Keyboards

Per Ola Kristensson[1], *Keith Vertanen*[2]

[1]School of Computer Science, University of St Andrews, United Kingdom
[2]Department of Computer Science, Princeton University, New Jersey, USA
pok@st-andrews.ac.uk, vertanen@princeton.edu

Abstract

We propose reducing errors in text entry by combining speech and gesture keyboard input. We describe a merge model that combines recognition results in an asynchronous and flexible manner. We collected speech and gesture data of users entering both short email sentences and web search queries. By merging recognition results from both modalities, word error rate was reduced by 53% relative for email sentences and 29% relative for web searches. For email utterances with speech errors, we investigated providing gesture keyboard corrections of only the erroneous words. Without the user explicitly indicating the incorrect words, our model was able to reduce the word error rate by 44% relative.

Index Terms: mobile text entry, multimodal interfaces

1. Introduction

It is difficult to correct speech recognition errors using speech alone. Therefore speech interfaces often provide users with a secondary input modality. In this paper we propose using a capacitive touch-screen gesture keyboard as a secondary modality for speech. Capacitive touch-screens are attractive because of their high-quality continuous touch-signals and their increasing ubiquity on mobile phones. Previously Sim [1] used a capacitive touch-screen keyboard to allow users to input information such as word boundaries or the first letter of the intended word. Sim found that touch input reduced the decoding time and the error rate on a 5K WSJ task.

Here we find that we can substantially reduce error rates by combining speech with a touch-screen gesture keyboard. A gesture keyboard enables users to quickly write words by swiping a finger over the touch-screen keyboard [2]. For example, to write the word "speech" the users pushes down on the S key and slides to the P, E, C and H keys before lifting up to complete the gesture (see figure 1). The system then performs a pattern match to find the word whose shape on the keyboard most resembles the gestured shape. Gesture keyboards have been commercialized as ShapeWriter, Swype, T9 Trace and Flext9.

We merge the speech and gesture keyboard modalities using our recently proposed merge model [3]. This model supports two key features: asynchronicity and *spotty correction*. Asynchronicity means that users do not need to synchronize speech and gestures. Informal testing revealed that users have difficulty speaking and gesturing simultaneously. This is similar to Sim's [1] finding that a user had difficulty speaking while touching the first letter of each word. In addition, the asynchronous merge model also allows users to perform *post hoc* error correction in which recognition results from a primary modality are later merged with the recognition results from a secondary modality.

The second key feature is spotty correction. Our merge model can merge recognition results from a modality that only received a subset of the words received by the other modality. This enables spotty correction in which the user only needs to input the erroneous words in the second modality to correct the errors in the first modality. This is done without explicitly specifying the erroneous words. For example, say the user spoke "the cat sat" and the system recognized "the *bat* sat". Using spotty correction the user simply gestures "cat" and the system then attempts to locate and replace the erroneous word.

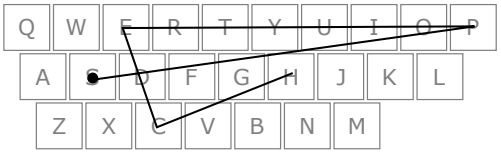

Figure 1: *The ideal shape of the word "speech" on a gesture keyboard. The starting point is indicated with a dot.*

2. Asynchronous Multimodal Text Entry

2.1. Speech Recognition

We used the CMU Sphinx speech recognizer, training a US-English acoustic model on 211 hours of WSJ data. We trained cross-word triphones with a 3-state left-to-right HMM topology. We used a 39-dimensional feature vector with 13 Mel-frequency cepstral coefficients, deltas and delta deltas. Our model had 8000 tied-states with 16 continuous Gaussians per state and diagonal covariance matrices. We used the CMU pronunciation dictionary (39 phones plus silence). Audio was recorded at 16 kHz. We performed cepstral mean normalization based on a prior window of audio. The recognizer was adapted to each user's voice using 40 sentences. We adapted the model means using maximum likelihood linear regression with 7 regression classes. We used the PocketSphinx decoder and tuned it to near real-time recognition.

