論文 / 著書情報
Article / Book Information

Title	Error Correction Using Long Context Match for Smartphone Speech Recognition
Authors	Yuan Liang, Koji Iwano, Koichi Shinoda
出典 / Citation	IEICE Transactions on Information and Systems, vol. E98-D, no. 11, pp. 1932-1942
発行日 / Pub. date	2015, 11
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2015 Institute of Electronics, Information and Communication Engineers.

IEICE TRANSACTIONS

on Information and Systems

VOL. E98-D NO. 11
NOVEMBER 2015

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

IEICE TRANS. INF. & SYST., VOL.E98–D, NO.11 NOVEMBER 2015

PAPER

Error Correction Using Long Context Match for Smartphone Speech Recognition

Yuan LIANG[†a)], ***Nonmember***, **Koji IWANO**[††b)], ***Member***, ***and*** **Koichi SHINODA**[†c)], ***Senior Member***

SUMMARY Most error correction interfaces for speech recognition applications on smartphones require the user to first mark an error region and choose the correct word from a candidate list. We propose a simple multimodal interface to make the process more efficient. We develop Long Context Match (LCM) to get candidates that complement the conventional word confusion network (WCN). Assuming that not only the preceding words but also the succeeding words of the error region are validated by users, we use such contexts to search higher-order *n*-grams corpora for matching word sequences. For this purpose, we also utilize the Web text data. Furthermore, we propose a combination of LCM and WCN ("LCM + WCN") to provide users with candidate lists that are more relevant than those yielded by WCN alone. We compare our interface with the WCN-based interface on the Corpus of Spontaneous Japanese (CSJ). Our proposed "LCM + WCN" method improved the 1-best accuracy by 23%, improved the Mean Reciprocal Rank (MRR) by 28%, and our interface reduced the user's load by 12%.

key words: *speech recognition, error correction, multimodal interface, word confusion network, context match*

1. Introduction

Speech input interfaces have become popular in smartphone applications [1]–[3]. In these interfaces, speech recognition errors are unavoidable due to the poor performance of automatic speech recognition (ASR) in real environments and out-of-vocabulary (OOV) words, and so on [4]. When high quality transcriptions are needed such as in email applications, users are required to verify and correct the ASR output. In the conventional speech interfaces, when a user finds some error words in the outputs, he/she corrects them one by one. First, he/she marks one error word and then either selects its corresponding correct word from a candidate list provided by the interface [5]–[7] or inputs the correct word by speech, handwriting, or virtual keyboard [8]–[11]. Taking the user's efforts and the limited user interface available in a smartphone into account, this operation should be simpler. The efficiency of a speech recognition interface depends both on its recognition accuracy and on the design of its correction interface [3]. In this paper, we focus on the latter. There are fundamentally two challenges in error correction research. The first is "how to reduce users' efforts?". The second is "how to evaluate a user interface?".

How to reduce users' efforts? One way is to design a simple user interface by using the advantage of multimodal inputs, such as speech, gestures, and handwritings [8]–[12]. Another way is to provide users a candidate list which assists users to speed up the procedure of correcting errors [2], [3], [5], [6], [13]. Word confusion network (WCN) [14] has proven to be effective to make a candidate list for an error [5]–[7]. Some researches effectively used the user validated information, which is obtained through the manual error correction procedure done by users [13], [15]–[17].

In this paper, we propose a simple multimodal error correction interface and its corresponding method to generate more accurate candidate lists. Assuming that not only the preceding contexts but also the succeeding contexts of the error word are validated by the user, we use these contexts to search higher-order *n*-grams data for the matched word sequence. We call this method Long Context Match (LCM). LCM dynamically estimates the language model (LM) score of the possible candidates from higher-order *n*-grams data, estimates the associated acoustic model (AM) score, and computes the posterior probabilities of candidates. We also utilized the Web information to complement the missed information from the in-domain data. We further propose a combination of LCM and WCN to provide users with candidate lists that are more relevant than those yielded by WCN alone.

How to evaluate a user interface? One way is using human subjective tests [3], [6]. Another way is using computer simulations. Baber et al. [18] noted that "there is a wide range of possible techniques which could be used for correcting recognition errors, and it is often difficult to compare the techniques objectively because their performance is closely related to their implementation.". In this paper, we simulate the users' load in offline experiments to avoid such dependence on implementation.

We evaluate our error correction method and interface using speech data from the Corpus of Spontaneous Japanese (CSJ) [19]. This article brings together the results that have been previously reported [20], [21] by the authors*

Manuscript received May 7, 2015.
Manuscript revised July 13, 2015.
Manuscript publicized July 31, 2015.

†The authors are with the Department of Computer Science, Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
††The author is with the Department of Information Ecology Studies, Tokyo City University, Yokohama-shi, 224–8551 Japan.
a) E-mail: yuan@ks.cs.titech.ac.jp
b) E-mail: iwano@tcu.ac.jp
c) E-mail: shinoda@cs.titech.ac.jp
DOI: 10.1587/transinf.2015EDP7179

*Paper [20] introduced the basic idea of LCM-based method and the interface design for the ideal situation, where there was only one substitution error exist in each test utterance. Paper [21] extended the LCM-based method to a situation that the contexts contained errors, but only dealt with a single substitution error and a single deletion error.

and presents new results.

2. Background

In this section, we explain the background of speech error correction interface, of candidate word lists, and of evaluation metrics.

2.1 Speech Error Correction Interface

2.1.1 Categorization

Speech error correction interfaces can be categorized from the following four different viewpoints.

1. Device: Smartphone [2], laptop, or desktop. The demand of speech interfaces for smartphones is increasing, because their keyboard interfaces are inconvenient [8]. In particular, we focus on the development of interface for smartphone applications.
2. User: Handicapped users or normal users. While voice-only error correction reported in [11] is suitable for handicapped users, we focus on the normal users in this paper.
3. Modality: Unimodal [11] or multimodal [2], [6]. We focus on multimodal. We provide users a touchscreen interface. Users can use speech input in the first step of dictation, and use a pen or fingers to correct the ASR errors. We also support keyboard input.
4. Interactivity: Without a user's input (one-step correction) or with a user's input (two-step correction) to help the system find and correct errors. We utilize the users' help to find and correct errors. This is done in two steps, first locate an error (location), and second correct an error (correction) [18]. In the first step, we ask users to locate an error word by simply marking it on the smartphone display. In the second step, we provide users a candidate list for its corresponding correct word.

2.1.2 Modality

In speech input interface, "unimodal" refers to use speech only, while "multimodal" refers to use the other modalities, including gestures and handwritings. It has been shown that multimodal error correction methods are more effective than using speech alone [12]. In the smartphone displays, gesture and handwriting inputs by means of a pen or fingers are referred to as pen inputs [8]. Most commercial applications supply a smartphone keyboard input method as the base error correction method.

