

Error Correction Using Long Context Match for Smartphone Speech Recognition

YUAN LIANG^{1,a)} KOJI IWANO² KOICHI SHINODA¹

Abstract: Correcting speech recognition errors on a smartphone is a challenging task that requires a lot of user effort. To reduce this user effort, we previously proposed an error correction method based on Long Context Match (LCM) with higher-order N-grams, which we combined with a simple gesture-based user interface. However, LCM was used when there was only one substitution error exists in an utterance. In this paper, we extended LCM to be used when the contexts of the error region contain errors. We examined the error regions which contain only one error and confirmed the effectiveness of the extended LCM-based method.

Keywords: speech recognition, error correction interface, Web n-gram, candidate list, context

1. Introduction

In recent years, speech input interfaces have become popular in smartphone applications (e.g., [1]). In these interfaces, speech recognition errors are unavoidable. When high quality transcriptions are needed, such as in e-mail applications, users are required to verify the ASR output and correct errors, this process is time-consuming. Therefore, simpler user interface and more efficient error correction methods have been strongly demanded.

Most error correction interfaces utilize a word confusion network (WCN) [2] to provide a candidate list for an error word [3], [4]. Some studies utilized a two pass approach where external resources, such as Web corpora, are used to increase the hit rate in the candidate list, and demonstrated its effectiveness in some search applications (e.g., [5]). Nevertheless, their performance is still insufficient for our application, error correction for dictation applications on smartphone.

One solution to solve this problem is to provide users a specific interface for error correction and to use the *user validated context* which is generated during user system interaction in error correction procedure. Rodriguez *et.al.* [6] 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 corrected word. They assumed that, when a user corrects an error word, all the preceding words and the corrected word are correct. They called this information *user validated prefix*. In our previous work [7], we proposed an error correction method based on Long Context Match (LCM). In LCM, we utilized not only *user validated prefix*, but also *user validated suffix*. We also designed a new simple gesture based interface to realize this method. But it was evaluated only when one substitution error exist in an utter-

ance. In our previous work [8], we extended LCM based method to be used in more general situations, when the preceding words and the succeeding words contain errors. It corrects not only substitution errors, but also deletion errors and insertion errors.

Compared to our previous work [8], in this paper, we explain the user system interaction procedure and theoretical basis more in detail, and also add a detailed analysis of the performance of our LCM based method in the situation when the preceding words and the succeeding words do not contain errors.

2. Interface

For each user utterance our interface first displays the 1-best hypothesis. We choose three *simple one-stroke gestures* in order to minimize user effort. Finger gesture “underlines” are used to mark substitution errors, “strikethroughs” are used to mark insertion errors, and “vertical lines” are used to mark deletion errors (**Fig. 1**). **Fig. 2** shows, when an error region contains more than one error word, we ask users to mark the errors using our proposed three gestures based on the types of errors, so for each error region the number of correct words equals to the number of “underlines” plus the number of “vertical lines”.

Fig. 3 shows an example of an error correction procedure. For each error region, the system substitutes the misrecognized words with top 1 candidate, and also shows a 5-best candidate list under the error region. For the errors that cannot be automatically corrected by top 1 candidate or cannot be corrected by choosing from a candidate list, a user can push the “Switch” button (**Fig. 4**) to edit them by virtual keyboard [3], [9], handwriting [9], or autocomplete [3]. Insertion errors can be ignored in this study, because we assume users could directly delete them by using gesture “strikethrough”.

