GETTING IT RIGHT THE SECOND TIME: RECOGNITION OF SPOKEN CORRECTIONS

Keith Vertanen, Per Ola Kristensson

Cavendish and Computer Laboratories, University of Cambridge, UK

ABSTRACT

We investigate ways to improve recognition accuracy on spoken corrections. We show that a variety of simple techniques can greatly improve the accuracy on corrections. We further develop a flexible merge model that improves accuracy by combining information from the original recognition and the spoken correction. Our merge model operates on word confusion networks and can easily incorporate prior beliefs about the recognition events (e.g. which words are likely correct or incorrect). By combining all of our techniques, the percentage of correctly recognized spoken corrections increased from 21% to 53%.

Index Terms— speech recognition, error correction, spoken corrections, word confusion network combination

1. INTRODUCTION

A practical speech-based text entry interface needs to not only minimize the recognition error rate, but also enable users to efficiently correct errors. Depending on the usage scenario, there are different ways to tackle error correction. For mobile speech dictation it may be possible to completely rely on an alternative modality, such as a touch-screen (e.g. the Parakeet system [1]). However, there are some usage scenarios, such as in-car systems, that require error correction to be done solely by speech. Even for desktop use, users suffering from repetitive strain injuries (RSI) may want to rely on speech as much as possible. Additionally, users initially tend to try to correct speech recognitions errors using speech [2]. Users often remain in the speech modality even when faced with repeated recognition errors [3]. In this paper we look at how to improve recognition of spoken corrections.

Typically, voice-only correction uses a two-step process. In the first step, users select a portion of the misrecognized text by voice (e.g. "select the *bat* sat"). Next, they speak their intended replacement text (e.g. "the *cat* sat"). However, as first suggested by McNair and Waibel [4], this two-step process can be replaced by a one-step process. In one-step correction, users speak their correction and the system infers both the error region and the replacement text. In prior work, we developed a system in which users speak their intended text and the error region is automatically determined [5]. Our system can also use an additional weak input channel to improve its detection. For example, the user can indicate an error by swiping their finger across a touch-screen (figure 1). In this work, we look at the next phase of the problem: accurately determining the replacement text.

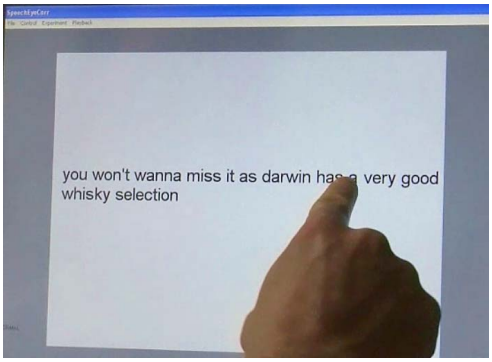

Fig. 1. The user is indicating the error region to the system.

We make the following contributions. First, we detail a series of simple and effective ways to cut the word error rate of corrections. Then we show the different ways language modeling information can be used to aid the recognition of corrections. Thereafter we present a merge model that combines information from the initial recognition result and the subsequent spoken correction. Last, we show how using all the above techniques can more than double how often spoken corrections are recognized completely correct.

2. DATA COLLECTION AND SETUP

In prior work [5], we had users read sentences to a speech recognition interface. The recognition result was displayed with any errors highlighted in red. Users were then prompted to respeak portions of the sentence around any errors (figure 2). Users spoke sentences drawn from the WSJ0 si_et_05 and WSJ1 si_et_s2 test sets. The corrections contained varying amounts of correct context around the error. We collected some utterances with no correct context, some with left context, some with right context, and some with both.

Our development set consists of 401 spoken corrections from the first three users (one of which was the first author). We used this set throughout this paper to tune our model parameters. The remaining 8 users were used as an evaluation test set. The evaluation set has 384 full sentences and 832

Fig. 2. Example of a user being asked to provide a correction with one word of correct context on the right.

spoken corrections. The full sentences consist of both sentences that were recognized completely correct and sentences which had recognition errors.