Some may think using speech while correcting errors, but it does not work well. People adopt a more hyperarticulated speaking style when correcting errors. Hyperarticulation increases the mismatch between spoken correction input and the AMs of the speech recognizer trained only on normally pronounced speech. Generally, correction by repeating in the same modality frequently leads to repeated errors, and switching to a different modality may help to avoid repeated errors [1], [8], [22].

Commercial dictation systems, e.g., Dragon Dictation [23], use pen or finger selection from a list of alternatives. First users use a pen or a finger to touch an error word. Then the system presents a candidate list under the error word, from which users select a correct word by touching it. Vertanen et al. [2] proposed a multimodal error correction interface for smartphones, which displays the recognizer's best hypothesis along a single line at the top. In addition to the best hypothesis, several alternatives for each word are also displayed. Then users can correct errors by touching a correct word in the candidate list or input the correct word by using the smartphone touch keyboard.

2.1.3 Interactivity

In one-step correction, the system corrects errors with one action from users. Users repeat a phrase including the correct words, and the system automatically recognizes the phrase, identifies the errors, and corrects the errors (e.g., [11]). But since it is often unreliable as mentioned in 2.1.2, most commercial speech recognition systems employ two-steps correction.

In two-steps correction, the first step is location. Suhm et al. [8] mentioned two methods for locating recognition errors: a user-initiated method and a system-initiated method. In the user-initiated method, users look from the speech recognition results displayed on a touch screen, and users indicate the presence of an error word by touching it or saying a command such as "select [word]". In the system-initiated method, those words for which the system is less confident are highlighted [8] or appear in a shade [7]. Since confidence scores are often not reliable, the error detection may be incorrect. Suhm et al. [8] found that confidence highlighting in their dictation system did not reduce users' load to locate errors.

2.2 Candidate Word List

2.2.1 Word Confusion Network (WCN)

A word confusion network (WCN) [14] is a compact representation of multiple aligned ASR hypotheses, which is made from a word lattice or graph generated by a speech recognizer. Each competing word in an aligned segment has a posterior probability, which can be used as a confidence score of each word. WCN has been widely used to generate a candidate list for a speech recognition error [2], [3], [5], [6], [13].

2.2.2 User Validated Information

The information obtained during the past error correction process can be used for the error correction at present. This information can be called *user validated information*.

Rodriguez et al. [15], [16] 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 succeeding transcription. They proposed a natural assumption 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*. Laurent et al. [13] used this user validated prefix, a higher-order n-gram LM, and a caching LM to re-order the WCN. Wang et al. [17] proposed a multimodal error correction interface with pen gestures, where they assumed that one preceding and one succeeding words of the error region are validated by users.

However, these previous researches did not use the user validated "suffix", because they wanted to correct errors instantly after a user marks an error [13], [15], [17]. We think that, if users can wait a little bit, especially when users input longer texts, we can use not only the contexts preceding the error but also the contexts succeeding the error, and thus we can provide users more accurate candidates.

2.2.3 Outer Resources

Web text data is helpful for improving the accuracy of candidate lists. They have been widely used in augmenting the training data for adapting a LM [24]–[26], in re-scoring the first pass speech recognition results [27], in recovering OOV words [28]. Several previous researches used them to correct ASR errors. For example, Nishizaki et al. [29] used them in spoken documents indexing for correcting misrecognized proper nouns. Their system includes an automatic error correction procedure without users' interaction. It used a part of the 1-best hypothesis from the speech recognition as a query to the Google search and substituted each error word by its best alternative candidate. Nevertheless, their performance is still insufficient for our application, error correction for speech interfaces on smartphones. In this paper, we provide a feasible way of using Google's Web n-grams corpora, which is complementary to the in-domain CSJ corpus.

2.3 Evaluation Metrics

2.3.1 The Performance of Correction Methods

We use n-best accuracy and mean reciprocal rank (MRR) to evaluate the performance of error correction.

Wang et al. [17] evaluated their error correction method by calculating the accuracy of the top candidate word and the percentage the n-best list involves the correct candidate word respectively. We name the percentage of the correct words in the n-best candidate list as n-best accuracy.

MRR is a statistic for evaluating any process that produces a list of possible responses to a query, ordered by the probability of correctness. MRR is the average of the reciprocal ranks of results for a set of queries Q:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{r_i}, \tag{1}$$

where r_i is the rank of the correct word in a candidate list, and Q is the number of error words. In this paper, Q is set to the total number of substitution errors and deletion errors. MRR becomes closer to 1 as more correct answers are placed near the top of the n-best list.

2.3.2 Interface Evaluation Metrics

As we mentioned in Sect. 1, one way to evaluate a user interface is using subjective tests, and another way is using computer simulations. Ogata and Goto [6] proposed an error correction interface, in which competitive candidates are successively displayed along with the 1-best recognition hypothesis, and a user can immediately correct an error word by selecting its corresponding correct word among its candidates. They evaluated the usability by using human subjects. They measured the time required to enter some sentences including the error correction using their method. Laurent et al. [13] used word stroke ratio (WSR) and the number of actions to evaluate their error correction method. In their paper, correcting one error word counts as one stroke, and hitting a key on the keyboard counts as one action. WSR, which is a measure used in computer assisted translation of speech (CATS) [30], is the number of words to be corrected divided by the total number of words in the reference. WSR is identical to word error rate (WER) when no words are to be corrected.

In this paper, we choose to use a simulation method in order to avoid the dependence on implementation. We modify the method for counting the number of users' actions [13] and use it as a simulation method (see Sect. 5) to evaluate our interface. As the user's load, the number of strokes or touches needed to correct ASR errors is used.

3. Interface

Figure 1 shows our interface design. Its touch display consists of two regions, a text region and a button region. In the text region, the recognized text is displayed, and users

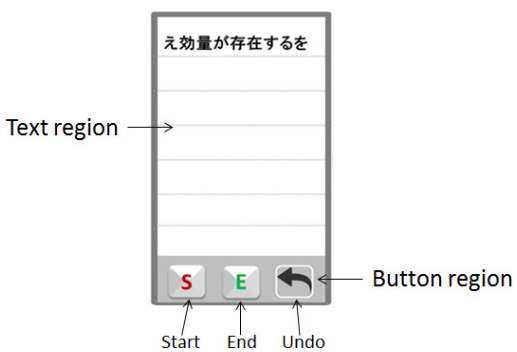


Fig. 1 Design of our interface containing two regions, a text region and a button region.