3. Overview of the user system interaction

Fig. 5 presents an overview of the user system interaction. The system processes user’s speech and displays the text. If the user

¹ Tokyo Institute of Technology

² Tokyo City University

^{a)} yuan@ks.cs.titech.ac.jp

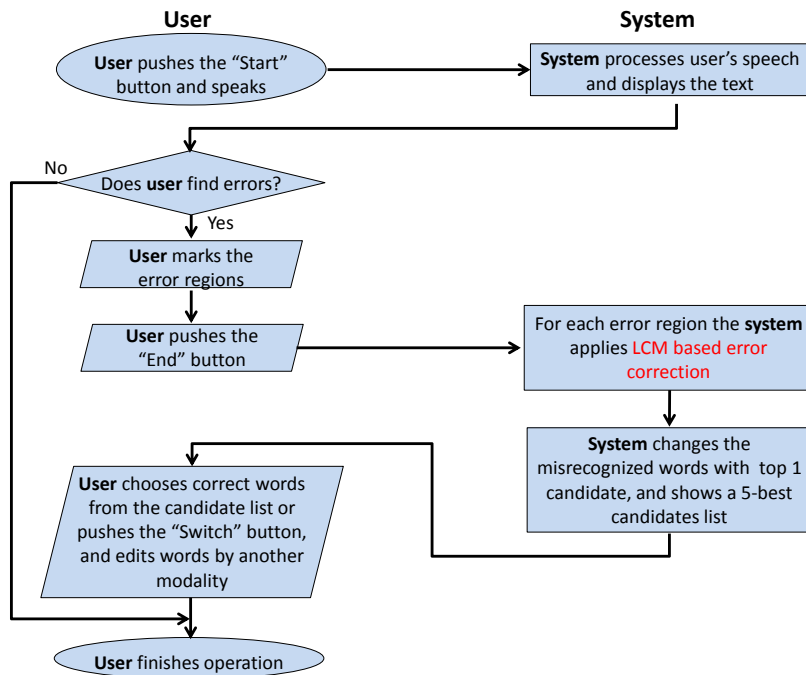


Fig. 5 Overview of the user system interaction.



| Error type and gesture name | Finger gestures |
|---|-----------------------------------|
| Substitution error  | 正解が存在する seikai ga sonzai kuru |
| Insertion error  | 結果をお報告する kekka o o hokoku suru |
| Deletion error  | 友人の関係を yujin no kankai o |

Fig. 1 Gestures for correcting errors.

| User Gestures |
|---|
| COR: 彼らは七から十一か月児を対象として REC: 彼らは七から九か月児を対象として |
| COR: 言語刺激は一応私達が馴染みのある刺激 REC: 言語刺激 / 愛知私達が馴染み / なり刺激 |
| COR: えーこの研究を通して私がやりましたことというのは REC: えーこの研究はどうして私がやりましたことというのは |

Number of correct words in each error region =
number of "underlines" + number of "vertical lines"

Fig. 2 When an error region contains multiple errors.

finds any errors, he/she starts marking and correcting them. After marking all error regions, the user pushes the "End" button. For each error region, the system extracts its prefix and suffix, and applies LCM based error correction. Then the system substitutes the misrecognized words with the top 1 candidate, and shows a 5-best candidate list. If the top 1 candidate is correct, the user does not need to do anything. Otherwise the user either chooses the correct words from the candidate list or pushes the "Switch" button, and edits the words by keyboard, handwriting, or autocomplete. Section 3 and Section 4 explain the detail of LCM process.

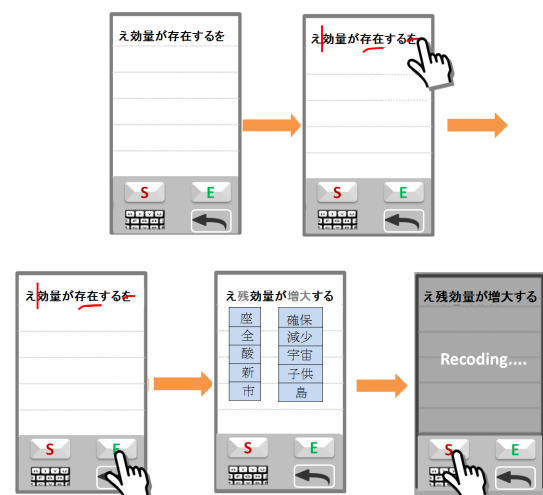


Fig. 3 Error correction procedure for correcting a deletion error, a substitution error and an insertion error.

