Error Correction for Recognition of Handwritten Kanji Names Using a Name Knowledge Base

Katsumi MARUKAWA*, Masashi KOGA*, Yoshihiro SHIMA*, and Hiromichi FUJISAWA**

* : Central Research Laboratory, Hitachi, Ltd.
1-280 Higashikoigakubo Kokubunji-shi Tokyo, 185 JAPAN
** : Software Development Center, Hitachi, Ltd.
549-6 Shinano-cho Totsuka-ku Yokohama-shi Kanagawa, 244 JAPAN

Abstract

The error correction algorithm for handwritten Kanji name recognition is presented, here automatically corrects errors and rejects even when users write Kanji names without being conscious of formal notation rules. The algorithm consists of a word matching function and an evaluation function. For word matching, a high-speed automaton-type method extracts, in real time, 1st and 2nd name from any position in one continuous name. The evaluation function is carried out only when 1st and 2nd name are written as one continuous string. The evaluation function determines the right name by using a positional condition between 1st and 2nd name. When this algorithm was run on 13,926 handwritten Japanese Kanji names, it corrected 63.3 percent of the recognition errors and rejects in real time.

1. Introduction

Efficient automation of offices will require information entry systems [1]-[3] to use an Optical Character Reader (OCR) that can recognize the information written on papers. Information entry systems must recognize large amounts of information, and a name is a useful reference for arranging of information. The system should perform the following three tasks: it should correct recognition errors and rejects with a high precision, it should recognize names in real time, and it should handle names written in several notations.

Several Kanji name entry methods have already been developed, and one uses only the results of an OCR. Handwritten Kanji recognition has problems. For example, Kanji has about 7,000 categories, many similar characters, and much variability between writers. This method is therefore not suitable for information entry systems. Another method [4]-[5] uses a knowledge base. In recognizing characters on a voucher, recognition items name, address and, so on are determined in advance, so that this method can recognize the items by using appropriate knowledge bases. The procedure matches results of recognition with words included in a knowledge base. That is, the procedure is word matching. Word matching algorithms for Kanji recognition are classified into two categories. One category (a) is a compound word matching [6].

The second category (b) of algorithm we propose is an automaton-type word matching [7]. In category (a), a string is compounded of candidate characters. If the string exists within a knowledge base, it is regarded as a candidate word. When a continuous string consists of multiple words, this algorithm creates huge strings. And when the right character is not among the candidate characters, the right word can not be extracted. The length of a word is L and the number of compounded strings is $\Sigma_{i=1,S}\Sigma_{j=1,i}K^{L}$. When K is about 3 [6], this algorithm has no problem. But when K is about 20, it is not suitable as the word matching algorithm. Algorithm (b), the other hand, uses a Finite State Automaton (FSA) [8]-[9] made from candidate characters. Words to input to the FSA are searched from within the knowledge base according to candidate characters. And this algorithm has a faculty for extracting words even when the right character is not among the candidate characters. As a result algorithm (b) is faster than algorithm (a) and can be more precise.

This paper proposes an error correction algorithm for handwritten Kanji name recognition. Errors and rejects of a character recognition are automatically corrected with the help of a name knowledge base. Furthermore, when users write Kanji name on a voucher, they usually must obey certain rules. Our proposed algorithm, however, allows users to write names without being conscious of the rules. It consists of a word matching function and an evaluation function. For word matching, a high-speed automaton-type method [10]-[11] extracts the 1st and 2nd names from any position in one continuous name in real time. The evaluation function is carried out only when 1st and 2nd name are written as one continuous string. The evaluation function evaluates the name by using the positional condition between the 1st and 2nd names. The algorithm has been run on 13,926 handwritten Japanese sample Kanji names, and it corrected 63.3 percent of the recognition errors and rejects in real time.