Error type	Gesture	Example
Substitution error	underline	正解が存在来る seikai ga sonzai kuru
Insertion error	strikethrough	結果をお報告する kekka o o hokoku suru
Deletion error	vertical line	友人の関係を yujin no kankei o

Fig. 2 Gestures for marking errors. An "underline" to mark a substitution error, a "strikethrough" to mark an insertion error, and a "vertical line" to mark a deletion error.

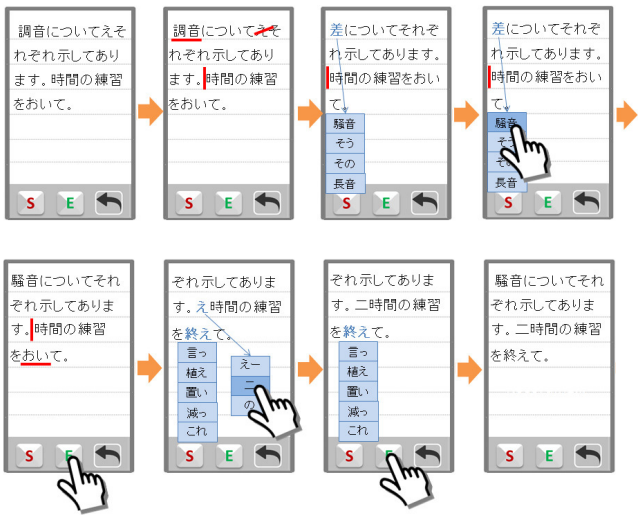


Fig. 3 Error correction procedure using our interface.

can use a pen or fingers to correct errors. The button region contains three buttons, "Start", "End", and "Undo". Since complete reversibility (Undo) is one of the key features for error handling [22], we provide users a "Undo" button. Figure 2 defines gestures for marking errors in the text region. We define *one-stroke gesture* for each of three error types: deletion, substitution, and insertion. In this section, we explain how to use our interface.

Our interface contains a recognizing function and a correcting function. Figure 3 shows an example of error correction procedures using our interface to correct errors.

Recognizing: A user taps the "Start" button and starts speaking, and taps the "End" button when he/she finishes speaking. The system recognizes the speech input and displays its 1-best hypothesis on the screen.

Correcting: Then he/she checks the text and marks errors. We suppose users mark errors from left to right. In order to use the "suffix", the system waits to correct errors until the suffix is validated by users. The system judges the suffix is validated when the user (1) marks some errors which make the number of suffix equal or larger than a predetermined number†, or (2) marks errors in the following sentence, or (3) pushes the "End" button. For a substitution

†The number is six in our paper.

Error types in an error region	Error patterns	Examples of recognized words in an error region	Number of recognized words	User rules to make gestures	System judgment	
					Error types	Number of correct words
S	S	A	1	A	S	1
D	D		0	|	D	1
I	I	A	2	~~A~~	I	0
S+	S+	A B	2	A B	SS	2
D+	D+		0	|||	DDD	3
I+	I+	A B C	3	~~A B C~~	III	0
D+S+/S+D+	D+S+	A B	2	A B	DSS	3
S+I+/I+S+	I+S+	A B C	3	A B C	ISS	2
(DI)+D+/D+(ID)+	D+S+	A	1	A	DS	2
(ID)+I+/I+(DI)+	I+S+	A B	2	A B	IS	1

Fig. 4 How our system identifies errors when multiple errors occurring successively. Error type "S" is a substitution error, "D" is a deletion error, and "I" is an insertion error. The plus sign "+" indicates there is one or more of the preceding element. For example, "(DI)+" denotes {"DI", "DIDI", "DIDIDI", ...}. "A", "B", and "C" denotes three different words. Our system identifies "DI" and "ID" as one "S". The second column shows the rules about how the system decides the error patterns.

error, the system substitutes the error word by the top-1 candidate, and shows a candidate list of the 2nd best word to the *n*th best word under the error word. For a deletion error, the system inserts the top-1 candidate and shows a candidate list of the 2nd to the *n*th best under the space where a deletion gesture is input. If the top-1 candidate is not correct, a user can select a correct word from the candidate list. If he/she cannot find a correct word in the candidate list, he/she touches the error word and inputs a correct word by handwriting [10] or keyboard [5], [10] in a pop-up window. The system judges the top-1 candidate is correct when the user marks the next error. When the user wants to finish correcting errors, he/she pushes the "End" button again or pushes the "Start" button to start inputting a new text. Considering the small size of smartphone displays, we limit the length of the candidate list to 5. The system simply deletes an insertion error.

Multiple errors often successively happen in the system output. Some interfaces ask users to mark these errors together, and they show a *n*-best phrase list. However, it is difficult to precisely estimate the correct number of words. Instead, our interface requires users to mark the same number of times as the number of the correct words (not the number of recognized words). The number of correct words equals to the sum of the number of "underlines" and the number of "vertical" lines. Our system shows a candidate list for each of substitution errors and deletion errors. While a user may use different combinations of gestures, the system decides the types of errors according to the rules. Figure 4 shows the rules about how a user makes gestures and how the system judges a user's gestures.

When a user marks one error word, both the conventional interface and our interface show a candidate list. In

the case of conventional interfaces such as Dragon Dictation [23], they keep the error word shown on the interface. For correcting the error, a user has to touch the display again to choose from the candidate list. However, our interface replaces the error word with a top-1 candidate in the first touch. If a top-1 candidate is the correct word, this one touch of a user can be saved. The 5-best candidate list of the conventional interface contains the 5-best candidates, while the candidate list of our interface contains the 2nd-6th candidates.

4. Long Context Match (LCM)

4.1 Algorithm Outline

For each error word, the system generates the candidate list for its corresponding correct word in the following steps:

Step-1 Extracts its location, error type, and user-validated prefix and suffix to make search queries.
Step-2 Uses those queries to search higher-order n-grams for matched word sequences, and from each of them extracts the word in its position as a candidate.
Step-3 If the number of candidates is one, the system directly outputs it. Otherwise, goes to Step 4.
Step-4 Calculates its LM score and AM score for each candidate.
Step-5 Calculates its posterior probability of each candidate, and orders candidates in their descending order.

We explain the details of Steps 1, 2, and 4 in the following.