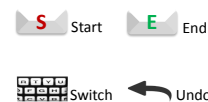


Fig. 4 Buttons.

4. LCM based error correction

We have already proposed an error correction method using long context match (LCM) in [7]. The idea was to use not only the user validated preceding words, but also the user validated succeeding words, of the error region to generate the most probable candidate W_e in the set of all candidate words W_E for each error region.

4.1 Theoretical basis

Let

$$W = \underbrace{w_1, \dots, w_{j-1}}_{W_p} \underbrace{w_j, w_{j+1}, \dots, w_k}_{W_e} \underbrace{w_{k+1}, \dots, w_T}_{W_s},$$

where W means the set of all possible word sequences corresponding to the sequence X of acoustic observations, and T means the number of words in the word sequences. In our case, W is divided into three fragments: a prefix W_p , an error region W_e , and a suffix W_s . W_p contains user validated context W_{cp} , and W_s contains user validated context W_{cs} . W_e is the user validated preceding and succeeding words, which is an integration of W_{cp} and W_{cs} .

We search for the W_e which has the largest probability given the acoustic features X and the user validated context W_c :

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

In order to solve Eq. (1), the signal X can be split into three fragments X_p, X_e, X_s , considering the boundary of the fragment is known. We rewrite equation (1) as:

$$\begin{aligned} \hat{W}_e &= \arg \max_{W_e} P(X_p, X_e, X_s|W_{cp}, W_e, W_{cs})P(W_e|W_c)P(W_c) \\ &= \arg \max_{W_e} P(X_p, X_e, X_s|W_{cp}, W_e, W_{cs})P(W_e|W_c) \end{aligned} \quad (2)$$

We can make an assumption that the probability of X_p given W_{cp} does not depend on the error region and suffix, the probability of X_e given W_e does not depend on the prefix and suffix, and the probability of X_s given W_{cs} does not depend on the prefix and error region. We rewrite equation (2) as:

$$\begin{aligned} \hat{W}_e &= \arg \max_{W_e} P(X_p|W_{cp})P(X_e|W_e)P(X_s|W_{cs})P(W_e|W_c) \\ &= \arg \max_{W_e} P(X_e|W_e)P(W_e|W_c) \\ &= \arg \max_{W_e} P(X_e|W_e)P(W_e|W_c)^\alpha, \end{aligned} \quad (3)$$

where we introduce a *language model (LM) weight* α between the acoustic model (AM) and the language model (LM).

4.2 Algorithms

We used high-order n -grams; the largest n is 7. Our target is to find word sequences which match to the search queries with candidate words, except the error word itself. Search queries are made up by user validated prefix and suffix. The number of preceding and succeeding words of the error region determines the longest length of queries, and also determines the number of possible queries we could use in each n -gram. A backoff search algorithm starts from the longest query with length " n " until length " 2 ". We search the n -length word sequences which match to all the possible queries in n -grams. If we can't find any candidate, we search again using $(n-1)$ -length queries in $(n-1)$ -grams. We continue this process until we find at least one candidate in the current n -gram.

When we get at least one candidate word from n -grams in a certain n , we stop search. If there is only one candidate, the system directly outputs it. Otherwise, the system calculates the LM and AM score of each candidate word as follows.

The LM score of candidates can be estimated from their counts in the n -grams, which is equal to the counts of that word sequence divided by the total number of counts of all the matched word sequences. For candidates we get from different queries, we set them equal weight. Let W_e be any candidate word or candidate sequence obtained from the n -grams data, W_c be the context of the error region, $\text{Count}(W_c, W_e)$ be the number of occurrences of (W_c, W_e) in the n -grams data, $\text{Count}(W_c)$ be the total number of occurrences of all the matched word sequences, and M be the number of candidates obtained from the n -grams data.

$$\text{Count}(W_c) = \sum_{e=1}^M \text{Count}(W_c, W_e). \quad (4)$$

We can estimate the LM probability of each candidate as:

$$P(W_e|W_c) = \frac{\text{Count}(W_c, W_e)}{\text{Count}(W_c)}. \quad (5)$$

We also calculate the AM probability of each candidate word, $P(X_e|W_e)$, by using the speech features of the error region to decode only on the error region whose time boundary information has already given. Finally, we calculate the weighted sum of LM and AM scores for each candidate word, and rank all the candidate words based on their combined LM and AM scores.

