Crossmodal Error Correction of Continuous Handwriting Recognition by Speech

Xiang Ao[■] *Xugang Wang*[■] *Feng Tian*[■] *Guozhong Dai*[▲] and *Hongan Wang*[▲]

[■]Intelligence Engineering Lab & Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, P.O.Box 8718, Beijing, China
+86-10-6254-0434
ax@iel.iscas.ac.cn
xugunw76@hotmail.com
tf@iel.iscas.ac.cn

[▲]Intelligence Engineering Lab & Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, P.O.Box 8718, Beijing, China
+86-10-6254-0434
dgz@iel.iscas.ac.cn
wha@iel.iscas.ac.cn

ABSTRACT

In recognition-based user interface, users' satisfaction is determined not only by recognition accuracy but also by effort to correct recognition errors. In this paper, we introduce a crossmodal error correction technique, which allows users to correct errors of Chinese handwriting recognition by speech. The focus of the paper is a multimodal fusion algorithm supporting the crossmodal error correction. By fusing handwriting and speech recognition, the algorithm can correct errors in both character extraction and recognition of handwriting. The experimental result indicates that the algorithm is effective and efficient. Moreover, the evaluation also shows the correction technique can help users to correct errors in handwriting recognition more efficiently than the other two error correction techniques.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. – Interaction styles; I.5.4 [Pattern Recognition]: Applications—Text processing, Signal processing;

General terms: Human Factors, Algorithms

Keywords: Error correction, multimodal fusion, handwriting recognition, speech, phoneme, weighted phoneme

1. INTRODUCTION

Pen-based user interfaces that support natural inputs by handwriting are becoming prevalent. Usually, handwriting recognition is used in these interfaces to understand user inputs. However, in these recognition-based interfaces [19], users are often frustrated by recognition errors. Study shows recognition errors significantly reduce the effectiveness of the natural input modalities, such as pen and speech [2][7]. Thus, an error correction mechanism is usually a requirement for these interfaces. Researches have shown that, in recognition-based interfaces, user satisfaction is determined not only by recognition accuracy, but also by the complexity of error correction dialogues [17] and by the efficiency of error correction [5]. Therefore, an efficient method of correcting handwriting recognition errors is important as well as necessary.

IUI'07, January 28–31, 2007, Honolulu, Hawaii, USA.

In this paper, we introduce a crossmodal error correction technique, which enables users to correct recognition errors of continuous Chinese handwriting by speech. Figure 1 gives an example. The handwriting in Fig.1 (a) is recognized as Fig.1 (b) with both character extraction errors and character recognition errors. To correct the errors, the user repeats the sentence in Fig.1 (a) by speech. By fusing the handwriting and the speech, the correct handwriting recognition result is found, as shown in Fig.1 (c).

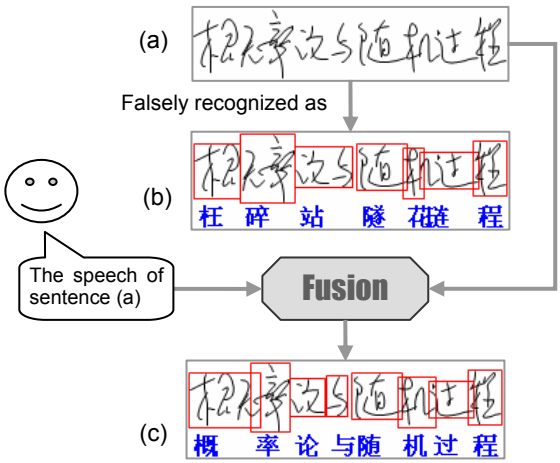


Figure 1, an example error correction of handwriting by speech. (a) the original handwriting; (b) the incorrect handwriting recognition result; (c) the correct recognition result by fusing the handwriting and speech.

The key of the error correction is a multimodal fusion algorithm, which uses speech to guide the search of the correct handwriting recognition result. We argue that the correction is to find a handwriting recognition candidate, whose pronunciation matches the user's speech best. Our analysis shows that, in order to construct such a multimodal fusion algorithm, we need to solve three problems. First, we should build a handwriting recognition candidates space to access each candidate conveniently. Second, in order to facilitate the matching, the pronunciations of the candidates and users' speech should be represented by the same format. Third, a fast search algorithm is indispensable for efficient

fusions. Based on the framework, we have constructed a crossmodal error correction algorithm on Chinese handwriting recognition by solving the three problems above. Evaluation shows the approach is effective and more efficient than the other two error correction techniques.

In the rest of the paper, we first address the motivation of the research in Section 2. Then, we review related work on error correction in recognition-based interfaces in Section 3. In Section 4, we introduce the framework of the multimodal fusion algorithm and its implementation on correct Chinese handwriting recognition errors. We show evaluation results in Section 5 and conclude the paper in Section 6.

2. MOTIVATION

We study the crossmodal error correction for three reasons. First, it is natural. Usually, people prefer to read silently when proofreading documents, digital or not. Correcting errors by speech is similar to the way (The difference is that users should read aloud, but not silently). Previous studies show that error correction strategies mimicking our daily behaviors are the most natural ways that users are pleased to accept [14]. Second, it is efficient. Efficiency is usually gained by using multiple modalities simultaneously. It needs little effort from users to speak a handwriting sentence. More important, it does not add more workload to the users' hands that are already busy. Finally, it is effective. Recent researches have shown that a system that fuses two or more information sources, which are complementary to each other, can be an effective means of reducing recognition uncertainty, thereby improving robustness [22] [23]. The audio-visual speech recognition [20] [13] is a successful application of the idea. Moreover, several researches have also showed the multimodal fusion, considering crossmodal dependencies, can improve the recognition accuracy of the unimodality [1] [4]. Therefore, we adopt a handwriting-speech fusion algorithm, by which the recognition accuracy of handwriting could be enhanced by the crossmodal influence.