2.1. Recognition setup

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 parameterized audio into a 39-dimensional feature vector consisting of 12 Mel-frequency cepstral coefficients plus the 0^{th} cepstral, deltas and delta deltas. We used 8000 tied-states with 16 continuous Gaussians per state and diagonal covariance matrices. We used the CMU pronunciation dictionary without stress markings (39 phones plus silence).

We streamed audio sampled at 16 kHz to the recognizer as soon as the Sennheiser PC 166 microphone was enabled. 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 provide near real-time recognition. We trained a trigram language model (LM) using text from the CSR-III newswire corpus (222M words), the most frequent 64K words, and interpolated modified Knesser-Ney smoothing.

3. ACOUSTIC MODEL AND DECODER

To improve accuracy on corrections, we made a number of changes to our acoustic model, decoding parameters, and audio processing. In this section, we describe each improvement (see table 1 for a summary of results).

Recognizing corrections with the same setup used for the initial full sentences had a high word error rate (WER) of 55%. Our first modification was to use a more complex acoustic model with more Gaussians per state (32 versus 16) and a more complex HMM topology (5-states with skip transitions versus 3-states without skips). This resulted in a 6.6% absolute reduction in WER.

Second, we tuned the decoder's insertion penalty, silence penalty, and language model scale factor on our development set of corrections. We found corrections required a much higher penalty for insertions, a lower penalty for silence, and a slightly higher language model scale factor. Changing these parameters resulted in a further 1.9% reduction in WER.

Third, we allowed the decoder to search harder. This (along with the more complex acoustic model) significantly slowed down recognition from 1.4 to $4.7 \times$ RT. But such a slowdown is probably justified since corrections tend to be short and the user experience can seriously degrade if errors occur while trying to correct errors [3]. Using wider beams resulted in a further 4.0% reduction in WER.

Finally, the corrections contained substantial amounts of silence. We discovered that this silence was adversely affecting our cepstral mean normalization. We used Sphinx's silence filtering module to remove silence sections (from all parts of the utterance, not just the start and end). Sphinx's silence filter has 12 free parameters. We tuned these parameters to provide good performance on a set of sentence, phrase, and word utterances recorded by the first author. Silence filtering made a big difference, reducing WER by another 12%.

3.1. Impact of model switching

Zweig [6] showed that switching to a different acoustic model significantly improved accuracy for web queries repeated after an error. We conducted an experiment to see if such a system switching approach could improve recognition for our problem domain. All our acoustic models were trained on identical WSJ training data and used the same trigram language model. We varied aspects of the acoustic model, such as front-end parameterization, HMM topology, number of tied-states, number of Gaussians per state, and decoder used (PocketSphinx or HDecode). We used speaker independent models and utterance-wide cepstral mean normalization.

We compared gains on our corrections (utterances known to be problematic) versus gains on a test set of a thousand WSJ utterances (utterances that may or may not be difficult to recognize). As expected, we found improvements on both test sets using more complex models with wider beams (table 2). However, we in fact saw larger relative gains on the WSJ data than we saw on the correction data. It is unclear why this was the case. Perhaps we should have used different acoustic training data for our correction acoustic model (as in [6]).

4. LANGUAGE MODEL

Normally, language models are trained on full sentences with special pseudo-words placed at the start and end of every sentence. A recognizer's search usually starts by assuming a language model context of the start word. This is suboptimal for corrections as they can consist of words appearing anywhere in a sentence.

The first and most obvious improvement is to use the words preceding and following the correction location. This has previously been shown to be effective [7]. In this paper, we assume the location of the correction is known exactly (perhaps the user has selected the region with a mouse or perhaps we have successfully inferred the location).