5. Generalized LCM based error correction

In our paper [7], the LCM based method was evaluated only in the *ideal situation*, where there is only one substitution error exist in each test utterance. In our previous study [8], we extended the LCM based method to a more *general situation*, where the preceding words or the succeeding words contain multiple errors, and each error region may contain more than one error words. Also, the extended method [8] can deal with not only substitution but also deletion errors by implementing the following "Error type dependent error correction method".

5.1 Make search queries in general situation

The difference of the general situation from the ideal situation is that the words around the error region contain at least one error. By using regular expressions, a wildcard word " $.*$ " is used to represent any word. If one ASR output utterance contains 9 words, the number 2, 3, 5, 6 word corresponds to four substitution errors. Number 2 and 3 errors are in one error region (Region 1), number 5 and 6 errors are in one error region (Region 2). For Region 2, the longest length of queries is 7, the shortest length of queries is 3, and the possible queries we could use in each n -gram as shown in Table 1. Here " w_* " is any candidate word, and " w_n " is the number n word in this utterance.

5.2 Error type dependent error correction

Fig. 6 shows the outline of the error type dependent LCM based error correction method.

For both substitution errors and deletion errors, the ways of

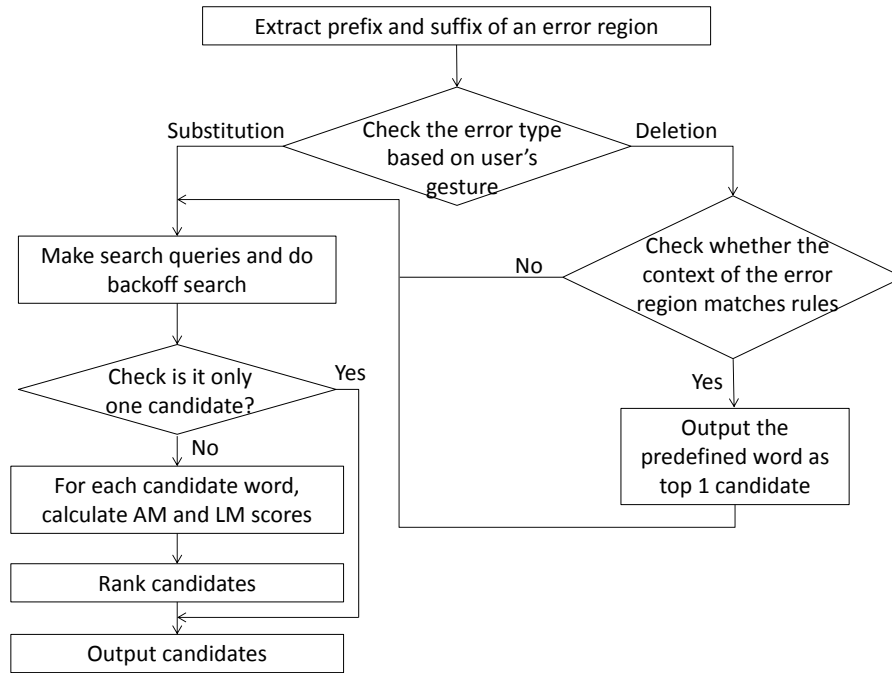


Fig. 6 The outline of the error type dependent error correction for one error region.