4.2 Making Search Queries

Let

$$W = \underbrace{w_1, \ldots, w_{j-1}}_{W_{pre}}, w_j, \underbrace{w_{j+1}, \ldots, w_T}_{W_{suf}},$$

be one of the word sequence hypotheses corresponding to the acoustic observations X, where T is the number of words in W. Suppose one word w_j is a substitution error or a deletion error. Then, W is divided into three fragments: a prefix W_{pre}, an error word w_j, and a suffix W_{suf}. We here assume that the prefix W_{pre} is validated by users, while some words in the suffix W_{suf} may not be validated by users. Practically, since we cannot deal with a very long context, we have to limit the length (the number of words) in the context. Let p be the number of words we use in the prefix and s be the number of words in the suffix, and let W_{cp} be $w_{j-p}, \ldots, w_{j-1}$, W_{cs} be $w_{j+1}, \ldots, w_{j+s}$, and W_c be a set (W_{cp}, W_{cs}). We use a wild-card word ".*" to represent an error word. Let the maximum number of words in a query is seven. This number corresponds to the largest n in our n-grams model. For example, suppose that one ASR output contains 9 words where the 5th, 7th, and 8th words correspond to substitution errors. For the error word w_5, the longest length of queries is 7, the shortest length of queries is 2. Table 1 shows all the possible queries in this case. Any word sequence in n-grams that matches those queries is an alternative word sequence, where the word in that error word position is a candidate word for the correct word.

Table 1 All the possible search queries for error word w_5 from a word sequence $w_1, \ldots, w_9$, where w_5, w_7, w_8 are substitution errors. We set the maximum length of query to 7 and the minimum length of query is 2. ".*" is a wild card word.

	$w_1\ w_2\ w_3\ w_4 \underline{w_5}\ w_6\ \underline{w_7}\ \underline{w_8}\ w_9$
7-grams	$w_1\ w_2\ w_3\ w_4.*\ w_6\ .*$
	$w_2\ w_3\ w_4.*\ w_6\ .*\ .*$
	$w_3\ w_4.*\ w_6\ .*\ .*\ w_9$
6-grams	$w_1\ w_2\ w_3\ w_4.*\ w_6$
	$w_2\ w_3\ w_4.*\ w_6\ .*$
	$w_3\ w_4.*\ w_6\ .*\ .*$
	$w_4.*\ w_6\ .*\ .*\ w_9$
5-grams	$w_1\ w_2\ w_3\ w_4.*$
	$w_2\ w_3\ w_4.*\ w_6$
	$w_3\ w_4.*\ w_6\ .*$
	$w_4.*\ w_6\ .*\ .*$
	$.*\ w_6\ .*\ .*\ w_9$
4-grams	$w_2\ w_3\ w_4.*$
	$w_3\ w_4.*\ w_6$
	$w_4.*\ w_6\ .*$
	$.*\ w_6\ .*\ .*$
3-grams	$w_3\ w_4.*$
	$w_4.*\ w_6$
	$.*\ w_6\ .*$
2-grams	$w_4.*$
	$.*\ w_6$

4.3 Long Context Match

We use higher-order in-domain n-grams and Web n-grams. Our target is to find the word sequences which match to the search queries except the error word itself. A backoff search algorithm starts from the longest query with length "7" to length "2". Algorithm 1 shows our backoff search algorithm. We search the n-length word sequences which match to all the possible queries in n-grams. If we can not find any candidate, we search the word sequences again by using $(n-1)$-length queries in $(n-1)$-grams. We continue this process until we find at least one candidate in the current n-grams. When we get at least one candidate word from n-grams in a certain n, we stop searching. We call this process Long Context Match (LCM). If there is only one candidate in a certain n, the system directly outputs it. Otherwise, the system calculates the score of each candidate word as follows.

The conventional ASR uses the maximum a posterior (MAP) decision rule to output a word sequence $\hat{W}$, which maximizes the posterior probability $P(W|X)$ given a sequence of acoustic feature vectors X:

$$\begin{aligned} \hat{W} &= \arg\max_{W \in \Sigma} P(W|X) \\ &= \arg\max_{W \in \Sigma} \frac{P(X|W)P(W)}{P(X)}, \end{aligned} \tag{2}$$

where Σ denotes the set of all possible sentences, $P(W)$ is

Algorithm 1 Long Context Match

for (n = 7;n−−;n > 1) **do**
  for (i = n-1;i−−;$i \geq 0$) **do**
    use (w_{j-i}, …, w_{j-1}, .*, w_{j+1}, …, $w_{j+n-i-1}$) 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 searching
  end if
end for
n: "n" in n-grams
j: the error word position in a word sequence
.*: any word

the probability of W obtained by using a LM, and $P(X|W)$ is the probability of X given W obtained by using an AM. The posterior probability $P(W|X)$ is often used as a confidence measure for the decision that X is recognized as W.

Then, the word posterior probability $P(w'_j|X, W_c)$ of candidate w'_j for the jth word is calculated as follows:

$$P(w'_j|X, W_c) = \frac{P(X, W_c, w'_j)}{\sum_{w_m \in W_e} P(X, W_c, w_m)}, \tag{3}$$

where w_m denotes any possible candidate word, and W_e denotes the set of all possible candidate words for the error.

We split the sequence of acoustic feature vectors X into three fragments X_{pre}, X_e, X_{suf}, assuming their boundaries are known. Then,

$$\begin{aligned} &P(w'_j|X, W_c) \\ &= \frac{P(X_{pre}, X_e, X_{suf}|w'_j, W_c)P(w'_j|W_c)P(W_c)}{\sum_{w_m \in W_e} P(X_{pre}, X_e, X_{suf}|w_m, W_c)P(w_m|W_c)P(W_c)} \\ &= \frac{P(X_{pre}, X_e, X_{suf}|W_{cp}, w'_j, W_{cs})P(w'_j|W_c)}{\sum_{w_m \in W_e} P(X_{pre}, X_e, X_{suf}|W_{cp}, w_m, W_{cs})P(w_m|W_c)}, \end{aligned} \tag{4}$$

where X_{pre}, X_e, and X_{suf} are a sequence of the acoustic feature vectors of the preceding contexts, that of the error word, and that of the succeeding contexts, respectively. We further assume that $P(X_{pre}|W_{cp})$ does not depend on the error and the suffix, $P(X_e|w_m)$ does not depend on the prefix and the suffix, and $P(X_{suf}|W_{cs})$ does not depend on the prefix and the error. Then we can rewrite Eq. (4) as:

$$\begin{aligned} &P(w'_j|X, W_c) \\ &= \frac{P(X_{pre}|W_{cp})P(X_e|w'_j)P(X_{suf}|W_{cs})P(w'_j|W_c)}{\sum_{w_m \in W_e} P(X_{pre}|W_{cp})P(X_e|w_m)P(X_{suf}|W_{cs})P(w_m|W_c)} \\ &= \frac{P(X_e|w'_j)P(w'_j|W_c)}{\sum_{w_m \in W_e} P(X_e|w_m)P(w_m|W_c)}. \end{aligned} \tag{5}$$