	Corrections			Full sentences		
	WER	Δ	×RT	WER	Δ	×RT
Original acoustic model, 3 state 16 Gaussians	54.9	.	1.4	18.6	.	0.9
+ Different acoustic model, 5 state 32 Gaussians	48.3	12.0%	3.4	15.6	16.1%	2.6
+ Correction-specific penalties	46.4	15.5%	3.3	15.8	15.1%	2.5
+ Wider beam widths	42.4	22.8%	4.7	12.0	35.5%	3.6
+ Silence filtering	30.8	43.9%	1.9	11.9	36.0%	3.1

Table 1. WER, relative improvement, and real-time factor on the corrections and the original full sentence utterances.

Acoustic model					Corrections			WSJ sentences		
features	topology	tied states	densities	decoder	WER	Δ	×RT	WER	Δ	×RT
MFCC	3 state	8000	16 / state	PocketSphinx	68.0	.	1.7	12.6	.	1.1
MFCC	3 state	10000	32 / state	PocketSphinx	63.2	7.1%	4.4	9.6	23.8%	3.2
MFCC	3 state	8000	64 / state	PocketSphinx	62.9	7.5%	6.9	9.4	25.4%	4.6
MFCC	5 state	8000	32 / state	PocketSphinx	59.3	12.8%	5.5	9.3	26.2%	3.8
MFCC	5 state	8000	1024 codebook	PocketSphinx	58.7	13.7%	2.8	10.0	20.6%	2.0
PLP	3 state	13000	16 / state	HDecode	49.3	27.5%	6.4	6.9	45.2%	3.6
MFCC	3 state	10000	32 / state	HDecode	45.0	33.8%	9.3	6.7	46.8%	7.3
Average					58.1	17.1%	5.3	9.2	31.3%	3.7

Table 2. Results of the acoustic model switching experiment. The top row is the acoustic model that misrecognized the initial sentence. WER is shown for the subsequent correction utterances as well as on a general WSJ test set.

There are two choices about when to incorporate context. The first is to use the context to weight the hypotheses explored during the recognizer's search. The second is to apply the context only after recognition completes, rescoring paths in the lattice. The advantage of using the context during the search is that this may avoid search errors resulting from using the wrong language model context. The disadvantage is that the correction location must be known before decoding can begin. An additional disadvantage is that multiple searches are required if several candidate locations are to be evaluated.

If we believe utterances are short and may consist of words drawn mid-sentence, we can also modify our language model's training data. We did this by cutting up the sentences in our newswire data by inserting sentence start and end symbols between every 1–4 words. Such a language model (which we term here a fragment LM) has the advantage that no knowledge about the context is needed.

As shown in table 3, using context was advantageous. It was somewhat better to apply context during the search than afterwards. Using the fragment language model was also surprisingly effective, even when no context was used.

Context	Context applied during	Sentence LM	Fragment LM
none	-	30.8	27.6
left	decoding	27.2	26.6
right	decoding	28.6	27.9
left+right	decoding	**24.6**	26.4
left+right	rescoring	25.7	26.5

Table 3. WER on the corrections using different language models and known context.

5. MERGE MODEL

For spoken corrections, there is potentially useful information in the original result. Successfully leveraging this information is challenging given the original result is known to be erroneous. Here we investigate a novel model that merges information from multiple recognition events.

5.1. Design of model

We designed our model with a number of criteria in mind. First, we wanted results integrated asynchronously and without the model needing deep knowledge about the recognition processes. This allows several recognizers to operate in parallel in a streaming mode and does not require both recognition events to occur simultaneously. It also allows us to fuse information from multiple input modalities which may be using very different recognizers.

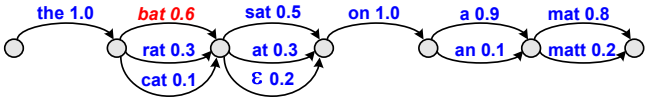

Fig. 3. Confusion network with the word error "bat" shown in red italics. The hypothesis that nothing was said in a cluster is denoted by the ϵ symbol.

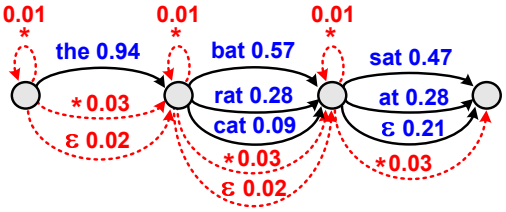

Fig. 4. The portion of the original confusion network that was cut out and softened. The added edges are in dotted red. The * symbol represents wildcard transitions.