Table 1 All the possible search queries in each n-gram.

| | Region 2 |
|--------|---|
| | $w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9$ |
| 7-gram | $w_1, .*, .*, w_4, w_5, w_6, w_7, w_8, w_9$ $.*, .*, w_4, w_5, w_6, w_7, w_8, w_9$ $.*, w_4, w_5, w_6, w_7, w_8, w_9$ |
| 6-gram | $w_1, .*, .*, w_4, w_5, w_6, w_7, w_8, w_9$ $.*, .*, w_4, w_5, w_6, w_7, w_8, w_9$ $.*, w_4, w_5, w_6, w_7, w_8, w_9$ $w_4, w_5, w_6, w_7, w_8, w_9$ |
| 5-gram | $.*, .*, w_4, w_5, w_6, w_7, w_8, w_9$ $.*, w_4, w_5, w_6, w_7, w_8, w_9$ $w_4, w_5, w_6, w_7, w_8, w_9$ w_5, w_6, w_7, w_8, w_9 |
| 4-gram | $.*, w_4, w_5, w_6, w_7, w_8, w_9$ $w_4, w_5, w_6, w_7, w_8, w_9$ w_5, w_6, w_7, w_8, w_9 |
| 3-gram | $w_4, w_5, w_6, w_7, w_8, w_9$ w_5, w_6, w_7, w_8, w_9 |

calculating LM scores are the same. We use the method as in 5.1 to make queries. The backoff search and LM score calculation is the same procedure as in Section 4.2.

For substitution errors, we follow the same procedure as in Section 4.2 to calculate AM scores. For deletion errors, since the time boundary information for the speech features of the deleted words is not given, we utilize the time boundary information of one preceding word and one succeeding word of the error region. We decode for three words, a preceding word, a candidate of the deleted word and a following word. Then we get the AM score of each candidate word.

We investigated the relation between the deleted words and the surrounding words in our preliminary experiments. We analyzed three features: word, phoneme, and part-of-speech. We found the relation between some deleted words and the last phoneme (and part-of speech) of a preceding word or the first phoneme (and part-of speech) of a succeeding word is relatively strong. For example, if the last phoneme of the preceding word is “o” or “o:”,

and the part-of-speech of that word is a noun, 75% the deleted word is a postpositional particle “を /wo/”. Based on these findings, we propose a language-dependent rule-based method for the deletion error case. If there is a deletion error, the system will firstly check the phonemes and part-of-speech of the preceding word or the succeeding word. If this information matches our pre-determined rules, the system will directly predict a deleted word as top 1 candidate. At the same time, the system will also use our LCM based method to generate a 5-best candidate list under the error region.

6. Experiments

6.1 Experimental setup

We are working on Japanese speech recognition. We evaluated the proposed method using speech data from academic and extemporaneous lectures in the Corpus of Spontaneous Japanese (CSJ) [10]. The number of lectures is 2701, and the total length of the data is 530 hours. We evaluated our method by cross-validation. We randomly divided this corpus into two sets: one set contains 1350 lectures, and the other set contains 1351 lectures. The triphone acoustic model and the trigram language model were constructed. The T^3 decoder [11] was used for recognition. The average word recognition accuracy is 65.2%. For these two sets there are in total 1,026,543 error regions. 31.7% error regions contain only one substitution error, 10.4% error regions contain only one deletion error, 3.0% error regions contain only one insertion error.

Among speech utterances which include more than one error region, we randomly chose 5000 utterances which include error regions with one substitution error, and 5000 utterances which include error regions with one deletion error.