Our email language model was trained on text from a Usenet corpus (1.8B words), a blog corpus (387M words), and four months of Twitter messages (109M words). We collected the Twitter data using the free streaming API which provide access to 5% of all tweets. We trained our language model only on sentences that were similar to short email sentences using cross-entropy difference selection [4]. In this recently proposed method, each sentence is scored by the difference in per-word cross-entropy between an in-domain language model and a background model. We did this separately for the Usenet, blog and Twitter datasets. Our in-domain trigram language model was trained on sentences drawn from the W3C corpus and the non-spam messages in the TREC 2006–7 spam track. We only used sentences with six or fewer words. Our background models were trained on a similar amount of training data as our in-

domain model but used sentences from Usenet, blog or Twitter. For each of these three sources we chose the cross-entropy difference threshold that optimized performance on held out W3C and TREC data. We built a mixture model from our three language models using linear interpolation with mixture weights optimized on the held out data. Our mixture model had 43M n-grams. All models used interpolated modified Kneser-Ney smoothing with no count cutoffs and a 64K vocabulary.

2.2. Gesture Keyboard Recognition

We will now describe the gesture recognition procedure, which is an adaptation of the standard algorithm [2].

If the spatial length of the trace is less than a threshold (38 pixels or 6 mm on a 4th generation iPod Touch), then the system assumes the user intended to touch a single key rather than articulate a touch-screen gesture. Unlike previous work [2] which only returned the nearest key on the keyboard when the user tapped a single key, we provide additional letter hypotheses to the merge model by computing a likelihood P_k for each key k:

$$P_k = \exp\left(-\frac{d_k^2}{\sigma_k^2}\right), \tag{1}$$

where d_k is the Euclidean distance between the first touch-point of the user's trace and the key k, and σ_k is a variance estimate. Motivated by previous empirical work on modeling on-screen keyboard touch-errors [5] we assume the distance between a touch-point and the center of an intended key is approximately normal.

If the trace is not a tap then we need to recognize the user's continuous gesture as a word. First define $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ as a point in homogeneous coordinates on a two-dimensional Cartesian plane. Then let the sequences $U = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n)$ and $V = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n)$ be two ordered sequences of n equidistant points. The sequence U represent the series of two-dimensional touch-points the user has traced on the touch-screen. The sequence V represents the shape of the ideal traced out word on the keyboard layout (see figure 1). This shape is generated by serially connecting the centers of the corresponding letter keys for a word. Both sequences are resampled to have the same number of n sampling points. Next we define $\mathbf{T}$ as an affine transform, also in homogeneous coordinates:

$$\mathbf{T} = \begin{bmatrix} s & 0 & dx \\ 0 & s & dy \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Here dx and dy are the horizontal and vertical translation components, and $s \in (0, 1]$ is a scale factor. dx and dy are set to the respective one-dimensional distances between the centroids of U and V, and s is set to the maximum ratio of the diagonals the bounding boxes of U and V. Given U and V we compute a likelihood P_w of a word w as:

$$P_w = \exp\left(-\left[\left(\frac{x_s^2}{\sigma_s^2}\right) + \left(\frac{x_l^2}{\sigma_l^2}\right)\right]\right), \tag{3}$$

where σ_s and σ_l are variance estimates, and

$$x_s = \frac{1}{n}\sum_i (||\mathbf{T}\mathbf{u}_i - \mathbf{v}_i||), \tag{4}$$

and

$$x_l = \frac{1}{n}\sum_i (||\mathbf{u}_i - \mathbf{v}_i||). \tag{5}$$

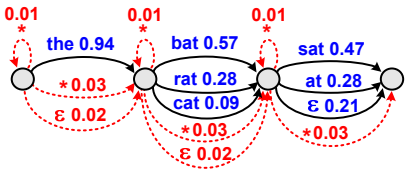

Figure 2: *An example of a speech recognition* WCN *that has been softened. The added edges are in dotted red. The * symbol represents wildcard transitions. This figure is adapted from [3].*

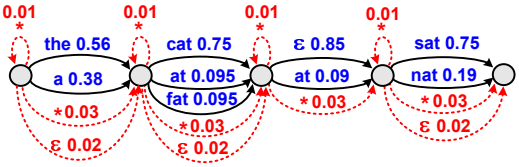

Figure 3: *An example of a gesture keyboard recognition* WCN *with added dotted red edges. This figure is adapted from [3].*

x_s is a matching score between U and V which is scale and translation invariant. x_l is a similar score, but is dependent on where U and V are positioned on the on-screen keyboard. It is possible to make the recognition scale-translation invariant by setting $x_l = 0$.