We also wanted the model to allow us to easily enforce beliefs we may have about the results being merged. For example, for spoken corrections we have found that users typically speak some amount of correct context around errors [5]. We would like a model that allows us to easily incorporate such information to improve performance.

5.2. Model description

Our model merges results represented as word confusion networks [8]. A confusion network is a time-ordered sequence of clusters where each cluster contains competing words and their probabilities (figure 3). The original confusions networks are softened by adding three extra transitions to every cluster. An epsilon transition is added going to the next cluster without generating a word. A wildcard self-loop allows the current cluster to generate any word while remaining in the same cluster. A wildcard next transition allows generation of any word and proceeds to the next cluster. The probability of each of the added transitions can be varied between different confusion networks that are being combined and can even be varied in different clusters within a confusion network.

Using our model for spoken corrections, the first confusion network is obtained by cutting out a section from the original full sentence confusion network based on the correction's location. The second confusion network is obtained from recognizing the spoken correction. Figure 4 shows an example network cut from the original sentence result. Figure 5 shows an example network from a spoken correction.

The model works by searching for a joint path through the softened confusion networks. We explain the search using the token passing model [9]. A token in our model tracks three pieces of information: the position in each of the confusion networks, the accumulated log probability, and the previous few words of language model context.

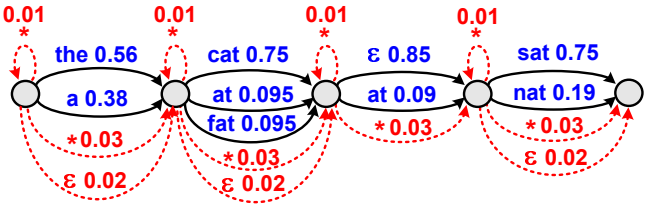

Fig. 5. Correction utterance with added dotted red edges.

Search begins with a token that starts in the first cluster of all networks. A token is finished when it reaches the last cluster in every network. At each step of the search, we select a token from the pool of unfinished tokens. From the selected token's position in each network, 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 network. 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 network can use a wildcard). Second, if multiple networks generate real words, these words must match.

For large networks there are a vast number of possible combinations and an admissible search is intractable. We apply a number of pruning beams to focus our search to only the most promising possibilities.

For all valid combinations, a new token is created at the location specified by the combination of moves. The new token's log probability is calculated as follows:

$$\log P(t_{\text{new}}) = \log P(t_{\text{orig}}) + \sum_{i=1}^{N} w_i \cdot \log P(m_i) + s \cdot \log P(g_{\text{new}})$$

where $P(t_{\text{orig}})$ is the probability of the original token, N is the number of confusion networks being combined, w_i is the weight of the i^{th} confusion network, $P(m_i)$ is the transition probability in the i^{th} network, s is the language model scale factor, and $P(g_{\text{new}})$ is the language model probability of generating the word produced by the token's move.

For example, consider the original recognition in figure 4 and the correction in figure 5. Assume a token is in the second-to-last state in both networks. Table 4 shows the current available moves in each network and the resulting valid combinations of moves. Each of the valid combinations would give rise to a new token. The best combination in this case is for both networks to agree on the word "sat".

5.3. Using knowledge of correct and incorrect words

We extended our model to use knowledge about whether words in the 1-best result were correct or incorrect. Even knowing just which words are incorrect has been shown to significantly improve accuracy [10]. We modify the clusters corresponding to the 1-best words in the network cut from the original recognition. We had three separate sets of wildcard

Original	Correction	Valid combos		Probability
$*_{self}$	$*_{self}$	$*_{self}$	sat	0.0075
sat	sat	$*_{self}$	nat	0.0019
at	nat	sat	$*_{self}$	0.0047
$*_{next}$	$*_{next}$	sat	sat	0.3525
		sat	$*_{next}$	0.0141
		at	$*_{self}$	0.0028
		at	$*_{next}$	0.0084
		$*_{next}$	sat	0.0225
		$*_{next}$	nat	0.0057

Table 4. Examples of available moves for a token in the second-to-last states of figure 4 and 5.

