Simple Gesture-based Error Correction Interface for Smartphone Speech Recognition

Yuan Liang[1], *Koji Iwano*[2], *and Koichi Shinoda*[1]

[1]Tokyo Institute of Technology, Japan
[2]Tokyo City University, Japan

yuan@ks.cs.titech.ac.jp, iwano@tcu.ac.jp, shinoda@cs.titech.ac.jp

Abstract

Conventional error correction interfaces for speech recognition require a user to first mark an error region and choose the correct word from a candidate list. Taking the user's effort and the limited user interface available in a smartphone use into account, this operation should be simpler. In this paper, we propose an interface where users mark the error region once, and then the word will be replaced by another candidate. Assuming that the words preceding/succeeding the error region are validated by the user, we search the Web n-grams for long word sequences matched to such a context. The acoustic features of the error region are also utilized to rerank the candidate words. The experimental result proved the effectiveness of our method. 30.2% of the error words were corrected by a single operation.
Index Terms: speech recognition, error correction interface, multi-modal, Web n-gram

1. Introduction

In recent years, speech input interfaces have become popular in smartphone applications [1, 2, 3]. In these interfaces, speech recognition errors are unavoidable, due to the poor performance of automatic speech recognition (ASR) in noisy environments and out-of-vocabulary (OOV) words, and so on [4]. When high quality transcriptions are needed, such as in e-mail applications, users are required to verify and correct the ASR output. In most speech interfaces, when a user finds any error words in the outputs, he/she corrects the error words one by one. First, he/she marks one error word and then either selects the correct word from a candidate list provided by the interface [5, 6, 7, 8], and/or, inputs the correct word by speech, handwriting, or virtual keyboard [9, 10, 11, 12, 13]. Since this error correction process is time-consuming, simpler and more efficient error correction interfaces have been strongly demanded.

Most error correction interfaces utilize a word confusion network (WCN) [14] to provide a candidate list for the error region [6, 7, 8]. Some studies utilized a two pass approach where external resources, such as Web data, are used to increase the hit rate in the candidate list, and demonstrated its effectiveness in some search applications (e.g., [15]). Nevertheless, their performance is still insufficient for our application, error correction for smartphone use. One alternative way to solve this problem is to provide users a specific interface for error correction and to use the constraints implicitly imposed by using such an interface to narrow down the search space for the correct words, more specifically, to use the *user validated* context which is generated during user interaction in error correction procedure [16, 17, 18, 19].

In this paper, we propose a simple gesture-based error correction interface where a user marks the error region once, and then the error region is replaced by the top 1 candidate. Assuming that not only the preceding words but also the succeeding words of the error region are validated by the user, we use such a context to search the Web n-grams data for matched word sequences. We evaluated our method for the case where only one substitution error exists in an utterance as the initial exploration. We proved that our method improved the Word Error Rate (WER) with less user effort than the conventional methods.

This paper is organized as follows. The next section presents the related work. Section 3 describes our proposed error correction interface. Section 4 describes the algorithm for realizing the interface. Section 5 shows our experimental results. Section 6 concludes the paper.

2. Related work

Researches for speech recognition error correction can be divided into two categories: automatic error correction [16, 17, 18, 19] and semi-automatic error correction [5, 6, 7, 8]. In automatic error correction, after a user marks an error region, the wrong word is automatically replaced by the top 1 candidate. In semi-automatic error correction, after a user locates the error region, the system will show a candidates list, and the user then can choose the correct word from a candidate list. Our research focus is on automatic error correction. We also utilized the advantage of external resources such as those on the Web in our error correction procedure.