2. Error Correction for Handwritten Kanji Names Recognition

2.1 Japanese Kanji names and how to write them

Japanese Kanji names are written, for example, as shown in Fig. 1. People write an name as one continuous string or as a 1st and 2nd name with delimiting spaces. Fig. 1 shows the name written as one continuous string. When people write names on vouchers, they must obey a name-writing rule: that names be written as 1st and 2nd name with delimiting spaces in order to improve the precision of recognition.

Japanese write a name as one continuous string without delimiting spaces. It is therefore important to extract words from any position in the string. And as shown in Table 1,

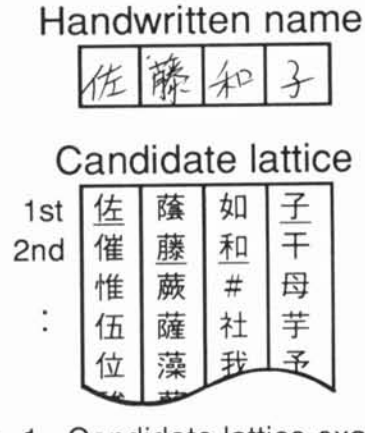

Fig. 1. Candidate lattice example.

Table1. Distribution of word lengths.

Length	1st name (words)	2nd name (words)
1	750	1,287
2	28,256	42,261
3	5,172	16,941
4	53	800
5	2	43
Total	34,233	61,332

Japanese Kanji names are short. For example, 1st and 2nd names have an average of 2.4 characters. There are also one-character names and two-characters names with only one different character, such as "大井" (Ooi) and "大田" (Ohta). When names are written as one continuous string, it is difficult to distinguish 1st and 2nd name such as "紀伊知子" ("紀伊" "知子" or "紀" "伊知子"). Finally, the Japanese language includes three kinds of character sets: Kanji, Hirakana, and Katakana. There are about 7,000 characters.

2.2 Target of error correction

The proposed algorithm recognizes Japanese names written on vouchers. A continuous name, for example, is scanned and up to K candidates for each character are output. For handwritten Kanji recognition, K is about 20. A candidate lattice is made from candidate characters (Fig. 1). The algorithm needs to accomplish the following three tasks. First, the word matching needs to extract the right names at any position from the candidate lattice by using a name knowledge base consisting of one hundred thousand names. Second, even if the right character is not among the candidate characters, the algorithm needs to acquire the right name in real time. Finally, even if names are not written in formal notation, the right name needs to be acquired.

3. Error Correction Algorithm

3.1 Outline of error correction

The proposed algorithm consists of a word matching function and an evaluation function (Fig. 2). The word matching function that we have already developed carries out high-speed automaton-type word matching by shifting the word matching position [10]-[12]. The evaluation function evaluates names by using a positional condition and penalties output at word matching. The general processing flow of this algorithm consists of four stages. First, candidate characters for a handwritten Kanji name are output. A candidate lattice is made from theses characters. In the second stage, word matching is carried out by shifting the word matching position. Only candidate names and their penalties are output. This penalty shows the ambiguity of a candidate name. In the third step, the evaluation function is carried out only when users write a name as one continuous string. In this function, the positional condition determines whether a candidate 1st name and a candidate 2nd name can be connected or not. Ambiguities of candidate names are calculated,and the right name is estimated according to the value of these ambiguities.

3.2 High-speed automaton-type word matching

A Kanji character is represented by 2 bytes. This algorithm transfers once for one character according to the code that has compressed a 2-byte code. This algorithm also extracts words at any position in the continuous string.

3.2.1 Principle of the word matching

The word matching consists of an automaton generator, a high-speed automaton-type word matching part, a name knowledge base, and a word search controller (Fig. 2). The general processing flow consists of four stages. First, a handwritten string is scanned and a Kanji recognizer outputs up to K candidate characters for each character. A candidate lattice is made from these candidate characters. In the second stage, the automaton generator generates a high-speed FSA from the candidate lattice. In the third stage, the word search controller searches for words included in a name knowledge base based on candidate characters, and when it finds them they are input to the high-speed automaton-type word matching part. Then word matching is carried out by using a word extracting method. Finally, only candidate words are output.