We tuned the number of sampling points and the parameters σ_k, σ_s and σ_l to optimal values on a development set. The gesture keyboard recognizer used the same 64K vocabulary as the speech recognizer. Each gesture is recognized independently and produces a set of words and likelihoods under the model. We then construct a lattice that connects each word with every word in the subsequent set. This lattice is then rescored with the speech recognizer's trigram language model. From the rescored lattice we construct a word confusion network (WCN) [6].

2.3. Merge Model

To combine the speech and gesture modalities we use a merge model that we have previously developed [3]. This model is capable of combining output from several recognizers asynchronously. The model was originally developed for combination of multiple speech signals for a speech-only correction interface [7]. Here we demonstrate how this model can also be used to fuse speech and gesture keyboard recognition results. What follows is a high-level overview of how the model works with illustrative figures adapted from the original paper [3].

The model operates on WCNs. The original WCNs are softened by adding three extra transitions to every cluster. First, an epsilon transition is added that enables the current cluster to go to the next cluster without generating a word. Second, a wildcard self-loop enables the current cluster to generate any word while remaining in the same cluster. Third, a wildcard-next transition allows a cluster to generate any word and proceed to the next cluster. The probability of each of these added transitions can be varied between the WCNs being combined and can also be varied between different clusters within a WCN.

The first WCN is obtained from the speech recognizer. The second WCN is obtained from the gesture keyboard recognizer. Figures 2 and 3 show example WCNs from the speech and gesture keyboard recognizers.

The model works by searching for a joint path through the softened WCNs. We explain the search using the token passing model [8]. A token in our model tracks three pieces of information. First, the position in each of the WCNs. Second, the accumulated log probability. Third, the previous few words of

language-model context.

A search is initiated with a token that starts in the first cluster in both WCNs. A token is finished when it reaches the last cluster in both WCNs. At each step of the search, we select a token from the pool of unfinished tokens. From the selected token's position in each WCN, we compute all possible moves that generate a single word (either a real word or a wildcard word). We then take the cross-product between candidate moves in each WCN. We consider a combination of moves valid only if it obeys two rules. First, at least one of the moves must generate a real word (i.e. not every WCN can use a wildcard). Second, if multiple WCNs generate real words, these words must match.

Every move is assessed a probability under a language model. The merge model uses the same language model as both recognizers. Since large WCNs have a vast number of possible combinations, an admissible search is intractable. We apply pruning beams to focus the search on only the most promising possibilities. See the original paper [3] for more details.

The free parameters of the merge model, such as the wildcard and epsilon transitions, were tuned on speech and gesture keyboard development data recorded by the authors. As we will describe shortly, the model was tested in three distinct scenarios: merging full speech and gesture results, merging spotty corrections, and merging preemptive corrections. Different sets of parameter values were tuned for each scenario.

3. Recognition Experiments

3.1. Mobile Email

We tested the entry of brief mobile email sentences using speech and a gesture keyboard. We selected sentences of length 1–6 words typed by Enron employees on their BlackBerry devices [9]. We collected the data for speech and gesture keyboard from separate pools of participants. Experiments were done offline.

3.1.1. Data Collection

Four American English speakers spoke the email sentences. Their audio was recorded at 16 kHz using a Plantronics Voyager Pro wireless microphone. The other four participants used a gesture keyboard to write the email sentences. These participants used a 4th generation iPod Touch with a capacitive touch-screen. The gesture keyboard on-screen display measured 49.9 mm $\times$ 22.4 mm (320 $\times$ 144 pixels). The dimensions of each individual key measured 5.0 $\times$ 7.5 mm (32 $\times$ 48 pixels). Each sample point received from the capacitive touch-screen was displayed as a red dot to provide trace feedback to the participant. The interface is shown in figure 4. In total we collected 148 paired sentences with each participant doing between 32 and 41 sentences. The sentences had an out-of-vocabulary rate of 0.7% with respect to our 64K vocabulary.