In automatic error correction, the main problem with this type of error correction is how to design the user interface to use the information generated in the human-machine interaction process, i.e., the user validated context, to generate a more accurate top 1 candidate word. Rodriguez $et.al.$ [16, 17] proposed a computer-assisted speech transcription system, in which every time a user corrects a word, the correction is immediately taken into account to re-evaluate the transcription of the succeeding words of the error word. They proposed a natural assumption: when a user corrects an error word, all the preceding words and the corrected word are correct. They called this information *user validated* prefix. Laurent $et.al.$ [18] used the user validated prefix, higher-order n-gram language model (LM), and caching LM to reorder the WCN. Wang $et.al.$ [19] proposed a multimodal error correction interface, where users use pen gestures on a touch screen interface. They assumed that one preceding and one succeeding word of the error region are the user validated contexts.

Web text data as an external resource has been widely used in rescoring the first pass speech recognition results [20], in

Error type and gesture name	Finger gestures	Number of correct words in error region	Number of gesture operations
Substitution error	正解が存在来る seikai ga sonzai kuru	1	1
underline	正解が損害来る seikai ga songai kuru	≥2	1
Insertion error	結果をお報告する kekka o o hokoku suru	0	1
strikethrough	結果をお御報告する kekka o o o hokoku suru	0	1
Deletion error	友人の関係を yujin no kankei o	1	1
vertical line	友人関係 を yujin kankei o	≥2	1

Figure 1: *Gestures for correcting errors.*

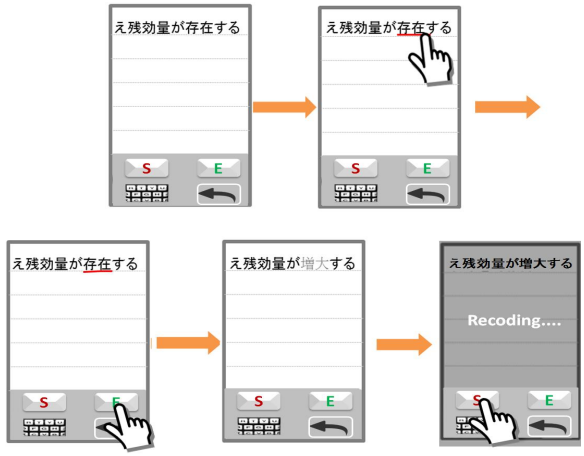

Figure 2: *Error correction procedure for a substitution error.*

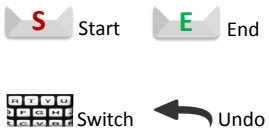

Figure 3: *Buttons.*

augmenting the training data for adapting LM [21, 22], and in correcting recognition errors [23, 24]. For example Nishizaki *et.al.* [15] tried to use Web information in spoken documents. Their system includes an automatic error correction procedure without the user's interaction. Their system used the correctly recognized proper nouns as a query to search the Google search engine, and substituted the error word by the best candidate.

3. Interface

We are working on Japanese speech recognition. For each user utterance our interface first displays the 1-best hypothesis, and asks the user to correct errors from left to right. Then the user uses the finger gestures "underline", "strikethrough" and "vertical line" to mark the error words. Different finger gestures mean different error types as shown in Figure 1. Figure 2 shows the error correction procedure for a substitution error. The system waits to correct an error region until the user marks the next error region or until the user pushes the "End" button (Figure 3). If the word is still not correct after this step, the user selects the error word again. This time the system uses the second best candidate word to replace the error word and at the same time shows a n-best candidate list under that error region. For the errors that cannot be corrected by choosing from the candidate list, the user can push the "Switch" button (Figure 3) to edit the correct word by handwriting, or by virtual keyboard.

Figure 1 shows the three error types in the ASR output, and the corresponding gestures. For example, if there is one substitution error, the user underlines the error word. If the error region includes more than one word, the user uses one long underline to mark the error region. The user deletes insertion errors directly by using a strikethrough. For deletion errors, the user uses a vertical line to mark the place to insert a new word. We choose these three *one-stroke gestures* in order to minimize the user effort. In this study, we focus on error correction for those sentences with only one substitution error, as the initial exploration.