The high-speed FSA is made from the candidate lattice. A state transfers once for each character making of an input word. As shown in Fig. 3, the FSA has (written characters + 1) states and (candidate characters + 1) paths. The penalty and the compressed code of the candidate character are given to each path. The state number corresponds to the written position number. When a compressed code U_i is input to a starting state, a state transfers from the state to next state through path U_i. At the same time, the penalty P_i is read. This penalty is assigned according to the order of the outputs of the character recognition. If the character corresponding to U_i is not among the candidate characters, the state transfers through another path. In this case, the penalty P_{other} is read. That is, the right word can be extracted even if the right character is not among the candidate characters. The

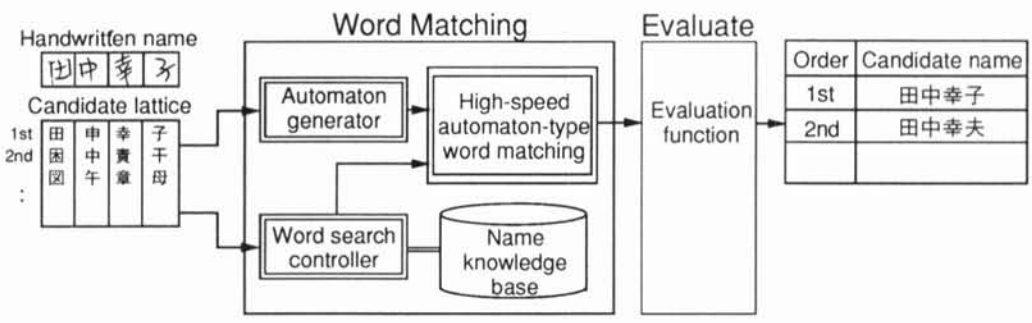

Fig. 2. Outline of the error correction algorithm.

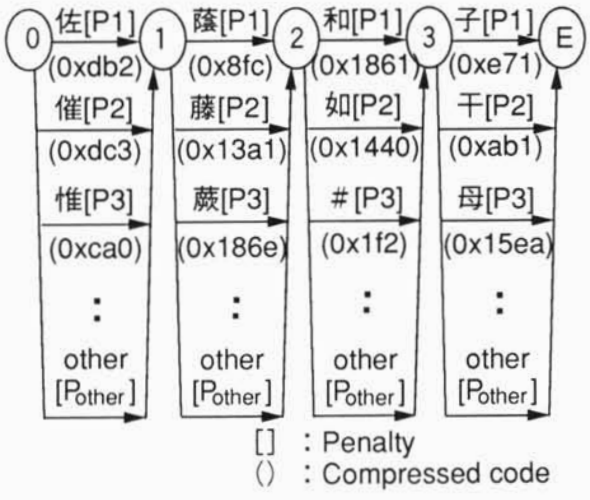

Fig. 3 Representation of high-speed FSA.

character recognition outputs candidate characters in order of highest reliance. So the smaller the penalty, the higher the reliance of the candidate character.

In the word matching process, the state transfers from the starting state to the next state after each compressed code making of a word is input to the FSA. The penalty is read from the FSA and accumulated whenever a state transfers. This process is repeated until the end of the word. The accumulated penalty represents the ambiguity of the word. The word is regarded as a suitable word only when the accumulated penalty is smaller than some specified value. When the accumulated penalty is larger than this value, the word matching is cancelled. The larger the value set, the higher the reliance of the word matching. Processing time, however, becomes excessive and the number of unsuitable words increases. The value therefore needs to be set by trading off between the processing time and recognition rate.