3. RELATED WORK

Correcting recognition errors in recognition-based interfaces has been studied for years. Many researches have touched on -this topic, especially on correcting speech recognition errors. "Respeaking" is one of the error correction techniques, which is believed to be intuitive and natural [4]. However, the traditional "repeaking" just replaces the old wrong output with new speech recognition output, which may also be wrong. Research shows "respeaking" by simple replacement is not very useful in practice [1]. "Spelling" helps correct misrecognized words, but it is not natural and efficient in use [3] [21]. "N-best list" is another typical way to correct errors, which lets users select the correct recognition result from a list of hypotheses [18] [15], but fails when the correct result is not in the n-best list.

If multimodality is used, error correction could be improved. One way is to offer users opportunities of modality switching [22]. For example, if speech recognition fails, users can try pen input. It works because errors that a modality's recognition is prone to make may not be the minefields of others'. Another way is "multimodal correction", also called "cross-modal correction", which uses the complementarities and redundancies of different modalities. Rather than replacing old output with the new, multimodal correction fuse the inputs, or/and the recognition outputs, of different modalities to get a better recognition result [3] [16] [6] [25] [24]. The fusions often happen between modalities such as speech / pen [24], speech / mouse [16], and speech / eye-movement [25].

The proposed error correction in this paper is similar to the "respeaking" correction. However, it is crossmodal. It differs from former "respeaking" corrections adopting "replacement" strategies. Until now, few researches have been carried out on multimodal correction of handwriting recognition errors. The most closely related work is our previous research [24]. It uses handwriting recognition candidate as the language model for speech recognition to correct character recognition errors in handwriting sentences. However, it cannot correct character extraction errors, which are common in continuous handwriting recognition. The method in this paper can correct both types of errors. Thus, it is of more practical use than the method proposed in [24].

4. THE FUSION ALGORITHM

4.1 The Framework

The key of the proposed error correction method is a multimodal fusion algorithm, whose basic idea is to optimize the search of the handwriting recognition result by speech. Let X_{hw} be the handwriting, X_{sp} be the speech to correct X_{hw}'s recognition errors. Let set $\{W\}$ be all character strings and $W^* \in \{W\}$ be the fusion result. First, we have

$$W^* = \arg\max_W P(W \mid X_{hw}, X_{sp}) \quad (1)$$

By rewriting equation (1) by Bayes rules, we get

$$W^* = \arg\max_W P(W \mid X_{hw}) P(X_{sp} \mid W, X_{hw}) \quad (2)$$

As X_{hw} and X_{sp} are conditionally independent given W, equation (2) is simplified as

$$W^* = \arg\max_W P(W \mid X_{hw}) P(X_{sp} \mid W) \quad (3)$$

Let S_W be the pronunciation of W. We have $P(S_W \mid W) = 1$, because S_W is obviously determined by W. Thus, we have

$$P(X_{sp} \mid W) = P(X_{sp} \mid W, S_W) = P(X_{sp} \mid S_W) \quad (4)$$

Therefore, equation (3) can be rewritten as

$$W^* = \arg\max_W P(W \mid X_{hw}) P(X_{sp} \mid S_W) \quad (5)$$

Until now, the optimization of W^* is in the infinite set $\{W\}$. In order to make the optimization manageable, we need to find a subset of $\{W\}$ to replace $\{W\}$. Note that, if $P(W \mid X_{hw})$ is quite small, the W will very unlikely be the output of equation(5). It means that we can search W^* in a finite subset of $\{W\}$ without worrying too much about missing the optimal solution. Based on the idea, we transform equation(5) into

$$W^* = \arg\max_{W \in \{W_{hw}\}^k} P(X_{sp} \mid S_W) \quad (6)$$

, where

$$\{W_{hw}\}^k = \arg\max_W^k P(W \mid X_{hw}) \qquad (7)$$

Equation(7) says $\{W_{hw}\}^k$ is the subset of $\{W\}$ with k elements, which produce the largest k values of $P(W \mid X_{hw})$. Equation(6) says the optimization is in $\{W_{hw}\}^k$.

Equation (6) and (7) clearly show the goal of the crossmodal error correction: $\arg\max_W^k P(W \mid X_{hw})$ in equation (7) stands for the top k handwriting recognition candidates of X_{hw}, which is denoted as $\{W_{hw}\}^k$; equation (6) tells us the goal is to search the candidate from $\{W_{hw}\}^k$ whose pronunciation matches best with X_{sp}.

Equation (6) and (7) also explicitly indicate the approach of the crossmodal error correction. The approach can be divided into the solutions of three problems. Solving the three problems forms a framework of the crossmodal error correction. We show these problems as follows and answer them within the scope of correcting continuous handwriting recognition errors by speech.