4. Algorithm

4.1. Outline

Our goal is to use the user validated context, preceding and/or succeeding words, of the error region to genearte the most probable candidate word w in the set of all candidate words W for the error region. We search for the w which has the largest probability given the acoustic features X and the user validated context W_c:

$$\begin{aligned}\hat{w} &= \arg\max_W P(W|X, W_c) \\ &= \arg\max_W P(X|W, W_c)P(W|W_c)P(W_c). \end{aligned} \quad (1)$$

We make a realistic assumption that the probability of X given W does not depend on the context W_c, so we rewirte this equation as:

$$\hat{w} = \arg\max_W P(X|W)P(W|W_c)^{\alpha}, \quad (2)$$

where we introduce a *language model (LM) weight* α between the acoustic model and the language model. The following is an outline of the error correction algorithm.

1. Generate the 1-best result, and get the time information of each word in 1-best. Then, the user starts marking the misrecognized words in the ASR output interface.
2. The system waits to correct an error word until the user marks the next error region or until the user pushes "End" button. For each error word, repeat the following steps:
   (a) Extract the words preceding/succeeding the error region, and search higher-order n-grams which match this context in the Web n-grams data
   (b) Calculate the LM score of each candidate word
   (c) Extract speech features of the error region, and calculate the AM score for each candidate word
   (d) For each candidate word, calculate the weighted sum of AM and LM scores
   (e) Rank the candidates using the scores in (d)

Algorithm 1 Backoff search

for ($n = 7;n - -;n > 1$) **do**
  for ($i = n - 1;i - -;i \geq 0$) **do**
    use (w_{p-i}, ..., w_{p-1}, w_*, w_{p+1}, ..., w_{p+n-i}) as searching queries to search the n-grams data; store the matched word sequences and their counts in the memory
  end for
  if in the stored data, there is at least one word sequence **then**
    Stop search
  end if
end for
n : "n" in n-gram
p : the error word position in the n-word sequence
w_* : any word

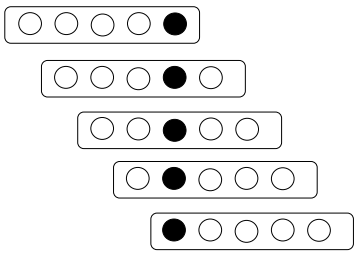

Figure 4: *All search queries on the 5-grams. Black circle is the error word, and white circle is the user validated word.*

In the following, we explain Step 2-(b) and 2-(c) more in detail.

4.2. LM score calculation

We used high-order Web n-grams data, which consists of word n-grams and their observed frequency counts collected from Web text. Algorithm 1 shows our backoff search algorithm. Every time we construct a query to search for the word sequences in the Web n-grams data, we start from the longest query with length n. In each n-gram, we use all the possible queries. Figure 4 shows all the search queries for the 5-grams. If we can't find any candidates, we search again use the queries with length $n - 1$. We continue this process until we find at least one matched candidate word.

After we get the candidate words from the Web n-grams data, we calculate the LM probability of each candidate word. The probability of candidate word can be estimated from its count in the Web n-gram, which is equal to the count of that word sequence divided by the total number of counts of all the matched word sequences.

Let w_m be any candidate word obtained from the Web n-gram data, W_c be the context of the error word, Count(W_c, w_m) be the number of occurrences of (W_c, w_m) in the Web n-grams data, Count(W_c) be the total number of occurrences of all the matched word sequences, and M be the number of candidate words obtained from the Web n-grams data.

$$\text{Count}(W_c) = \sum_{m=1}^{M} \text{Count}(W_c, w_m). \quad (3)$$

We can estimate the probability of each candidate word as:

$$p(w_m|W_c) = \frac{\text{Count}(W_c, w_m)}{\text{Count}(W_c)}. \quad (4)$$

4.3. AM score calculation

In Eq. 2 the acoustic probability $P(X|W)$ corresponds to the probability attributed to the acoustic model. We use the acoustic model and the speech features of the error word to calculate the acoustic score of each candidate word.