This algorithm searches for words within a large knowledge base by using candidate characters [11],[13]. The knowledge base is constructed from index tables and a word table (Fig. 4). Words included in the word table are sorted by the same key character and searched for by using index tables. It is supposed that the Pth character of words is regarded as the key. When the Pth character is among the candidate characters, the right word is obtained. That is, the right word can be obtained even if the right character is not among the candidate characters output by the OCR. In addition, the same words are sometime searched for according to different keys at different points. This has to be avoided. The word table is sorted by the 1st key. The words having the same Pth key are linked by the Pth key pointer. The word search procedure consists of the following four steps. For this explanation one word is written on a sheet and P is 2. First, candidate characters are regarded as the 1st key. Words I $\{w_j, w_k, w_l, \bullet\bullet\}$ are searched for at position1 by using the 1st index table, and are loaded into a searched word table. In the second step, words II $\{w_i, w_k, w_p, \bullet\bullet\}$ are still searched for at (position1 + 1) by the 2nd index table. In the third step, these words are checked to see whether they have already been searched for. The check process is:

IF the 1st character of words II is a candidate at position1
THEN
    Search for the next word.
ELSE
    Load the word. The word has not yet been searched for.

Finally, only the words II $\{w_i, w_p, \bullet\bullet\}$ that have not yet been searched are loaded into the searched word table. By repeating this procedure until the end of a state, it is easy to apply the word extracting method.

3.2.2 Word extraction at any position

This algorithm extracts words at any position in the continuous string [11]-[12]. The principle of this procedure is that a starting address controls the state that starts the word matching (Fig. 5), and the word matching is carried out. Thus, it is possible to extract words written at any position. This procedure has two steps. First, the starting address is set to a starting state, and the high-speed automaton-type word matching is carried out. Only suitable words are output to the candidate file. Second, the starting address is shifted by one state and the word matching process is repeated. This second procedure is repeated until the starting address reaches the end of the state. Word matching is carried out only when the following condition is satisfied: the length of an input word is less than the number of states that the word matching has not yet started.

3.3 Evaluation of candidate names

When a name is written as one continuous string, the right name is evaluated according to the positional information and the penalties at word matching. The evaluation function has four procedures (Fig. 6). First, candidate 1st names and candidate 2nd names are compounded. In the second step, the positional condition between candidate 1st names and candidate 2nd names is

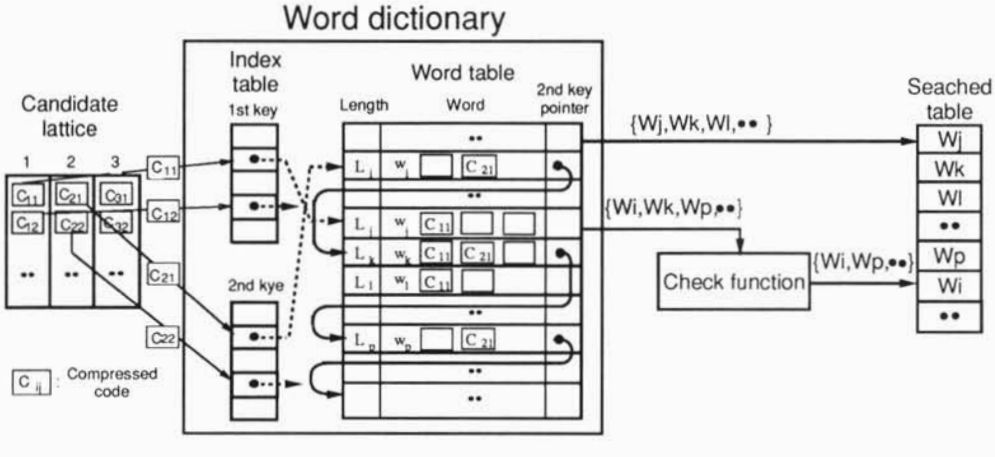

Fig. 4. Representation of a knowledge base for word searching.