1) How can we construct $\{W_{hw}\}^k$? Usually, handwriting character extraction and recognition algorithms return candidates. These candidates imply a huge set, which is actually the $\{W_{hw}\}^k$. We discuss how to construct $\{W_{hw}\}^k$ for the error correction of Chinese handwriting in Section 4.2.
2) How does the matching between S_W and X_{sp} work? Obviously, S_W and X_{sp} need the same representation format. For the error correction of Chinese handwriting, we use phonemes to represent S_W and X_{sp}. A phoneme is a symbolized representation of the pronunciation of a character, which is discussed in Section 4.3. As speech recognition may also make mistakes as handwriting recognition, we introduce "weighted phoneme" in Section 4.6 for a more accurate phonemic representation of speech.
3) How can the search of W^* be efficient? Users care about the efficiency. If the correction takes a long time, it becomes meaningless. We show that an exhaustive search, which is presented in Section 4.4, would be extremely inefficient. However, a divide-conquer strategy, which is covered in Section 4.5, would be much better.

4.2. Handwriting Recognition Errors and Candidates

Before introducing $\{W_{hw}\}^k$, we first have some discussion on handwriting recognition errors and candidates. For continuous Chinese handwriting recognition, errors are caused both by character extraction and by character recognition. A character recognition error refers to that a handwriting character is recognized as what it should not be. For example, the character  is recognized as "棍", though it should be "概". Usually, a character recognizer returns not only the best guess but also a list of candidates. It is very likely that the correct recognition result is in the candidate list.

Character extraction errors refer to a handwriting sentence is segmented into characters in a wrong way. For example, compared with the correct extraction shown in Fig.1(c), there are six characters involved extraction errors in Fig.1(b). For example,  is segmented as  , though it should be  . Usually, if character extraction is wrong, the following character recognition is meaningless. (As characters are extracted by segmenting handwriting sentences, we use "extraction" and "segmentation" interchangeably in the rest of the paper.)

We want character extraction have candidates as character recognition. An "over-segmentation" algorithm is adopted to generate all possible segmentations. Segmentation results are actually stroke blocks. Over-segmentation means every stroke block generated by the segmentation is a character or only a part of a character. These stroke blocks are called "fragments". The result of over-segmentation of the handwriting in Fig.1 (a) is shown in Fig.2.

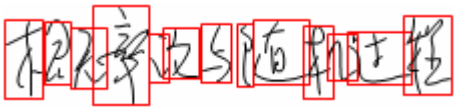
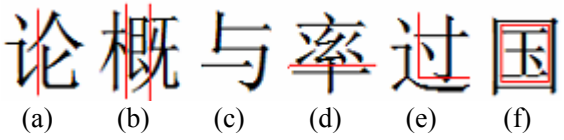


Figure 2, Over-segmentation of a sentence. The sentence in Fig.1 is segmented to 13 fragments.

Suppose sentence S is over-segmented into a fragment sequence $F = f_0 f_1 ... f_{T-1}$, where each f_i is a fragment. As any subsequence $f_j f_{j+1} ... f_k$, $0 \le i \le k < T$ of F could possibly form a character, an M-character segmentation $\mathbb{S}^M$ of S is denoted as

$$\mathbb{S}^M = (f_0 ... f_{k_1})(f_{k_1+1} ... f_{k_2}) ... (f_{k_{M-1}+1} ... f_{T-1}) \qquad (8)$$

We care how long a subsequence could be. Chinese characters have six graphemic patterns, which are shown in Fig. 3. Obviously, the over-segmentation algorithm is only sensitive to horizontal patterns. The "left-middle-right" pattern, which is most horizontally complicated, has three parts. Thus, a character could be composed of at most three fragments. We set the maximum length of a fragment subsequence as **3**.



(a) (b) (c) (d) (e) (f)

Figure 3. Six graphemic patterns of Chinese characters: (a) left-right; (b) left-middle-right; (c) single; (d) up-bottom; (e) half-surrounded; (f) fully-surrounded.

The fragments are organized as a directed graph G. The vertexes are the fragments $\{f_0, f_1, ..., f_{T-1}\}$ plus an extra node f_T. Each vertex has directed edges linked to its three succeeding, if existed, vertexes. Mathematically,

$$\begin{cases} G = \langle V, E \rangle \\ V = \{ f_i \mid i \in [0, T-1] \} \cup \{ f_T \} \\ E = \left\{ \langle f_i, f_j \rangle \mid (j \le T) \wedge (0 \le i < j \le i+3) \right\} \end{cases} \quad (9)$$

Figure 4 show a graph of seven fragments plus an extra vertex f_7 as the ending. With the graph, we can get all possible segmentations by enumerate paths starting at f_0 and ending at f_T. For example, path $f_0 \to f_1 \to f_4 \to f_5 \to f_7$ denotes the segmentation $(f_0)(f_1 f_2 f_3)(f_4)(f_5 f_6)$

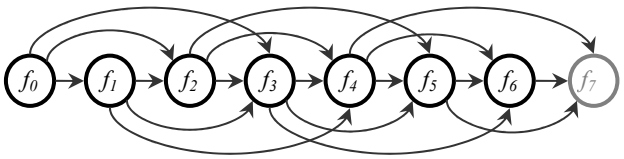


Figure 4. the graph of seven fragments. Note f_7 is the extra node, which is not a fragment.

We now show how to build the $\{W_{hw}\}^k$ in Equation(7)and(8). Suppose the character recognizer returns t candidates for a handwriting character. As segmentation $\mathbb{S}^M$ of S has M characters, it has t^M recognition candidate sequences, each of which is a W_{hw} in $\{W_{hw}\}^k$.