We used Google Japanese Web n -grams [12]. It consists of Japanese word n -grams and their observed frequency counts gen-

erated from over 255 billion tokens of text. 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 utterances. Web n -grams doesn't provide the pronunciation for each word. For obtaining the phone sequence of each candidate word W , we utilize grapheme-to-phoneme conversion tool Chasen Morphological Analyzer [13] to convert from Japanese characters to monophones. For "In-domain" LCM, we used CSJ text data to generate the word n -grams and their observed frequency counts. We also evaluate the performance when using a word confusion network (WCN) based error correction method. WCN [2] is a compact representation of multiple aligned ASR hypotheses. It is obtained by converting a word lattice. Each competing word in an aligned segment has a posterior probability, which can be used as a confidence score of that word, and the scores of all words in an aligned segment are summed into one. The LCM based method utilizes the user validated context while WCN based method doesn't. This fact suggests that the WCN based method and LCM based method are complementary to each other. Motivated by this, we combine these two methods by linear interpolation of their scores. In order to generate the WCN based candidate list, we employed the SRILM toolkit [14].

6.2 Experimental results

Table 2 shows the results for correcting substitution errors when their contexts contain errors. As we can see in Table 2, the "Web-scale LCM" got a little bit worse result than WCN. We found the combined WCN and LCM still got better results. We used two-fold cross-validation to estimate the interpolation weights; 0.4 for the WCN and 0.6 for the LCM. For correcting substitution errors 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 data. So in the substitution case among all the individual LCM based methods, "Web-scale LCM" got best results.

Table 3 shows the results for correcting deletion errors when their contexts contain errors. We found "In-domain LCM" obtained better results than "Web-scale LCM", and "In-domain LCM (w/o AM)" got better results than "In-domain LCM". Compared to the substitution errors, the length of the deleted words is relatively short, the quality of speech features of the deleted word is relatively low, and the deleted words are more common in In-domain data. This may be the reason why in-domain n -grams are more efficient for our LCM based method. We also proved the effectiveness of the rule-based method for correcting deletion errors. By using "In-domain LCM (w/o AM) + Rule", the error correction rate increased to 37.7. So in the deletion case among all the individual LCM based methods, "In-domain LCM (w/o AM) + Rule" got best results.

Based on the results we obtained from Table 2 and Table 3, we realized that the best method for substitution errors was "Web-scale LCM", and the best method for deletion errors was "In-domain LCM (w/o AM) + Rule". By using the combination of the best methods, we make an error correction method based on the "Web-scale LCM for Sub" + "In-domain LCM (w/o AM) +

Table 2 For correcting substitution errors, % occurrence of the correct word in the N -best candidate list.

| N | 1 | 5 | 10 |
|------------------------|-------------|-------------|-------------|
| WCN | 20.7 | 34.7 | 38.0 |
| In-domain LCM (w/o AM) | 8.3 | 18.2 | 21.1 |
| In-domain LCM | 16.9 | 22.2 | 23.1 |
| Web-scale LCM (w/o AM) | 8.7 | 18.9 | 23.7 |
| Web-scale LCM | 19.7 | 25.9 | 27.8 |
| WCN + In-domain LCM | 20.1 | 42.7 | 47.9 |
| WCN + Web-scale LCM | 22.8 | 45.5 | 50.8 |

Table 3 For correcting deletion errors, % occurrence of the correct word in the N -best candidate list.

| N | 1 | 5 | 10 |
|-------------------------------------|-------------|-------------|-------------|
| WCN | 23.1 | 48.4 | 53.2 |
| In-domain LCM (w/o AM) | 31.9 | 49.0 | 53.4 |
| In-domain LCM | 26.0 | 42.1 | 47.8 |
| Web-scale LCM (w/o AM) | 26.6 | 44.4 | 50.0 |
| Web-scale LCM | 24.7 | 40.9 | 46.8 |
| In-domain LCM (w/o AM) + Rule | 37.7 | 52.7 | 56.4 |
| Web-scale LCM (w/o AM) + Rule | 31.6 | 47.1 | 51.8 |
| WCN + In-domain LCM (w/o AM) | 37.4 | 64.0 | 70.9 |
| WCN + Web-scale LCM (w/o AM) | 36.2 | 62.1 | 69.3 |
| WCN + In-domain LCM (w/o AM) + Rule | 41.5 | 65.5 | 71.6 |
| WCN + Web-scale LCM (w/o AM) + Rule | 39.7 | 63.2 | 69.8 |