In order to get the speech features of the error word, we utilize the error word time information. We do decoding only on the error region to obtain the AM score for each candidate word. For getting the phone sequence of each candidate word W, we utilized the grapheme-to-phoneme conversion toolkit.

5. Experiments

First, we compared our LM-based method with acoustic information and without it. Then, we compared our LM-based method with the conventional WCN based method. Finally we compared our Web-scale LM with "*in-domain*" LM.

5.1. Experimental setup

We evaluated the proposed method using speech data from academic and extemporaneous lectures in the Corpus of Spontaneous Japanese (CSJ) [25]. The number of lectures is 2701, and the total length of the data is 530 hours. We randomly divided this corpus into two subsets; one subset contains 1350 lectures, and the other subset contains 1351 lectures. We evaluated our method by cross-validation using them. The triphone acoustic model and the trigram language model were constructed using a training set. The T^3 decoder [26] was used for recognition. The average word recognition accuracy is 65.2%.

There were 547,234 utterances including errors, and among them 69,764 selected included only one substitution error and did not include any deletion and insertion errors. Among them, we further selected 48,238 sentences in which the correct word corresponding to each misrecognized word was not a filler word.

We used Google Japanese Web n-grams data [27]. It consists of Japanese word n-grams and their observed frequency counts generated from over 255 billion tokens of text. The length of the n-grams ranges from unigrams to 7-grams. The n-grams were extracted from publicly accessible web pages that were crawled by Google in July 2007. This data set contains only n-grams that appear at least 20 times in the processed sentences.

In order to generate the WCN based candidate list, we employed the SRILM toolkit [28].

For "*In-domain*" LM, we used CSJ text data to generate the word n-grams and their observed frequency counts. In this study, we used Chasen Morphological Analyzer [29] to convert from Japanese characters to monophones.

5.2. Results

The results in Table 1 proved the effectiveness of our LM-based (Web-scale LM based) method in error correction. 11.4% of the error words can be recovered by using the top 1 candidate. By combining AM and LM, the percentage of error words that can be recovered by the top 1 increased to 25.5%. We also observed the importance of the acoustic information of the error region, not only for out-domain data (Web data) but also for in-domain

Table 1: *The ratio of the cases where the correct word is included in the N-best candidate list (%).*

N	1	5	10
WCN based	23.6	41.4	46.1
In-domain LM based (w/o AM)	9.6	16.3	18.4
In-domain LM based	17.6	21.2	21.9
Web-scale LM based (w/o AM)	11.4	21.2	25.6
Web-scale LM based	25.5	32.2	34.2

Table 2: *The ratio of the cases where the correct word is included in the N-best candidate list (%).*

N	1	5	10
WCN based	23.6	41.4	46.1
Web-scale LM based	25.5	32.2	34.2
WCN + Web-scale LM based	30.2	53.3	58.5

data. The percentage of the top 1 candidate words that were correct increased from 9.6% to 17.6% for in-domain data.

By comparing the Web-scale LM with the in-domain LM, we confirmed the effectiveness of using Web higher-order n-grams in error correction.

As we can see from the Table 1 the performance of "top 5" and "top 10" results by using our Web-scale LM based method was worse than WCN based method. We checked the overlap of the top 1 result between WCN based method and Web-scale LM based method. We found the overlap was only around 33%. It suggests that the WCN based method and Web-scale LM based method are complementary to each other. Motivated by this, we combined the two methods by linear interpolation of their scores. We used two-fold cross-validation to estimate the interpolation weights; weight 0.4 for the WCN based method and weight 0.6 for Web-scale LM based. We found the combined method got the best result not only in "top 1" result, but also in "top 5" and "top 10" result.

5.3. User load analysis

We compared user load using our WCN + Web-scale LM with the conventional WCN. The following analysis was done under the assumption that system correctly identified the error region. Considering the small size of smartphone displays, we limited the length of the candidate list to 5.