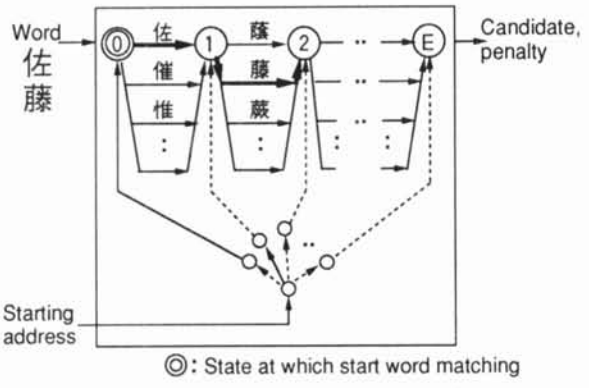

Fig. 5. Word extraction at any position.

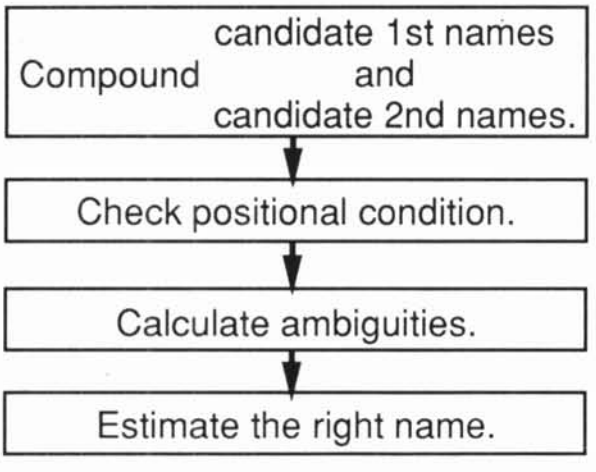

Fig. 6. Evaluation of candidate name for one continuous string.

checked. As a result, 1st and 2nd names are selected. In the third step, ambiguities of one continuous name are calculated by using the penalties of the 1st and 2nd names. Finally, the right name is estimated from the value of these ambiguities.

4. Experimental results

4.1 A method of experiments

We implemented this algorithm at a workstation as software written in C language. The workstation used in the experiments a 32-bit CPU (MC68020, 20 MHz) and a 16-Mbyte main memory. The processing faculty of this workstation was about 1 MIPS.

We verified the effectiveness of this algorithm by using 13,916 samples names. Both the 1st names and 2nd names had the average of 2.4 characters. The number of candidate characters was 15, so the candidate lattice was a 15*L matrix (L is the length of an input name.) The name knowledge base include 34,233 1st names and 61,332 2nd names (Table 1). Two percent of all 1st and 2nd names were one-character names; the other 98 percent were names with more than two characters.

4.2 An example of error correction

An example of an error correcting result is shown in Fig. 7. This figure shows a handwritten name "田中幸子" (Tanaka Sachiko), a candidate lattice, and a processing result. Character recognition outputs the right character for

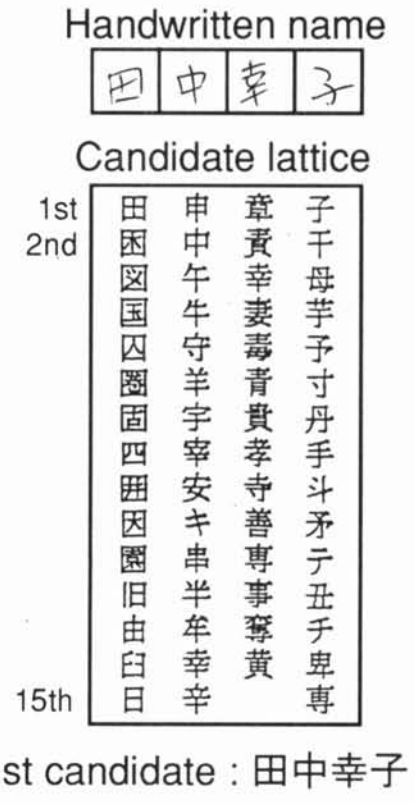

Fig. 7. Corrected example.

the 2nd written character "中" (naka) in the 2nd order, for the 3rd written character "幸" (sachi) in the 3rd order, and some right characters as the first candidate. The proposed algorithm acquired the right 1st and 2nd name as one continuous string.