Table 4 % occurrence of the correct word in the N -best candidate list.

| N | 1 | 5 | 10 |
|---|-------------|-------------|-------------|
| WCN | 21.9 | 41.6 | 45.6 |
| "Web-scale LCM for Sub" + "In-domain LCM (w/o AM) + Rule for Del" | 28.7 | 39.3 | 42.1 |
| WCN + "Web-scale LCM for Sub" + "In-domain LCM (w/o AM) + Rule for Del" | 32.2 | 55.5 | 61.2 |

Rule for Del". The system judges the error types depending on the user's gestures, and chooses either "Web-scale LCM" or "In-domain LCM (w/o AM) + Rule" according to the error types. **Table 4** shows the results of correcting not only substitution errors but also deletion errors. The results in Table 4 proved the effectiveness of our error type dependent error correction method: "Web-scale LCM for Sub" + "In-domain LCM (w/o AM) + Rule for Del". 28.7% of the error words were recovered by using the top 1 candidate. We also found the combined WCN and "Web-scale LCM for Sub" + "In-domain LCM (w/o AM) + Rule for Del" got the best result not only in top 1 result, but also in top 5 and top 10 result.

6.3 Analysis of LCM results

In order to analyze the effect of the recognition error contained in the context of the error region on the performance, we investigated the performance of our methods in the situation when the context of the error region does not contain any errors. **Table 5** shows the results of correcting not only substitution errors, but also deletion errors in the situation when the context of the error region does not contain errors.

Comparing Tables 4 and 5, we can conclude that the recognition errors in the context significantly make the performance worse; 3~6% of performance degradation is confirmed. Thus, we consider that it is important to propose some techniques for avoiding this influence.

Table 5 In the situation when the context of the error region does not contain errors, % occurrence of the correct word in the N -best candidate list.

| N | 1 | 5 | 10 |
|---|-------------|-------------|-------------|
| WCN | 26.0 | 47.1 | 51.8 |
| “Web-scale LCM for Sub” + “In-domain LCM (w/o AM) + Rule for Del” | 33.5 | 42.9 | 45.1 |
| WCN + “Web-scale LCM for Sub” + “In-domain LCM (w/o AM) + Rule for Del” | 37.9 | 61.0 | 66.3 |

Table 6 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 | 42 | 25 |
| Correct Type 3 errors | ≥ 116 | ≥ 86 |
| Push the “End” button | - | 1 |
| Total | ≥ 258 | ≥ 212 |

6.4 User load analysis

We compared user load using our proposed interface based on WCN + “Web-scale LCM for Sub” + “In-domain LCM (w/o AM) + Rule for Del” with the conventional interface based on WCN. Considering the small size of smartphone displays, we limited the length of the candidate list to 5.

We divided the errors into three types: Type 1 error, which can be automatically corrected by the top 1 candidate word; Type 2 error, which can be corrected by choosing from a 5-best candidate list; Type 3 error, which can be corrected through other input modalities.

In the conventional WCN based interface, after a user marks an error region, the system immediately provides the user a 5-best candidate list which includes top 1 to top 5 candidates. There are only two error types, Type 2 and Type 3. The number of operations a user needs to correct these 2 type errors are: (1) operation for correcting one Type 2 Error; (≥ 2) operations for correcting one Type 3 Error.

In our WCN + “Web-scale LCM for Sub” + “In-domain LCM (w/o AM) + Rule for Del” based interface: the 5-best candidate list includes from top 2 to top 6 candidates. The number of operations a user needs to correct these 3 type errors are: (0) operation for correcting one Type 1 Error; (1) operation for correcting one Type 2 Error; (≥ 2) operations for correcting one Type 3 Error.

Table 6 shows the analysis of user load (the number of strokes/touches) for correcting 100 error words. We used the recognition results of all the test utterances to collect the statistics, not only used the test utterances which have one error region but also used the test utterances which have more than one error region. In our method, we assume after a user marks 100 error words, he/she pushes the “End” button.