3.1.2. Results

We first tested the merge model on complete utterances with gesture traces for every word in each utterance. The results are shown in table 1. Overall speech recognition (SR) was the least accurate modality with 27% WER. The gesture keyboard (GK) was much better at 14%WER and the scale-translation invariant version of the gesture keyboard (IGK) performed about the same (14% WER). We find it interesting that the scale-translation invariant version of the gesture keyboard had similar performance as the location-dependent version. We conjecture this is because location information can both aid and hinder recognition

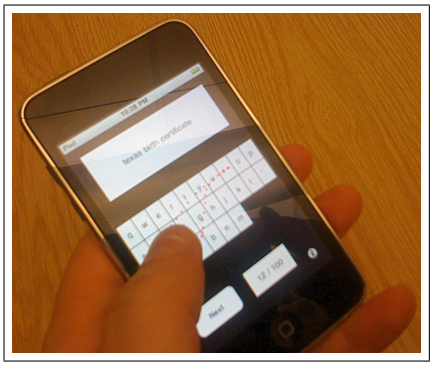

Figure 4: *The iPod Touch gesture keyboard interface.*

Recognizer(s)	Combo model	WER	SER	Oracle WER
SR	-	27.2%	54.7%	8.6%
GK	-	14.2%	44.6%	8.1%
IGK	-	14.1%	41.9%	8.2%
SR+GK	Merge	6.6%	25.0%	3.3%
SR+IGK	Merge	6.6%	25.0%	3.5%
SR+GK	CNC	10.3%	32.4%	1.0%
SR+IGK	CNC	7.7%	27.0%	1.2%

Table 1: *Results for a single modality and for combining modalities in the mobile email domain.*

depending on how carefully the user is gesturing on the keyboard. We also computed the WCN oracle WER which is the path through the WCN with the lowest error rate. As expected the oracle WER was substantially lower for all modalities.

Combining the modalities resulted in a 53% relative reduction in WER compared to just using the gesture keyboard, the most accurate single modality. Our best result from the merge model was at a lower WER than even the best oracle WER of any of the modalities. This demonstrates the advantage of complementary modalities which recognize input in different ways. We also compared our merge model to confusion network combination (CNC) [10] as implemented by SRILM [11]. Our merge model model provided superior gains to CNC in WER. However, CNC did have a better oracle WER. The oracle WER for CNC is better since nothing gets eliminated during the combination while our merge model performs pruning during its search.

As shown by the sentence error rate (SER) in table 1, combining both modalities results in three out of four sentences being recognized completely correct. With speech alone less than half of all sentences were recognized completely correct.

Our merge model is also capable of spotty error correction. In spotty correction, one modality only receives recognition input for a time-ordered subset of the words of the full sentence in the other modality. Our idea is to enable users to see incorrectly recognized words in the speech modality and trace only the words that are in error. This is done without providing any location information about where the incorrect words are located in the speech result. Table 2 shows the results on using spotty correction on the 81 sentences which had at least one incorrect word in the speech modality. As shown in the table, spotty correction substantially reduced WER by 44% relative.

Last, we also tested preemptive error correction. In preemptive error correction, users speak an utterance and simultaneously, or shortly thereafter, input a single word using a gesture keyboard. If users can predict the most likely word to be mis-

Recognizer(s)	Combo model	WER	SER
SR	-	48.5%	100.0%
SR+GK	Merge	28.8%	79.0%
SR+IGK	Merge	27.1%	81.5%

Table 2: *Results when merging spotty gesture keyboard corrections with speech recognition results for email sentences where the speech recognizer made at least one word error.*

Recognizer(s)	Combo model	WER	SER
SR	-	27.2%	54.7%
SR+GK	Merge	26.6%	58.8%
SR+IGK	Merge	26.8%	58.8%

Table 3: *Results when merging a one word preemptive correction with speech recognition results in the mobile email domain.*

recognized by the speech recognizer, a successful merge may prevent the error. To test the viability of this idea, we first recruited two participants who did not take part in any of the other data collection tasks reported in this paper. None of the participants had any speech recognition experience. These participants were shown email sentences and instructed to underline a single word in each sentence which they thought was the most likely to be misrecognized. We then selected these individual words from the gesture keyboard data and merged the resulting WCNs against the complete sentences from the speech recognizer. Table 3 shows that this form of preemptive error correction had little impact.