Since it is difficult to precisely calculate the denominator in Eq. (5), we calculate the product $P(X_e|w_m)P(w_m|W_c)$ only for candidate words that match the search queries. Finally, we introduce a LM weight λ in order to balance between the AM score and the LM score:

$$\begin{aligned} &P(w'_j|X, W_c) \\ &= \frac{\exp(\ln P(X_e|w'_j) + \lambda \ln P(w'_j|W_c))}{\sum_{w_m \in W_e} \exp(\ln P(X_e|w_m) + \lambda \ln P(w_m|W_c))}. \end{aligned} \tag{6}$$

4.4 LM and AM Score Calculation

If we get more than one candidate words from the LCM, we calculate the LM probability of each candidate word. The probability of a candidate word w'_j is equal to the count of the word sequence including w'_j divided by the total number of counts of all the matched word sequences:

$$\begin{aligned} P(w'_j|W_c) &= \frac{C(W_c, w'_j)}{C(W_c)} \\ &= \frac{C(W_c, w'_j)}{\sum_{m=1}^{M} C(W_c, w_m)}, \end{aligned} \tag{7}$$

where $C(W_c, w'_j)$ is the number of occurrences of sequence (W_c, w'_j) in the n-grams data, $C(W_c)$ is the total number of occurrences of all the matched word sequences, and M is the number of candidate words obtained from the n-grams data.

We use the same AM as the one used in the first step speech recognition, to calculate the acoustic probability $P(X_e|w'_j)$. We utilize the grapheme-to-phoneme conversion toolkit to obtain the phoneme label sequence of each candidate word. In order to obtain speech feature vectors X_e, we need its starting time and ending time. For a substitution error, we assume that the segmentation for X is correctly done in the speech recognition process, and we use the result of segmentation. For a single deletion error, we decode the speech segment from the starting time of the previous word to the ending time of the next word for three words, and extract an AM score of the deleted word at the center. If a deletion error exists before or after another error word, its corresponding speech segment is difficult to extract. We set the AM score to zero in LCM method for such cases.

4.5 Combination of LCM and WCN

The LCM method utilizes the posterior probability obtained from the user validated contexts, while WCN method calculates the posterior probability from a word lattice or graph generated by a speech decoder. Since these two methods are complementary to each other, we combine them by linearly interpolating their posterior probabilities. For each candidate word w_r obtained from LCM and WCN, we use Eq. (8) to calculate the final posterior probability:

$$\begin{aligned} &P(w_r|X, W_c, L) \\ &= \alpha P_{\mathrm{WCN}}(w_r|L) + (1 - \alpha)P_{\mathrm{LCM}}(w_r|X, W_c), \end{aligned} \tag{8}$$

where L is a word lattice or graph for X, and α is the interpolation weight for WCN, where $0 \leq \alpha \leq 1$. Since both LCM and WCN utilize the most probable word sequence hypothesis for segmenting an utterance into words, they share the same segment for a substitution error. However, for a

deletion error, LCM may not have its aligned segments in WCN. For a substitution error or a deletion error, its candidate words may appear only in WCN or only in LCM. If a candidate word appears in LCM but does not appear in WCN, we set its posterior probability in WCN to zero. If a candidate word appears in WCN but does not appear in LCM, we set its posterior probability in LCM to zero.

5. Evaluation of the User Load

For insertion errors, in the conventional interface, two strokes are needed for correcting one insertion error. When a user locates the error, the interface will show a candidate list, and a user can choose [Delete] on the candidate list (such as [23]). In our interface, only one stroke is needed for correcting one insertion error. A user draws a "strikethrough" to delete an insertion error.

For substitution and deletion errors, we divided the errors into three types:

Type-1 error : Can be automatically corrected by the top-1 candidate word. A user does not need to do anything.

Type-2 error : Can be corrected by tapping the correct word from the candidate list.

Type-3 error : Can be corrected through the other input modalities. A user taps the error word again and edits the intended word by using the keyboard or handwriting in a pop-up window.

In the conventional interface, after a user marks an error word, the system immediately provides a 5-best candidate list. There are only two types of errors, Type-2 and Type-3 errors. The number of strokes a user needs to correct these two types of errors are: one stroke for correcting one Type-2 error, and more than one strokes for correcting one Type-3 error. In our interface based on the combination of LCM and WCN method ("LCM + WCN"), the candidate list includes from top-2 to top-6 candidates. The number of strokes required to correct these three types of errors are: zero stroke for correcting one Type-1 error, one stroke for correcting one Type-2 error, and more than one strokes for correcting one Type-3 error.

6. Experiments

We carried out two kinds of experiments. One is to confirm that our proposed "LCM + WCN" method generates more relevant candidate lists than only WCN does, and the other is to evaluate users' load of our proposed error correction interface.

6.1 Experimental Setup

We evaluated the proposed method using speech data from academic and extemporaneous lectures in the Corpus of Spontaneous Japanese (CSJ) [19]. The number of lectures is 2701, and the total length of the speech data is 530 hours. We evaluated our method by cross-validation. We randomly divided the lectures into two sets: set A contains 1350 lectures, and set B contains 1351 lectures. The triphone AM and the trigram LM were constructed. The T^3 decoder [31] was used for speech recognition. The average word recognition accuracy is 65.2%. For these two sets, there are 1,558,761 substitution errors, 593,495 deletion errors, and 248,919 insertion errors. The system identifies the types of errors based on users' gestures. We assume that this identification is always correct in this paper. While users may use many ways to mark successive multiple errors, collecting such data may take much effort. Instead, we align the recognized word sequence and the correct word sequence by using dynamic programming, and decide the error type (S, I, or D) for each error word automatically.

For in-domain LCM, we used CSJ text data to generate the word 1-grams to 7-grams and their observed frequency counts. For Web LCM, we used Google Japanese Web n-grams [32]. It consists of Japanese word 1-grams to 7-grams and their observed frequency counts generated from over 255 billion words of texts. 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 text data. Web n-grams data does not provide the pronunciation for each word. We utilized a grapheme-to-phoneme conversion tool, Chasen Morphological Analyzer [33], to convert Japanese characters to monophones. The size of Web text data, ranging from 2-grams to 7-grams, is 49.7 G. We utilized inverted index and binary search provided by Apache Solr for searching this large-scale Web text data. The minimum and maximum lengths of contexts to use LCM method is one and six, respectively.

We also evaluated the performance of the WCN-based error correction method. In order to generate the WCN-based candidate list, we employed the SRILM toolkit [34] which implements a simplified algorithm to construct WCNs [35].

6.2 Comparison of Methods

In Tables 2 and 3, we compare the n-best accuracies and MRRs of the baseline WCN-based method and the proposed LCM-based methods for Sub and Del errors. We confirmed that the ranks of correct words in the candidates lists were improved by using user validated information and higher-order n-grams text data. For example, for Sub errors, "Web LCM + WCN" improved 1-best accuracy by 11.5% and MRR by 16.7%. For Del errors, "In-domain LCM (w/o AM) + WCN" improved 1-best accuracy by 102.3% and MRR by 92.3%.