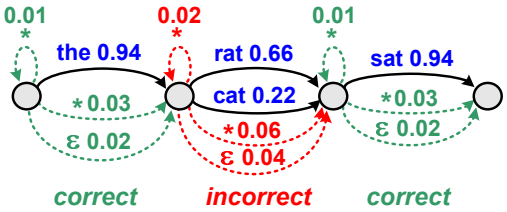

Fig. 6. Cut out portion after modification based on whether words were classified as correct, incorrect or unknown.

and epsilon probabilities for words deemed correct, incorrect, or unknown. In addition, if a word was classified as correct, we eliminated all other words from that cluster. If a word was classified as incorrect, we removed just the best word from that cluster. After removing words, the probabilities of the cluster were renormalized. See figure 6 for an example.

Knowledge about the correctness of words might be obtained from the user. For example, a user might strike through incorrect words with his or her finger. Another option would be to learn from past history whether a user typically provides correct left or right context words. For cases without such user information, we also explored automatically classifying words by using the posterior probabilities in the confusion network. Words with a high probability were classified as correct, words with a low probability were classified as incorrect, and things in between were classified as unknown. The thresholds for this classification were set to optimize performance on our development set.

5.4. Results

A popular way of combining results from multiple recognizers is confusion network combination (CNC) [11]. In CNC, networks are aligned and merged using an edit distance algorithm with a distance metric defined over clusters. We compared our model against CNC as implemented by the SRILM toolkit. Weights can be assigned to each network in the combination. We tuned these weights using our development set.

As shown in table 5, CNC provides a 0.7% absolute reduction in WER. Our merge model does even better, reducing WER by another 0.2%. This is perhaps because our model uses language model constraints during its search. CNC can put clusters next to each other that may result in improbable word sequences. Using our extended model with inferred correct, incorrect and unknown classifications, we obtain a further reduction of 0.4%. If we know with certaintity which words are correct and incorrect, WER drops by another 2.6%.

	WER	Δ
No use of original result	24.6	.
Confusion network combination	23.9	2.8%
Merge model	23.7	3.7%
Merge model, correct+incorrect	23.3	5.3%
Merge model, oracle incorrect	22.7	7.7%
Merge model, oracle correct	21.0	14.6%
Merge model, oracle correct+incorrect	20.7	15.9%

Table 5. Results showing improvement by using information from the original confusion network result.

Correct context	Num utts	Words per utt	Original % correct	Final % correct
none	200	1.9	26.5	57.5
left	187	3.1	26.2	55.6
right	213	3.4	19.2	50.7
both	232	4.4	14.7	49.6
all data	832	3.3	21.3	53.1

Table 6. Table showing how often corrections were recognized completely correct before and after applying all our techniques (excluding oracle knowledge). These results used the setup "Merge model, correct+incorrect" from table 5.

6. DISCUSSION

Spoken corrections are difficult to recognize since the reason the user spoke a correction is that the system failed in the first place. From a user perspective a spoken correction is really only successful if all the words in the correction are correct. Table 6 shows the percent of time corrections where completely correct before and after applying all our techniques (excluding the use of oracle knowledge in our merge model). The probability of a completely correct recognition more than doubled. In the case where the user has provided both left and right context around the error, the probability of completely correct recognition more than tripled. Note that table 6 shows that providing more surrounding context tends to reduce how often corrections are completely correct. This is because as the number of words per utterance increases, it becomes more difficult to recognize every word correctly. However, after applying our improvements, the percent correct is much higher

and more stable regardless of the amount of context used.