3.2. Mobile Web Search

We also tested our merge model in the mobile web search domain. We used the system and the data from our past work [12].

We collected search queries from seven participants recruited from the university campus. Each participant was shown a search query on the screen and asked to input it using either speech or the gesture keyboard while simultaneously walking around. The three American English participants who spoke the search queries had their audio recorded at 16 kHz using a Jabra M5390 wireless microphone. The other four participants used a gesture keyboard to write the search queries. These participants used the same iPod Touch device and interface as described for mobile email. In total, we collected 398 paired search queries with each participant doing between 80 and 120 queries.

The resulting WER for speech recognition (SR), gesture keyboard (GK), and a combination of the two are shown in table 4. Combining the modalities resulted in a 29% relative reduction in WER in comparison to just using the gesture keyboard.

Recognizer(s)	Combo model	WER	SER
SR	-	34.2%	49.3%
GK	-	15.3%	31.4%
SR+GK	Merge	10.8%	24.6%

Table 4: *Results when using a single modality and when combining modalities in the mobile search domain.*

4. Conclusions

We have shown how speech and gesture keyboard input can be combined to reduce errors in text entry. We described a merge model that combined recognition results in an asynchronous and flexible manner. We collected speech and gesture data from users entering both email sentences and web search queries. By merging recognition results from both modalities, word error rate was reduced by 53% relative for emails and 29% relative for web searches. For email utterances with speech errors, we investigated providing gesture keyboard corrections of only the erroneous words. Without the user needing to explicitly indicate the incorrect words, our model was able to reduce word error by 44% relative. Our results show that the gesture keyboard is a promising complementary input modality to speech.

5. Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (grant number EP/H027408/1).

6. References

[1] K. C. Sim, "Haptic voice recognition: Augmenting speech modality with touch events for efficient speech recognition," in *Proceedings of the 3rd IEEE Workshop on Spoken Language Technology*. IEEE Press, 2010, pp. 73–78.

[2] P. O. Kristensson and S. Zhai, "SHARK^2: a large vocabulary shorthand writing system for pen-based computers," in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. ACM Press, 2004, pp. 43–52.

[3] K. Vertanen and P. O. Kristensson, "Getting it right the second time: Recognition of spoken corrections," in *Proceedings of the 3rd IEEE Workshop on Spoken Language Technology*. IEEE Press, 2010, pp. 277–282.

[4] R. C. Moore and W. Lewis, "Intelligent selection of language model training data," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL, 2010, pp. 220–224.

[5] J. Goodman, G. Venolia, K. Steury, and C. Parker, "Language modeling for soft keyboards," in *Proceedings of the 17th National Conference on Artificial Intelligence*. AAAI Press, 2002, pp. 419–424.

[6] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[7] K. Vertanen and P. O. Kristensson, "Automatic selection of recognition errors by respeaking the intended text," in *Proceedings of the 11th Biannual IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE Press, 2009, pp. 130–135.

[8] S. J. Young, N. Russell, and J. Thornton, "Token passing: A simple conceptual model for connected speech recognition systems," University of Cambridge, Tech. Rep., 1989.

[9] K. Vertanen and P. O. Kristensson, "A versatile dataset for text entry evaluations based on genuine mobile emails," in *Proceedings of the 13th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM Press, 2011, forthcoming.

[10] G. Evermann and P.C.Woodland, "Posterior probability decoding, confidence estimation and system combination," in *Proceedings of the NIST Speech Transcription Workshop*, 2000.

[11] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of the International Conference on Spoken Language Processing*. ISCA, 2002, pp. 901–904.

[12] K. Vertanen and P. O. Kristensson, "Recognition and correction of voice web search queries," in *Proc. Interspeech 2009*. ISCA, 2009, pp. 1863–1866.