We only considered a single segmentation above. However, as Fig.4 demonstrates, there are many segmenting ways for a sentence. For a sentence with T fragments, $C(T)$ denotes the number of its different segmentations. We have

$$\begin{cases} C(T) = \sum_{i=1}^{3} C(T-i) \\ C(1) = 1;\ C(2) = 2;\ C(3) = 4 \end{cases} \quad (10)$$

,where $C(T)$ is actually a Tribonacci Number [11] , which is equal to $[\![\alpha \times \beta^{T+1}]\!]$, where $[\![x]\!]$ is the nearest integer of x, $\alpha \approx 0.336$ and $\beta \approx 1.839$. Combing the character recognition and segmentation, we eventually know the size of $\{W_{hw}\}^k$ as

$$\left| \{W_{hw}\}^k \right| = \sum_{i=0}^{[\![\alpha \times \beta^{T+1}]\!]} t^{|\mathbb{S}_i|} \quad (11)$$

where $|\mathbb{S}_i|$ stands for the number of characters in $\mathbb{S}_i$. According to the definition of $\{W_{hw}\}^k$, we have $k = \left| \{W_{hw}\}^k \right|$. Obviously, k is tremendous. This is why we say in subsection 4.1 that $\{W_{hw}\}^k$ would be "a huge set" and why we need a fast search algorithm for W^*.

4.3 Phonemes

To unify the representation format of S_W and X_{sp}, we use phonemes. A phoneme is the symbolized representation of a character's pronunciation. For Chinese, We use Hanyu Pinyin to denote phonemes. Hanyu Pinyin (pinyin) is the phonemic notation system for the pronunciation of Chinese (Standard Mandarin). Every Chinese character has its pinyin, which is a compound of an initial (*in*), a final (*fn*) and a tone. For example, the pinyin for character "逃" (escape) is "táo", in which the "t" is the initial as [t] in English, the "ao" is the final as [au] in English, and the " ´ " is the tone mark. An introduction of pinyin can be found in [9]. A phoneme *ph* is a pair of an initial and a final denoted as

$$ph = [in, fn] \quad (12)$$

There are 23 initials and 38 finals in Chinese (But not all combinations of them are permitted.). To avoid confusion, we use in^i to denote the ith initial in the initial table and fn^j to denote the jth final in the final table. We use in_k and fn_k to denote the initial and final in phoneme ph_k respectively.

To compare phonemes, we define phonemic similarity. The similarity, $S(ph_1, ph_2)$ of phonemes $ph_1 = [in_1, fn_1]$ and $ph_2 = [in_2, fn_2]$ is defined as

$$S(ph_1, ph_2) = sIn(in_1, in_2) + sFn(fn_1, fn_2) \quad (13)$$

, where $sIn(in_1, in_2)$ and $sFn(fn_2, fn_2)$ are the similarity of two initials and finals respectively. The $sIn(in_j, in_k)$ and $sFn(fn_j, fn_k)$ come from observations. Intuitively, the two similarity metrics equals 0 when two compared items are same, and equals 1 when they sound completely different.

The similarity of phoneme sequences is also necessary. As phoneme *ph* is an initial-final pair $[in, fn]$, a phoneme sequence $PH = ph_1 ph_2 \ldots ph_{N-1}$ can be written as a token sequence $PH = in_0 fn_0 in_1 fn_1 \ldots in_{N-1} fn_{n-1}$, in which each initial or final is a token. To compare two token sequences, Levenshtein distance (LD, also called "edit distance") [8] is an appropriate measure, which calculates the minimum number of operations needed to transform one token sequence into the other, where an operation is an insertion, deletion, or substitution of a token. Thus, given phoneme sequence PH_1 and PH_2, the similarity between them is defined as their Levenshtein distance $LD(PH_1, PH_2)$. In our implementation, the cost of substitution *cost_sub* is adjusted as

$$cost\_sub(a,b) = \begin{cases} sIn(a,b), & \text{if both a and b are intials} \\ sFn(a,b), & \text{if both a and b are finals} \\ \infty, & \text{otherwise} \end{cases} \quad (14)$$

Note $cost\_sub(a,b) = \infty$ when one parameter is initial and the other is not. It means that we forbid interchanges between initials and finals.

The phonemes for users' speech are got by transforming the speech recognition result into a phoneme sequence. If the speech recognizer supports intermediate phonemic output, we just need to translate the output to our phoneme format. However, if the speech recognizer can only return text, we transform the text to the phoneme sequence by looking up the pronunciations of the text from a dictionary. The phonemes for the pronunciation of the handwriting recognition candidate $W \in \{W_{hw}\}^k$ are got by using a dictionary similarly as handling the speech.

4.4. Fusion by an Exhaustive Search

In this subsection, we introduce a straightforward but slow search strategy to find W^* from $\{W_{hw}\}^k$. It just exhausts all $\{W_{hw}\}^k$ by comparing the pronunciation S_W of W with the speech, and choose the most matched on as W^*. The matching is done by using the function $LD(PH_W, PH_{SP})$ defined in subsection 4.3. In the $LD(PH_W, PH_{SP})$ here, PH_W denotes the pronunciation of W, and PH_{SP} de-

notes the speech, both of which are phoneme sequences. We call the search $ExaustiveFusion(F, PH_{SP})$. The first parameter F is the fragment sequence, which actually denotes the handwriting sentence. The second PH_{SP} has introduced above. $ExaustiveFusion(F, PH_{SP})$ returns the minimum matching cost, which implicitly indicates W^*.