We divided the errors into four types: Type 1 error, which can be automatically corrected by the top 1 candidate word; Type 2 error, which can be automatically corrected by the top 2 candidate word; Type 3 error, which can be corrected from a 5-best candidate list; Type 4 error, which can be corrected through other input modality, like keyboard, etc.

For our WCN + Web-scale LM based interface: for the case when automatic correction failed, after the user marks the error word again, the system will use the top 2 candidate word to replace the error word and show a 5-best candidates list which includes top 3 to top 7 candidates. Table 3 lists the number of operations a user needs to correct these 4 type errors in our interface.

For the conventional WCN based interface: the user marks the error word first, the system will provide the user a 5-best candidate list which includes top 1 to top 5 candidates. There is only two error types, Type 3 and Type 4. Table 4 lists the number of operations a user needs to correct these 2 type errors in the conventional interface.

Table 3: *The number of user operations for the 4 types of error in our interface*

Error type	Number of operations
1	0
2	1
3	2
4	≥ 1

Table 4: *The number of user operations for the 4 types of error in the conventional WCN based interface*

Error type	Number of operations
1	-
2	-
3	1
4	≥ 1

Table 5: *The number of strokes users need to correct 100 error words in different user interfaces.*

Operations	Conventional interface	Our interface
Mark errors	100	100
Correct Type 1 errors	-	0
Correct Type 2 errors	-	12
Correct Type 3 errors	41	27
Correct Type 4 errors	≥ 59	≥ 44
Push the "End" button	-	1
Total	≥ 200	≥ 184

Table 5 analyzes user load (the number of strokes/touches) for correcting errors where we suppose the user must correct 100 error words. We used the recognition results of all the 48,238 errors utterances to collect the statistics. For our method, we assume the user marks all the 100 error words, then pushes the "End" button. In our experiment the ratio of the cases where the correct word is included in the top 2 candidates is 12.1%, and is included in the top 3 to top 7 candidates is 13.7%.

In Table 5, even though in some stage our interface requires more user effort, on average we saved users' effort by 8% from the conventional WCN based user interface.

6. Conclusions

We have proposed a Web-scale LM based simple gesture-based error correction interface where users mark the error word once, and then the word will be replaced by another candidate. We used user validated information to search the Web n-grams for long word sequences. The acoustic features of the error region are also utilized to rerank the candidate words. The result proved the effectiveness of our method using combined WCN + Web-scale LM. 30.2% error words were recovered by using top 1 result. 8% of user effort was saved by using our interface rather the conventional error correction interfaces.

In this paper we only tested our methods on utterances with one substitution error. We plan to use the same idea in more general situations, when the error region includes more than one error word and deletion and insertion errors appear.

7. References

[1] K. Vertanen, "Speech and speech recognition during dictation corrections", in Proc. Interspeech, 2006.

[2] K. Vertanen, and P. O. Kristensson, "Parakeet: A continuous speech recognition system for mobile touch-screen devices", in Proc. the fourteenth ACM International Conference on Intelligent User Interfaces, 2009.

[3] K. Vertanen, and P. O. Kristensson, "Intelligently aiding human-guided correction of speech recognition", in Proc. AAAI, pp. 1698-1701, 2010.

[4] P. O. Kristensson, "Five challenges for intelligent text entry methods", in Artificial Intelligence Magazine, vol. 30, no. 4, pp. 85-94, 2009.

[5] K. Larson, and D. Mowatt, "Speech error correction: The story of the alternates list", International Jounal of Speech Technology, vol. 6, pp. 183-194, 2003.

[6] J. Sturm and L. Boves, "Effective error recovery strategies for multimodal form-filling applications", Speech communication, vol. 45, no. 3, pp. 289-303, 2005.

[7] J. Ogata, and M. Goto, "Speech Repair: Quick error correction just by using selection operation for speech input interface", in Proc. Interspeech, pp. 133-136, 2005.