We only applied our merge model to the problem of spoken corrections. The model may also be applicable to other scenarios, such as fusing multiple modalities, performing system combination of multiple recognizers, and recognizing repeated utterances. The latter problem has been investigated before (e.g. [12, 6]) and a comparison would be interesting.

We previously found that users tend to change their speaking style noticeably when performing corrections [13]. We suspected substantial gains might be possible by using correction-specific training data instead of WSJ read-speech. However, since we lacked large amounts of correction data, we instead tried adapting our WSJ models using corrections from other speakers obtained in a previous experiment [13]. We were unable to show gains on our development set. We suspect substantially more correction data is required to to make this idea viable. In particular more speakers, more utterances, and more diverse adaptation texts may be required.

7. CONCLUSIONS

In this paper we investigated improving the recognition of spoken corrections. We first showed that a variety of relatively straight-forward techniques such as silence filtering reduced the error rate of spoken corrections from 54.9% to 30.8% (table 1). We showed the importance of using language information specific to corrections. We then described a flexible merge model that combines information from the original recognition result and the subsequent spoken correction. The merge model operates on confusion networks and can easily incorporate prior beliefs about the recognition events, such as which words were likely correct or incorrect. Combining all these techniques, the probability of a spoken correction being recognized completely correct more than doubled (table 6).

The results in this paper extend our previous work on locating the error regions for spoken corrections [5]. Using the techniques in this paper, we are now building a complete system for one-step correction of speech recognition errors. In general, we believe spoken corrections would benefit from more research. As we alluded to in the introduction, the efficiency of a speech recognition system is determined not only by recognition accuracy, but also by how well the system aids users in correcting errors. There has been numerous studies in human-computer interaction on how error correction affects users' speech recognition performance (e.g. [3, 2]). However, there have been relatively fewer studies exploring the challenges associated with recognizing error corrections. We hope this paper will help stimulate progress in this area.

8. ACKNOWLEDGMENTS

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

9. REFERENCES

[1] Keith Vertanen and Per Ola Kristensson, "Parakeet: A continuous speech recognition system for mobile touch-screen devices," in *IUI '09: Proceedings of the* 14^{th} *International Conference on Intelligent User Interfaces*. 2009, pp. 237–246, ACM.

[2] Sharon Oviatt and Robert Van Gent, "Error resolution during multimodal human-computer interaction," in *Proceedings of the International Conference on Spoken Language Processing*, 1996, pp. 204–207.

[3] Christine A. Halverson, Daniel B. Horn, Clare-Marie Karat, and John Karat, "The beauty of errors: Patterns of error correction in desktop speech systems," in *Proceedings of INTERACT*, 1999, pp. 133–140.

[4] Arthur E. McNair and Alex Waibel, "Improving recognizer acceptance through robust, natural speech repair," in *Proceedings of the International Conference on Spoken Language Processing*, 1994.

[5] Keith Vertanen and Per Ola Kristensson, "Automatic selection of recognition errors by respeaking the intended text," in *ASRU '09: IEEE Workshop on Automatic Speech Recognition and Understanding*, December 2009, pp. 130–135.

[6] Geoffrey Zweig, "New methods for the analysis of repeated utterances," in *Proceedings of the International Conference on Spoken Language Processing*, September 2009, pp. 2791–2794.

[7] Bernhard Suhm and Alex Waibel, "Exploiting repair context in interactive error recovery," in *Proceedings of the European Conference on Speech Communication and Technology*, 1997, pp. 1659–1662.

[8] Lidia Mangu, Eric Brill, and Andreas 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.

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

[10] Gregory Yu, "Efficient error correction for speech systems using constrained re-recognition," M.S. thesis, MIT, July 2008.

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

[12] Federico Cesari, Horacio Franco, Gregory Myers, and Harry Bratt, "MUESLI: multiple utterance error correction for a spoken language interface," in *Proceedings of the International Conference on Spoken Language Processing*, September 2008, pp. 199–202.

[13] Keith Vertanen, "Speech and speech recognition during dictation corrections," in *Proceedings of the International Conference on Spoken Language Processing*, September 2006, pp. 1890–1893.