Let us estimate the complexity of $ExaustiveFusion(F, PH_{SP})$. As the size of $\{W_{hw}\}^k$ has been shown in Equation(11) and the complexity of $LD(PH_W, PH_{SP})$ is $O(|PH_1|, |PH_2|)$ [8], the complexity of $ExaustiveFusion(F, PH_{SP})$ is

$$O\left(\sum_{i=0}^{\lfloor \alpha \times \beta^{T+1} \rfloor} t^{|\mathbb{S}_i|}\left(|\mathbb{S}_i| \times |PH_{SP}|\right)\right),$$

It is obvious that $ExaustiveFusion(F, PH_{SP})$ is extremely inefficient.

4.5. Fusion by a Divide-Conquer Search

In this subsection, a fast search algorithm, which adopts a divide-conquer strategy, is introduced to find W^* from $\{W_{hw}\}^k$. First, note that for the fragment sequence

$$F = f_0 \cdots f_{i-1} f_i f_{i+1} f_{i+2} \cdots f_{T-1},$$

f_{i-1}, f_i, f_{i+1} and f_{i+2} cannot all be in the same character, because a character has at most three fragments. Two adjacent fragments are called "separate fragments" if they are not in the same character. In F, either f_{i-1} and f_i, or f_i and f_{i+1}, or f_{i+1} and f_{i+2} must be separate. It is called a "separation" that F is divided into two subsequences by a separate fragment pair. For example, if f_i and f_{i+1} are separate, F is divided into two subsequences

$$\begin{cases} F_{0,i} = f_0 \cdots f_{i-1} f_i \\ F_{i+1,T-1} = f_{i+1} f_{i+2} \cdots f_{T-1} \end{cases}.$$

Based on the idea, we define function $DCFusion(F_{i,j}, PH_{k,l})$ as the minimum matching cost of fragment sequence $F_{i,j}$ and phoneme sequence $PH_{k,l}$, where $PH_{k,l}$ refers to a subsequence of PH_{SP}, which starts from the kth phoneme and ends at the lth phoneme in PH_{SP}. $DCFusion(F_{i,j}, PH_{k,l})$, which adopts a divide-conquer strategy, is calculated as follows.

$$DCFusion(F_{i,j}, PH_{k,l}) = \begin{cases} ExaustiveFusion(F_{i,j}, PH_{k,l}), if\ j - i < threshold \\ \min_{t-1 \le p \le t+1}\left(DCCost(p, F_{i,j}, PH_{k,l})\right), else \end{cases} \quad (15)$$

where

$$t = (i+j)/2 \quad (16)$$

and

$$DCCost(p, F_{i,j}, PH_{k,l}) = \min_{k \le q \le l}\begin{pmatrix} DCFusion(F_{i,p}, PH_{k,q}) \\ + DCFusion(F_{p+1,j}, PH_{q+1,l}) \end{pmatrix} \quad (17)$$

Equation (15) and (16) say that if $F_{i,j}$ is shorter than a $threshold$ it is calculated directly by the exhaustive fusion introduced in subsection 4.4, else three separations of it are tried. In our implementation, $threshold = 5$. Equation (17) says, for a separation, the two fragment subsequences try to match all possible partitions of phoneme sequences.

Equation (17) could be improved by narrowing the range of parameter q without losing much accuracy. A fragment subsequence $F_{i,p}$ only needs to match those phoneme subsequences, which indicate similar character range as indicated by $F_{i,p}$. We define $Len(i, p)$ as the geometric width from f_i to f_p, and ratio

$$Pos(p) = Len(i, p) / Len(i, j)$$

as the normalized position of f_p in $F_{i,j}$. In fact, $Pos(p)$ also indicates the approximate normalized position of the character, to which f_i belongs, in the potential character sequence. As each character corresponds to a phoneme in $PH_{k,l}$, thus

$$PH_{k,PhIdx(p)} = ph_k ph_{k+1} \cdots ph_{PhIdx(p)}$$

where $PhIdx(p) = k + \left\lfloor |PH_{k,l}| \times Pos(p) \right\rfloor$ is the most appropriate phoneme subsequence that could be matched by $F_{i,p}$. Figure 5 illustrates calculating $PhIdx(p)$. Now we can narrow the range of parameter q in Equation(17) as $PhIdx(p) - \lambda \le q \le PhIdx(p) + \lambda$. In our implementation, $\lambda = 2$.

Let us estimate the efficiency of $DCFusion$. Applying $DCFusion$ to the handwriting in Fig.1, we eventually decompose the whole problem to 775 $ExaustiveFusion$ problems, whose T, M and N are only about a quarter of the originals respectively. Approximately, the time complexity of $DCFusion$ is

$$O\left(E(t) \times 30^{\log_2(|F|/t)}\right),$$

where t is the $threshold$ in Equation (15) and $E(t)$ is the expectation of execution time of $ExaustiveFusion$ on t fragments. Obviously, $DCFusion$ is much more efficient than $ExaustiveFusion$.

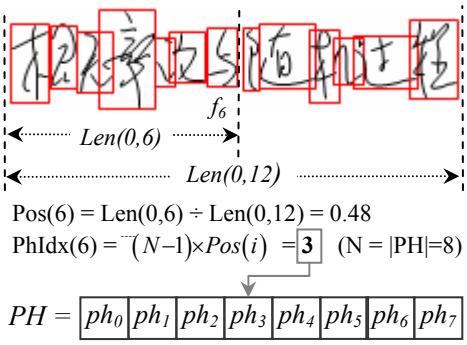


Figure 5 computation of which sub-sequence of speech phoneme sequence match the fragment sequence. In this example, for $f_0 f_1 \cdots f_6$, the speech sub-sequence is $ph_0 ph_1 ph_2 ph_3$

4.6. Weighted Phoneme

Speech recognition has errors, which make the phonemes produced by it inaccurate. However, speech recognition also returns recognition candidates as handwriting recognition does. These candidates provide more information of users' utterances. We use "weighted phoneme" to represent them.