From Tables 2 and 3, we also found that the best LCM method for Sub errors was "Web LCM", and the best LCM method for Del errors was "In-domain LCM (w/o AM)". We also examined their combination "Web LCM + WCN for Sub" + "In-domain LCM (w/o AM) + WCN for Del". Table 4 shows its results for Sub and Del errors. It improved the 1-best accuracy by 23.0%, and improved MRR by 28.0%

Table 2 The n-best accuracy and MRR when correcting Sub errors. The results are obtained by using all the Sub errors from set A and set B. "w/o AM" means without using acoustic model probability. α ($0 \leqslant \alpha \leqslant 1$) is the interpolation weight for WCN method, which is optimized by using two-fold cross-validation.

Method	N-best accuracy (%)			MRR
	1-best	5-best	10-best	
WCN	22.6	39.3	44.4	0.30
In-domain LCM (w/o AM)	13.9	23.4	26.7	0.18
In-domain LCM	17.1	25.7	28.4	0.21
Web LCM (w/o AM)	12.7	22.4	26.0	0.17
Web LCM	16.1	24.9	27.8	0.20
In-domain LCM + WCN ($\alpha = 0.7$)	24.6	46.5	53.0	0.34
Web LCM + WCN ($\alpha = 0.7$)	25.2	46.1	52.3	0.35

Table 3 The n-best accuracy and MRR when correcting Del errors. The results are obtained by using all the Del errors from set A and set B. "w/o AM" means without using acoustic model probability. α ($0 \leqslant \alpha \leqslant 1$) is the interpolation weight for WCN method, which is optimized by using two-fold cross-validation.

Method	N-best accuracy (%)			MRR
	1-best	5-best	10-best	
WCN	8.6	19.4	22.7	0.13
In-domain LCM (w/o AM)	16.0	28.2	32.7	0.22
In-domain LCM	15.0	27.2	31.9	0.21
Web LCM (w/o AM)	13.1	24.9	29.9	0.19
Web LCM	12.1	23.8	28.9	0.18
In-domain LCM (w/o AM) + WCN ($\alpha = 0.2$)	17.4	34.4	41.6	0.25
Web LCM (w/o AM) + WCN ($\alpha = 0.4$)	15.1	32.0	39.2	0.23

Table 4 The n-best accuracy and MRR when correcting all the Sub and Del errors. LCM represents "Web LCM for Sub" + "In-domain LCM (w/o AM) for Del". "LCM + WCN" represents "Web LCM + WCN for Sub" + "In-domain LCM (w/o AM) + WCN for Del". The results are obtained by using all the Sub and Del errors from set A and set B.

Method	N-best accuracy (%)			MRR
	1-best	5-best	10-best	
WCN	18.7	33.8	38.4	0.25
LCM	16.1	25.8	29.2	0.21
LCM + WCN	23.0	42.9	49.4	0.32

from WCN.

We confirmed that the effectiveness of using large-scale Web text data for Sub errors. However, for Del errors, LCM using in-domain text data is better than that using Web text data. This might be due to 49% deleted words are particle words, which more often appear in the in-domain text data than in the Web text data.

We found that the acoustic probabilities are useful for correcting Sub errors. Without using acoustic probabilities, some candidate words were semantically similar but acoustically much different. For example, in an ASR output sentence "当事 者 間 で 本数 が 生じ て" (当事/t o: j i/ 者/sh a/ 間/k a N/ で/d e/ 本数/h o N s u:/ が/g a/ 生じ/sh o: j i/ て/t e/), "本数/h o N s u:/" was the error word. Before using acoustic probabilities, the top-1 candidate for corercting the error was "トラブル/t o r a b u r u/". After using acouscit probabilities, the top-1 candidate was "紛争/f u N s o:/", which is the correct word.

When correcting Del errors, we found that the acoustic probability is not helpful. The reason is that the acoustic models can not distinguish a deleted word and its surrounding words well. Most of the deleted words are short Japanese particle words, such as "を/o/", "が/g a/", "の/n o/", "と/t o/", "で/d e/". The ASR system misrecognizes a deleted word as a part of its preceding or following word. For example, in a two word phrase "情報/j o: h o:/ を/o/", "を/o/" was deleted from the recognition result. The acoustic model is not useful to recover this error, because "情報/j o: h o:/" and "情報/j o: h o:/ を/o/" are acoustically almost identical.

6.3 Comparison of Interfaces

Table 5 shows the analysis of the user's load (the number of strokes/touches) for correcting 100 error words (excluding the insertion errors) using the conventional interface [23] and proposed interface. We used all the Sub errors and Del errors to collect the statistics of the n-best accuracies by using WCN and "LCM + WCN". In our proposed method, we assume that a user pushes the "End" button after he/she marks 100 error words. When a user finishes correcting all the errors, and then he/she pushes the "End" button again. The number of strokes/touches is simulated by using the results of Table 4. From Table 5 we can see that our user interface is more effective than the conventional interface. By using "LCM + WCN"-based interface, 12% of the user's load was reduced from using WCN-based conventional interface. Furthermore, the advantage of our user interface is more obvious when correcting insertion errors.

6.4 Detailed Analysis

Here we report several diagnostic experiments and associated analyses of our LCM-based method.

6.4.1 The Effect of Multiple Errors

In order to analyze the effect of multiple errors on the performance, we investigated the performance of our LCM method for the following two groups, single error and multiple errors. We show the results in Table 6.

We noticed that in both Sub and Del error cases, the multiple errors occurring successively make the LCM performance worse, 18.1% of performance degradation for Sub case, and 46.3% of performance degradation for Del case. The ability of using LCM method to correct Del errors is rather limited in the case of successive multiple errors.

Table 5 The number of strokes which a user needs to correct 100 error words (only Sub and Del errors) in the proposed user interface and in the conventional interface. Since the conventional interface does not replace the error by using a top-1 candidate and does not have an "End" button. The symbol "-" is used for indicating zero action.

Interface	Conventional		Proposed	
Method	WCN	LCM + WCN	WCN	LCM + WCN
Mark errors	100	100	100	100
Correct Type 1 errors	-	-	0	0
Correct Type 2 errors	34	43	16	22
Correct Type 3 errors	≥ 132	≥ 114	≥ 130	≥ 110
Push the "End" button	-	-	2	2
Total	≥ 266	≥ 257	≥ 248	≥ 234

Table 6 The analysis of the effect of multiple errors. All the Sub errors and Del errors are divided into two groups, single error and multiple errors. "w/o AM" means without using acoustic model probability. Percentage written in parentheses indicates that the proportion of the single error and the multiple errors in Sub errors or Del errors.