On average we saved users’ effort by 17.8% from the conventional user interface as shown in Table 6.

7. Conclusion and future work

We have proposed a simple gesture-based error correction interface based on LCM, where users mark the error word once, and then the word will be replaced by another candidate. We

used user validated information to search n -grams for long word sequences. The acoustic features of the error region are also utilized to re-rank the candidate words for the substitution case. For recovering deletion errors, it predicts a deleted word based on the phonemes and the part-of-speech tags of its surrounding words. The results proved the effectiveness of our method. The combination of WCN and LCM obtained the best results. 32.2% error words were recovered by using top 1 result. 17.8% of user effort was saved by using our interface rather the conventional user interfaces. Based on our analysis of LCM results we found that the recognition errors in the context significantly make the performance worse; 3~6% of performance degradation is confirmed. Thus, we consider that it is important to propose some techniques for avoiding this influence.

In the future work, we will examine the usefulness of our error correction method based on LCM in more general situations, when the error region contains more than one error.

References

- [1] Vertanen, K. and Kristensson, P. O.: Parakeet: a continuous speech recognition system for mobile touch-screen devices, *Proc. IUI*, pp. 237–246 (2009).
- [2] Mangu, L., Brill, E. and Stolcke, A.: Finding consensus in speech recognition: word error minimization and other applications of confusion networks, *Computer Speech & Language*, Vol. 14, No. 4, pp. 373–400 (2000).
- [3] Sturm, J. and Boves, L.: Effective error recovery strategies for multimodal form-filling applications, *Speech Communication*, Vol. 45, No. 3, pp. 289–303 (2005).
- [4] Ogata, J. and Goto, M.: Speech repair: quick error correction just by using selection operation for speech input interfaces., *Proc. INTERSPEECH*, pp. 133–136 (2005).
- [5] Nishizaki, H. and Sekiguchi, Y.: Word error correction of continuous speech recognition using WEB documents for spoken document indexing, *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead*, pp. 213–221 (2006).
- [6] Rodríguez, L., Casacuberta, F. and Vidal, E.: Computer assisted transcription of speech, *Pattern Recognition and Image Analysis*, pp. 241–248 (2007).
- [7] Liang, Y., Iwano, K. and Shinoda, K.: Simple Gesture-based Error Correction Interface for Smartphone Speech Recognition, *Proc. INTERSPEECH*, pp. 1194–1198 (2014).
- [8] Liang, Y., Iwano, K. and Shinoda, K.: An Efficient Error Correction Interface for Speech Recognition on Mobile Touchscreen Devices., *Proc. SLT* (2014).
- [9] Shinoda, K., Watanabe, Y., Iwata, K., Liang, Y., Nakagawa, R. and Furui, S.: Semi-synchronous speech and pen input for mobile user interfaces, *Speech Communication*, Vol. 53, No. 3, pp. 283–291 (2011).
- [10] Maekawa, K., Koiso, H., Furui, S. and Isahara, H.: Spontaneous speech corpus of Japanese, *Proc. LREC2000*, Vol. 2, pp. 947–952 (2000).
- [11] Dixon, P. R., Caseiro, D. A., Oonishi, T. and Furui, S.: The TITECH large vocabulary WFST speech recognition system, *Proc. ASRU*, pp. 443–448 (2007).
- [12] Kudo, T. and Kazawa, H.: Japanese Web N-gram Version 1, *Linguistic Data Consortium, Philadelphia* (2009).
- [13] Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Matsuda, H., Takaoka, K. and Asahara, M.: ChaSen morphological analyzer version 2.4.0 user’s manual. Nara Institute of Science and Technology (2007).
- [14] Stolcke, A., Zheng, J., Wang, W. and Abrash, V.: SRILM at sixteen: Update and outlook, *Proc. ASRU*, p. 5 (2011).