A weighted phoneme wph, denoted by $[win, wfn]$, is the mixture of several phonemes, in which win is a weighted initial and wfn is a weighted final. We have

$$\begin{cases} win = [w_0, ..., w_{22}],\ \forall w_{j\in[0,22]}, w_j \geq 0, \sum_{j=0}^{22} w_j \leq 1 \\ wfn = [v_0, ..., v_{37}],\quad \forall v_{k\in[0,37]}, v_k \geq 0,\ \sum_{k=1}^{37} v_k \leq 1 \end{cases} \tag{18}$$

,where w_j and v_j are weights. Thus, a win is a linear combination of all 23 initials and a wfn is a linear combination of all 38 finals. Without losing meaning, we simplify the notations of win and wfn as

$$\begin{cases} win = [w_0 \cdot in^0, w_1 \cdot in^1, ..., w_{M-1} \cdot in^{22}] \\ wfn = [v_0 \cdot fn^0, v_1 \cdot fn^1, ..., v_{N-1} \cdot fn^{37}] \end{cases}$$

We can omit items whose weights are zero. Trivially, a phoneme is also a weighted phoneme, whose win and wfn have only one item respectively with weight being one.

Suppose the speech recognition result for character C has candidate $ph_0, ph_1, ..., ph_{k-1}$. Each $ph_i = [in_i, fn_i]$ is a phoneme with confidence t_i. The weighted phoneme for C is denote as wph. The weights in wph are calculated as

$$\begin{cases} w_i = \sum_{j=0}^{k} [t_j \times equal(in_j, in^i)] \\ v_i = \sum_{j=0}^{k} [t_j \times equal(fn_j, fn^i)] \end{cases} \tag{19}$$

in which

$$equal(a,b) = \begin{cases} 1, & \text{if } a \text{ and } b \text{ are same} \\ 0, & \text{else} \end{cases} \tag{20}$$

For example, if speech recognition candidates for character "逃" is $[t, ao], [t, iao], [d, ao], [t, ou]$,with confidences t_0, t_1, t_2 and t_3 respectively, the weighted phoneme is

$$wph_{逃}: \begin{cases} win_{逃} = [t_2 \cdot d, (t_0 + t_1 + t_3) \cdot t] \\ wfn_{逃} = [(t_0 + t_2) \cdot ao, t_1 \cdot iao, t_3 \cdot ou] \end{cases}$$

Weighted phonemes can also represent different segmentations in speech recognition. For example, suppose the speech recognition result is denoted by candidates $(ph_0), (ph_1), (ph_{2,0}, ph_{2,1})$, with confidences t_0, t_1 and t_2 respectively. The first and second candidates have one phoneme, but the third has two. It means the recognizer is not sure whether the utterance corresponds to a character or two. We represent the utterance by two weighted phonemes wph_1 and wph_2. First, we add null phonemes $ph_{0,1}$ and $ph_{1,1}$ to candidates 1 and 2 respectively to let the candidates become $(ph_{0,0}, ph_{0,1}), (ph_{1,0}, ph_{1,1}), (ph_{2,0}, ph_{2,1})$. Now we calculate wph_1 by $ph_{0,0}, ph_{1,0}$ and $ph_{2,0}$, and wph_2 by $ph_{0,1}$, $ph_{1,1}$ and $ph_{2,1}$, by using Equation(19). As $ph_{0,1}$ and $ph_{1,1}$ do not exist in fact, the sum of the weights of the wph_2's weighted initial is less than one. (In fact, it equals to t_2). It is same for wph_2's weighted final. The results reflect that wph_2 may indicate a segmentation error of speech recognition.

Usually, the speech recognition result of a sentence is denoted by a sequence of phrases. (The phrase segmentation is considered correct.) Each phrase corresponds to at least one character. A phrase can be denoted by weighted phonemes by the way introduced above. By combining the weighted phonemes of these phrases, we get a weighted phonemes sequence WPH for the speech recognition result for a sentence.

We can substitute phonemes with weighted phonemes for computation. To do this, we need to define the similarity between two weighted phonemes.

The similarity between weighted phonemes is calculated as follows. Given weighted phonemes $wph_1 = [win_1, wfn_1]$ and $wph_2 = [win_2, wfn_2]$. The Similarity $S(wph_1, wph_2)$ of is calculated as

$$S(wph_1, wph_2) = swIn(win_1, win_2) + swFn(wfn_1, wfn_2) \tag{21}$$

in which $swIn(win_1, win_2)$ is the similarity between two weighted initials and $swFn(wfn_1, wfn_2)$ is the similarity between two weighted finals:

$$\begin{cases} swIn(win_1, win_2) = \sum_{j=0}^{M-1} \sum_{k=0}^{M-1} w_{1j} w_{2k} sIn(in^j, in^k) \\ swFn(wfn_1, wfn_2) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} v_{1j} v_{2k} sFn(fn^j, fn^k) \end{cases} \tag{22}$$

5. EVALUATION

We have conducted a preliminary evaluation on the proposed crossmodal error correction technique. Three aspects are concerned: a) Is it effective in correcting errors? b) Is it efficient in computation? c) Compared with other correction techniques, is it more efficient?