Error groups		LCM method	1-best accuracy (%)
Sub	Single error (20%)	Web LCM	18.8
	Multiple errors (80%)	Web LCM	15.4
Del	Single error (19%)	In-domain LCM (w/o AM)	25.5
	Multiple errors (81%)	In-domain LCM (w/o AM)	13.7

Table 7 The analysis of the effect of suffix errors. All the Sub errors and Del errors are divided into two groups, correct suffix and suffix containing errors. "w/o AM" means without using acoustic model probability. Percentage written in parentheses indicates that the proportion of the correct suffix and the suffix containing errors in Sub errors or Del errors.

Error groups		LCM method	1-best accuracy (%)
Sub	Correct suffix (30%)	Web LCM	25.1
	Suffix containing errors (70%)	Web LCM	12.1
Del	Correct suffix (11%)	In-domain LCM (w/o AM)	32.2
	Suffix containing errors (89%)	In-domain LCM (w/o AM)	13.9

6.4.2 The Effect of Suffix Errors

In order to analyze the effect of suffix errors on the performance, we investigated the performance of our LCM method for the following two groups, correct suffix and suffix containing errors. We show the results in Table 7.

We noticed that in both Sub and Del error cases, the errors in the suffix make the performance worse, 51.5% of performance degradation for Sub case, and 56.8% of performance degradation for Del case. We found on average the influence of suffix errors is larger than the influence of multiple errors in case of performance degradation.

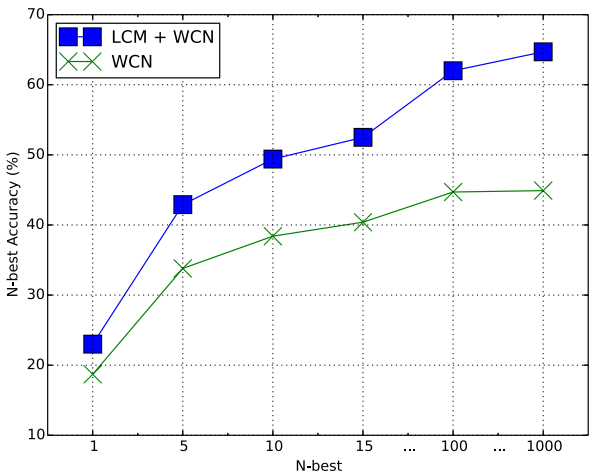


Fig. 5 *N*-best accuracy of "LCM + WCN" method and WCN method.

6.4.3 *N*-best Accuracy

We checked the *n*-best accuracy when we increased the number of candidate words into 100 and 1000 in order to know the limitation of "LCM + WCN" method and WCN method. We show the results in Fig. 5. From 100-best list to 1000-best list, the *n*-best accuracy continued to rise by using "LCM + WCN", while there was almost no change by using WCN method. We also realized that even using 1000-best list, there are still 35% of errors can not be corrected. In the future, we plan to improve the performance of the baseline speech recognizer.

6.4.4 Advantages and Disadvantages of the Proposed Method

The proposed method uses not only user validated preceding words but also user validated succeeding words, and higher-order *n*-grams text data. As a result, it can generate candidates more accurate than the WCN-based method. However, when a user marks an error, the user needs to mark the following errors to trigger the process of this error, which means the user needs to wait a little bit. In addition to this, our method is more computationally costly than the WCN-based method.

6.4.5 Usability Discussion

The system reaction time (the time required for the system to generate the feedback to the user) has a significant effect on the usability. Since the current implementation of our methods was not optimized for real use, it does not operate in real-time. Searching large-scale Web text data is the most time consuming part. As we mentioned in Sect. 6.1, we utilized inverted index and binary search provided by Apache Solr. One query search takes 0.26 sec on average when using 2 GB memory cache. The current search speed can be improved by using different ways, for example by raising Solr's cache size, by getting faster hardware, especially a faster I/O system, etc. It is not difficult to realize real-time operation.

7. Conclusions and Future Work

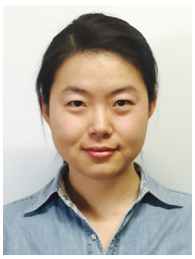
In this study, we proposed a simple multimodal error correction interface and a "LCM + WCN" method to provide users more relevant candidate lists than WCN alone. We developed a LCM method to obtain the candidates complementary to the conventional WCN results. Assuming that not only the preceding words but also the succeeding words of the error word are validated by the user, we used such a context to search the corpora of higher-order n-grams for the matching word sequences. We also utilized the Web text data, which includes additional data that does not exist in in-domain text data. By using the CSJ corpus, we confirmed the effectiveness of the proposed method and interface. Compared to the conventional WCN method, the proposed "LCM + WCN" method improved the 1-best accuracy by 23% and improved the MRR by 28%. Compared to the conventional WCN-based interface, our "LCM + WCN"-based interface successfully reduced the user's load by 12%.

Based on the analysis of LCM results, we found that the speech recognition errors in the suffix and the multiple errors significantly degrade the performance. Therefore, we consider that it is important to propose some techniques for avoiding this influence. We also found even using 1000-best candidate list, some 35% of errors still can not be corrected. Therefore, we plan to improve the performance of the baseline speech recognizer.