[8] M. Burke, and B. Amento and P. L. Isenhour, "Error correction of voicemail transcripts in SCANMail", in Proc. CHI, pp. 339-348, 2006.

[9] J. R. Lewis, "Effect of error correction strategy on speech dictation throughput", in Proc. Human Factors & Ergonomics Soc. 43rd Annual Meeting, pp. 457-461, 1999.

[10] B. Suhm, B. Myers, and A. Waibel, "Multimodal error correction for speech user interfaces", ACM Transaction on Computer-Human Interaction, vol. 8, no. 1, pp. 60-98, 2001.

[11] D. H. Daines, and A. I. Rudnicky, "Interactive ASR error correction for touch screen devices", in Proc. ACL, pp. 17-19, 2008.

[12] K. Shinoda, Y. Watanabe, K. Iwata, Y. Liang, R. Nakagawa, S. Furui, "Semi-synchronous speech and pen input for mobile user interfaces", in Speech Communication, vol. 53, no. 3, pp. 283-291, 2011.

[13] J. Choi, K. Kim, S. Lee, S. Kim, D. Lee, I. Lee, G. G. Lee, "Seamless error correction interface for voice word processor", in Proc. ICASSP, pp. 4973-4976, 2012.

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

[15] H. Nishizaki, and Y. Sekiguchi, "Word Error Correction of Continuous Speech Recognition Using Web Documents for Spoken Document Indexing", in Proc. the Computer Processing of Oriental Languages (ICCPOL), 2006.

[16] L. Rodriguez, F. Casacuberta, and E. Vidal , "Computer assisted transcription of speech", in Proc. the Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), 2007.

[17] E. Vidal, L. Rodriguez, F. Casacuberta, and I. G. Varea, "Interactive pattern recognition", in Proc. MLMI 2007, pp. 60-71, 2007.

[18] A. Laurent, S. Meignier, T. Merlin, and P. Delglise, "Computer-assisted transcription of speech based on confusion network reordering", in Proc. ICASSP, pp. 4884-4887, 2011.

[19] L. Wang, T. Hu, P. Liu and F. Soong, "Efficient handwriting correction of speech recognition errors with Temlate Constrained Posterior (TCP)", in Proc. Interspeech, 2008.

[20] X. Zhu, and R. Rosenfeld, "Improving trigram language modeling with the World Wide Web", in Proc. ICASSP, pp. 533-536, 2001.

[21] G. Lecorve, J. Dines, T. Hain, P. Motlicek, "Supervised and unsupervised Web-based language model domain adaptation", in Proc. Interspeech, 2012.

[22] T. Schlippe, L. Gren, N. T. Vu, and T. Schultz, "Unsupervised Language Model Adaptation for Automatic Speech Recognition of BroadcastNews using Web 2.0", in Proc. Interspeech, 2013.

[23] C. Parada, A. Sethy, M. Dredze, F. Jelinek, "A spoken term detection framwork for recovering OOV words using the web", in Proc. Interspeech, 2010.

[24] K. Yoshino, S. Mori, and T. Kawahara, "Incorporating Semantic Information to selecion of Web Texts for language model of spoken dialogue system", in Proc. ICASSP, 2013.

[25] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous Speech Corpus of Japanese", in Proc. the Second International Conference on Language Resources and Evaluation (LREC2000), vol. 2, pp. 947-952, 2000.

[26] P. Dixon, D. Caseiro, T. Oonishi, and S. Furui, "The Titech Large Vocabulary WFST Speech Recognition System", in the Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 1301-1304, 2007.

[27] T. Kudo and H. Kazawa. "Japanese Web N-gram Version 1", Linguistic Data Consortium, Philadelphia, 2009.

[28] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "SRILM at sixteen: update and outlook", in Proc. IEEE workshop on speech recognition and understanding (ASRU), 2011.

[29] Y. Matsumoto, K. Takaoka, and M. Asahara, "ChaSen morphological analysis version 2.4.0 User's manual", Nara Institute of Science and Technology (NAIST), 2007.