In our implementation, the recognizers of speech and handwriting were the Microsoft SAPI 5.1 [12] and the HANGWANG Chinese handwriting character recognizer [10] respectively. The computer used is a Tablet PC, which is equipped with a 1.6GH CPU and 512MB memory. The pen-input supported LCD screen of Tablet PC is used for handwriting, and the built-in microphone is used for speech input. Twelve subjects participated in the experiment. They are graduate students from our lab. They are divided into three groups G1, G2 and G3, each of which has four people.

Sixty handwriting sentences with both segmentation and recognition errors were used as experimental data. These sentences were divided into three data sets D1, D2 and D3, each of which has twenty sentences. We define "error rate" of a sentence as

$$\text{error rate} = \left(1 - \frac{|\text{correctly recognized characters}|}{|\text{characters}|}\right) \times 100\%$$

Error rate of sentences in D1 is around 15%. Error rate of sentences in D2 is around 30%. Error rate of sentences in D3 is around 50%. Three correction techniques T1, T2 and T3 are used for comparison. T1 is the error correction technique that uses pen gestures and character recognition candidate list (See [24] for its details). T2 is our former speech correction technique introduced in [24], which can only correct character recognition errors but not segmentation errors. T3 refers to the correction technique proposed in the paper.

We let subjects in G1 use technique T1 to correct the errors in all data sets. We let G2 use T2 and G3 use T3 to perform the same task as G1. For each sentence, G2 can only use T2 once and G3 can only use T3 once. G2 and G3 could use T1 for help, because both of the two speech correction techniques cannot guarantee 100 percent of error corrections. However, when using T3, subjects were asked to use the speech correction before using T1.

Figure 6 shows the experiment result on the comparisons of the efficiency of the three techniques T1, T2 and T3. It shows that, in D1, three techniques took nearly the same time. However, in D2 and D3, T3 considerably outperformed T1 and T2. The result is easy to understand. T1 is a pen-only correction. The time consumed by it will increase much with the increase of the error number. T2 cannot correct segmentation errors. When errors number increases, the segmentation errors, which require manual corrections by users using T2, are also becoming more and more. T3 can correct both segmentation and recognition errors. Thus, it is more efficient than the other two techniques when error rate is large. The result also suggests our technique T3 is more suitable for error-intensive correction.

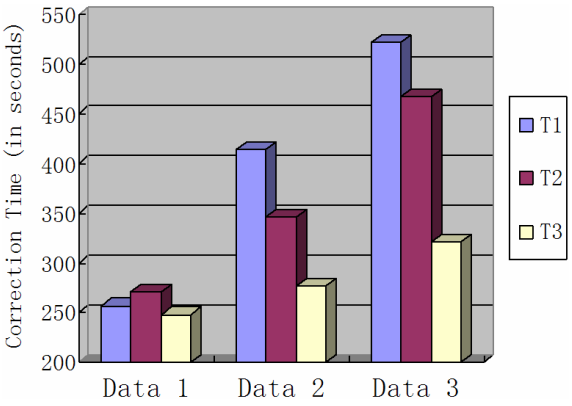


Figure 6. Comparisons of the performance on efficiency of three correction techniques on three data sets. The Y-axis stands for the time that subjects took for corrections. The three data sets are in the X-axis.

Figure 7 demonstrates effectiveness of the proposed correction technique, by comparing error numbers before and after correction by speech. The result in Fig.7 shows our speech correction techniques can recover most of errors. (The rest of errors can be corrected by pen gestures.) The result also justifies the adoption of weighted phoneme. The fusions using weighted phonemes for speech representation remarkably outperform those using phonemes.

The correction might fail in the following situations: a) The candidates of character recognition of handwriting does not contain the correct output. b) The candidates are similar both in appearance and in pronunciation. c) There are failures in "over-segmentation" introduced in Section 4.1. d) Speech recognition severely fails, which means the recognized pronunciations are far from what should be.

Figure 8 shows execution times of our fusion algorithm. The fusion takes about 0.25 seconds for 5-character sentence, which is nearly unnoticeable, and 3.3 seconds for 18-character sentence, which was considered acceptable by the subjects.

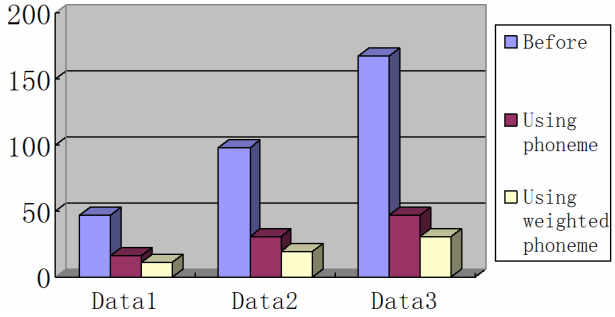


Figure 7. Comparisons of error counts before correction, after a correction by using phoneme and after a error correction by using weighted pho-

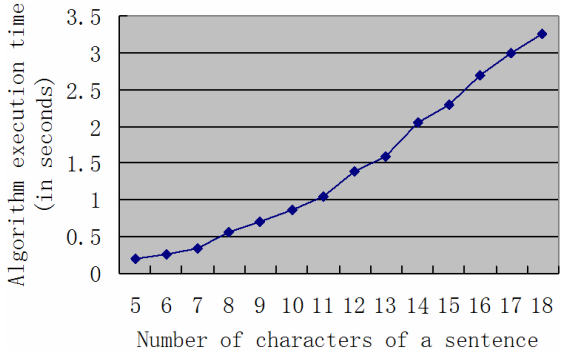


Figure 8. The time to take to for the execution of the fusion algorithm proposed in this paper on sentences of different numbers of words.