References

[1] K. Vertanen, "Speech and speech recognition during dictation corrections," Proc. INTERSPEECH, pp.1890–1893, 2006.
[2] K. Vertanen and P.O. Kristensson, "Parakeet: a continuous speech recognition system for mobile touch-screen devices," Proc. IUI, pp.237–246, 2009.
[3] K. Vertanen and P.O. Kristensson, "Intelligently aiding human-guided correction of speech recognition," Proc. AAAI, pp.1698–1701, 2010.
[4] P.O. Kristensson, "Five challenges for intelligent text entry methods," AI Magazine, vol.30, no.4, pp.85–94, 2009.
[5] J. Sturm and L. Boves, "Effective error recovery strategies for multimodal form-filling applications," Speech Communication, vol.45, no.3, pp.289–303, 2005.
[6] J. Ogata and M. Goto, "Speech repair: quick error correction just by using selection operation for speech input interfaces," Proc. INTERSPEECH, pp.133–136, 2005.
[7] M. Burke, B. Amento, and P. Isenhour, "Error correction of voicemail transcripts in scanmail," Proc. SIGCHI, pp.339–348, ACM, 2006.
[8] B. Suhm, B. Myers, and A. Waibel, "Multimodal error correction for speech user interfaces," ACM transactions on computer-human interaction (TOCHI), vol.8, no.1, pp.60–98, 2001.
[9] D. Huggins-Daines and A.I. Rudnicky, "Interactive asr error correction for touchscreen devices," Proc. ACL, pp.17–19, 2008.
[10] K. Shinoda, Y. Watanabe, K. Iwata, Y. Liang, R. Nakagawa, and S. Furui, "Semi-synchronous speech and pen input for mobile user interfaces," Speech Communication, vol.53, no.3, pp.283–291, 2011.
[11] J. Choi, K. Kim, S. Lee, S. Kim, D. Lee, I. Lee, and G.G. Lee, "Seamless error correction interface for voice word processor," Proc. ICASSP, pp.4973–4976, 2012.
[12] J.R. Lewis, "Effect of error correction strategy on speech dictation throughput," Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol.43, no.5, pp.457–461, 1999.
[13] A. Laurent, S. Meignier, T. Merlin, and P. Deléglise, "Computer-assisted transcription of speech based on confusion network reordering," Proc. ICASSP, pp.4884–4887, 2011.
[14] 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.
[15] L. Rodríguez, F. Casacuberta, and E. Vidal, "Computer assisted transcription of speech," Proc. IbPRIA, pp.241–248, 2007.
[16] E. Vidal, L. Rodríguez, F. Casacuberta, and I. García-Varea, "Interactive pattern recognition," Proc. MLMI, pp.60–71, 2007.
[17] L. Wang, T. Hu, P. Liu, and F.K. Soong, "Efficient handwriting correction of speech recognition errors with template constrained posterior (TCP)," Proc. INTERSPEECH, pp.2659–2662, 2008.
[18] C. Baber and K.S. Hone, "Modelling error recovery and repair in automatic speech recognition," International Journal of Man-Machine Studies, vol.39, no.3, pp.495–515, 1993.
[19] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of japanese," Proc. LREC, pp.947–952, 2000.
[20] Y. Liang, K. Iwano, and K. Shinoda, "Simple gesture-based error correction interface for smartphone speech recognition," Proc. INTERSPEECH, pp.1194–1198, 2014.
[21] Y. Liang, K. Iwano, and K. Shinoda, "An efficient error correction interface for speech recognition on mobile touchscreen devices," Proc. SLT, pp.454–459, 2014.
[22] J. Mankoff and G.D. Abowd, "Error correction techniques for handwriting, speech, and other ambiguous or error prone systems," Tech. Rep., GVU Center and College of Computing Georgia Institute of Technology, 1999.
[23] "Dragon dictation app." http://www.nuance.com/for-individuals/mobile-applications/dragon-dictation/index.htm/. Accessed May 1, 2015.
[24] G. Lecorvé, J. Dines, T. Hain, and P. Motlícek, "Supervised and unsupervised web-based language model domain adaptation," Proc. INTERSPEECH, pp.182–185, 2012.
[25] K. Yoshino, S. Mori, and T. Kawahara, "Incorporating semantic information to selection of web texts for language model of spoken dialogue system," Proc. ICASSP, pp.8252–8256, 2013.
[26] T. Schlippe, L. Gren, N.T. Vu, and T. Schultz, "Unsupervised language model adaptation for automatic speech recognition of broadcast news using web 2.0," Proc. INTERSPEECH, pp.2698–2702, 2013.
[27] X. Zhu and R. Rosenfeld, "Improving trigram language modeling with the world wide web," Proc. ICASSP, pp.533–536, 2001.

[28] C. Parada, A. Sethy, M. Dredze, and F. Jelinek, "A spoken term detection framework for recovering out-of-vocabulary words using the web," Proc. INTERSPEECH, pp.1269–1272, 2010.
[29] H. Nishizaki and Y. Sekiguchi, "Word error correction of continuous speech recognition using WEB documents for spoken document indexing," Proc. ICCPOL, pp.213–221, 2006.
[30] J. Civera, J. Vilar, E. Cubel, A. Lagarda, S. Barrachina, F. Casacuberta, E. Vidal, D. Picó, and J. González, "A syntactic pattern recognition approach to computer assisted translation," Structural, Syntactic, and Statistical Pattern Recognition, pp.207–215, 2004.
[31] P.R. Dixon, D.A. Caseiro, T. Oonishi, and S. Furui, "The titech large vocabulary wfst speech recognition system," Proc. ASRU, pp.443–448, 2007.
[32] T. Kudo and H. Kazawa, "Japanese web n-gram version 1," Linguistic Data Consortium, Philadelphia, 2009.
[33] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, and M. Asahara, "Chasen morphological analyzer version 2.4.0 user's manual," Tech. Rep., Nara Institute of Science and Technology, 2007.
[34] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "Srilm at sixteen: Update and outlook," Proc. ASRU, 2011.
[35] D. Hakkani-Tür, F. Béchet, G. Riccardi, and G. Tur, "Beyond asr 1-best: Using word confusion networks in spoken language understanding," Computer Speech and Language, vol.20, no.4, pp.495–514, 2006.



Yuan Liang received her B.E. and M.E. in computer science from Jilin Jianzhu University, China in 2005, and Changchun University of Science and Technology, China in 2008 respectively. She is currently pursuing a Ph.D. at the Tokyo Institute of Technology. She is a member of the International Speech Communication Association (ISCA).

Koji Iwano received the B.E. degree in information and communication engineering in 1995, and the M.E. and Ph.D. degrees in information engineering respectively in 1997 and 2000 from the University of Tokyo. He is currently a Professor at Tokyo City University, Faculty of Informatics. His research interests include speech recognition, speaker verification, speech synthesis, and music information processing. He is a member of the IEEE, International Speech Communication Association (ISCA), the Information Processing Society of Japan (IPSJ).

Koichi Shinoda received his B.S. in 1987 and his M.S. in 1989, both in physics, from the University of Tokyo. He received his D.Eng. in computer science from Tokyo Institute of Technology in 2001. In 1989, he joined NEC Corporation and was involved in research on automatic speech recognition. From 1997 to 1998, he was a visiting scholar with Bell Labs, Lucent Technologies. From 2001, he was an Associate Professor with the University of Tokyo. He is currently a Professor at the Tokyo Institute of Technology. His research interests include speech recognition, video information retrieval, and man-machine interaction. Dr. Shinoda received the Awaya Prize from the Acoustic Society of Japan in 1997 and the Excellent Paper Award from the Institute of Electronics, Information, and Communication Engineers (IEICE) in 1998. From 2014, he is Chief of Spoken Language Information Processing (SLP) SIG of Information Processing Society of Japan (IPSJ). He is currently an Associate Editor of Computer Speech and Language and a Subject Editor of Speech Communication. He is a member of IEEE, ACM, ASJ, IEICE, IPSJ, and JSAI.