6. CONCLUSION

To naturally and efficiently correct handwriting recognition errors, we propose a crossmodal error correction technique, which allows a user to correct both character extraction and recognition errors of a Chinese handwriting. For the correction, a multimodal fusion algorithm is introduced with its framework and implementation details, whose main idea is to find the handwriting recognition result whose pronunciation matches the speech best. Evaluation shows the algorithm works effectively and efficiently. Compared to the

other two correction techniques, the proposed technique is more efficient.

7. ACKNOWLEDGMENTS

This research is supported by the National Fundamental Research Project of China (973 Project) (2002CB312103), the National Natural Science Foundation of China under Grant No. 60603073, 60503054 and 60605018.

REFERENCES

1. Ainsworth, W. A. And Pratt, S. R. 1992. Feedback strategies for error correction in speech recognition systems. Int. J. Man-Mach. Stud. 36, 6 (June), pp. 833–842.
2. B. Suhm, B. Myers and A. Waibel, Model-based and empirical evaluation of multimodal interactive error correction, Proc. ACM CHI'99, pp. 584-591, 1999,
3. B. Suhm, B. Myers and A. Waibel, Multimodal error correction for speech user interfaces, ACM Transactions on Computer-Human Interaction pp.60-98, 2001
4. BABER, C. AND HONE, K. S. 1993. Modeling error recovery and repair in automatic speech recognition. Int. J. Man-Mach. Stud. 39, 3 (Sept.), pp. 495–515
5. C. Frankish, D. Jones and K. Hapeshi. Decline in accuracy of automatic speech recognition as function of time on task: fatigue or voice drift. International Journal of Man-Machine Studies, 36(6): pp.797–816, 1992.
6. C. Halverson, D.B. Horn, C. Karat and J. Karat. The beauty of errors: patterns of error correction in desktop speech systems. In Proceedings of INTERACT '99, pp.133–140, IOS Press. 1999
7. C-M. Karat, C. Halverson, D.Horn and J. Karat, Patterns of entry and correction in large vocabulary contentious speech recognition systems, Proc. ACM CHI'99, 1999, pp. 568-575.
8. http://en.wikipedia.org/wiki/Levenshtein_distance
9. http://en.wikipedia.org/wiki/Pinyin
10. http://www.hwpen.net/
11. http://mathworld.wolfram.com/TribonacciNumber.html
12. http://www.microsoft.com/speech/download/sdk51/
13. J. Luettin, "Visual Speech and Speaker Recognition," Ph.D. dissertation, Univ. Sheffield, U.K., 1997.
14. J. Mankoff and G. Abowd. Error correction techniques for handwriting, speech, and other ambiguous or error prone systems. GVU Technical Report Number: GIT-GVU-99-18,1999.
15. J. Mankoff, S. Hudson and G.D. Abowd. Providing Integrated toolkit-level support for ambiguity in recognition-based interfaces, Proc. ACM CHI'00, pp. 368-375, 2000.
16. J. Sturm and L. Boves, Effective error recovery strategies for multimodal form-filling applications, Speech Communication 45, pp.289 – 303, 2005
17. M. Zajicek and J. Hewitt. An investigation into the use of error recovery dialogues in a user interface management system for speech recognition. In Proceedings of 3rd IFIP International Conference on Human-Computer Interaction, IFIP INTERACT'90, pp. 755–760.
18. Murray, A. C., Frankish, C. R., And Jones, D. M. 1993. Data-entry by voice: Facilitating correction of misrecognitions. In Interactive Speech Technology: Human Factors Issues in the Application of Speech Input/Output to Computers, C. Baber and J. M. Noyes, Eds. Taylor and Francis, Inc., Bristol, PA, pp.137–144
19. Rhyne,J.R. and Wolf, C.G. 1993. Recognition-based user interfaces. In Advances in Human-Computer Interaction, H. R. Hartson and D. Hix, Eds. Ablex Publishing Corp., Norwood, NJ, pp. 191–212
20. S. Dupont and J. Luettin Audio-Visual Speech Modeling for Continuous Speech Recognition, IEEE Transactions On Multimedia, Vol. 2, No. 3, September 2000, pp. 141-151
21. S. Oviatt and R. van Gent, Error Resolution During Multimodal Human–Computer Interaction, Proc of the Fourth International Conference on Spoken Language Processing, October 1996, pp. 204-207
22. S. Oviatt. Taming recognition errors with a multimodal interface. Communication of the ACM, 43(9): pp. 45–51, 2000
23. S.Oviatt, Ten Myths of Multimodal Interaction, Communications of the ACM, November 1999/Vol. 42, No. 11 pp. 74 – 81
24. Xugang Wang, Junfeng Li, Xiang Ao, Gang Wang and Guozhong Dai, Multimodal Error Correction for Continuous Handwriting Recognition in Pen-based user Interfaces, Proceedings of IUI 2006, pp.324-326
25. Yeow Kee Tan, Nasser Sherkat and Tony Allen, Error Recovery in a Blended Style Eye Gaze and Speech Interface, Prceedings of ICMI 2003, pp. 196 – 202