4.3 Correction rate

The correction rate is the rate of correcting recognition errors and rejects. We used unconstrained handwritten samples when measuring the correction rate of the 1st and 2nd names. Fig. 8 shows the relation between the correction rate and the order of the candidate characters. For the 1st name, 62.8 percent of the recognition errors and rejects were corrected, and for the 2nd name 63.8 percent of the recognition errors and rejects were corrected. An average of 63.3 percent of the recognition errors and rejects in the first order were corrected.

4.4 Processing time

We measured the processing time of 1st and 2nd names and the total processing time per Japanese sample name (Table 2). The total processing time was 116.9 msec, which is fast enough to be practical. The high-speed automaton-type word matching acquires names by using candidate characters. The time it requires depends on the number of candidate characters, so the proposed algorithm is faster if the character recognition outputs fewer candidates.

5. Conclusion

We proposed an error correction algorithm for that uses a name knowledge base handwritten Kanji name recognition. This algorithm uses high-speed automaton-type word matching based on the shifting word matching position. Users can write names on vouchers without being conscious of

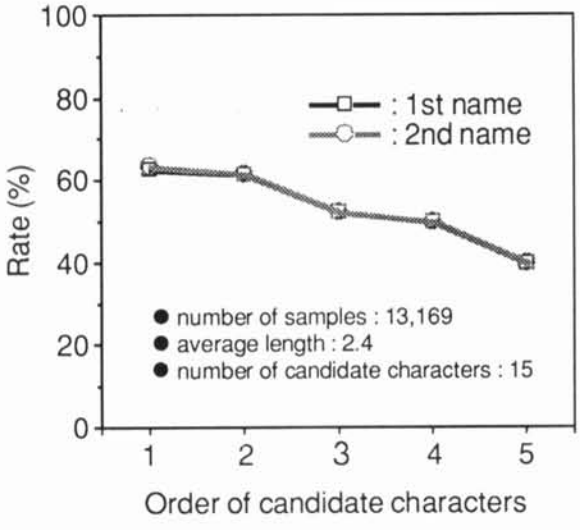

Fig. 8. Correction rate for Kanji names.

of formal notation rules. Experimental results show that 63.3 percent of the character recognition errors and rejects from 13,916 handwritten Japanese sample names were corrected in real time.

Table 2. Processing time.

Procedure	Measured time (msec)
First name	65.5
Second name	51.4
Total	116.9

REFERENCES

[1] K. Seino, Y. Tanabe and K. Sakai, "A Linguistic Post Processing based on Word Occurrence Probability, *"Proc. 2nd Int. Workshop on Frontiers in Handwriting Recognition*, pp. 191-199 (1991).

[2] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "An Error Correction Algorithm for Handwritten Chinese Character Address Recognition, *"Proc. ICDAR 91 1st Int. Conf. on Document Analysis and Recognition*, pp. 916-924 (1991).

[3] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "A Resolution of Input String Ambiguity by Using Address Knowledge base, *"National Convention Record of Japanese Inst. of Electron. Infor. and Comm. Eng.* , 6-51, (1991, in Japanese).

[4] E. M. Riseman and A. R. Hanson, "A Contextual Postprocessing System for Error Correction Using Binary N-grams, *"IEEE Trans. on computer*, Vol. C-23, No. 5, pp. 480-493 (May 1974).

[5] T. Kawada, S. Amano, and K. Sakai "Linguistic Error Correction of Japanese Sentences, *"Proc. COLING 80*, pp. 257-261 (1980).

[6] T. Sugimura, "Error correction method for character recognition based on confusion matrix and morphological analysis, *"Trans. of Japanese Inst. of Electron. Infor. and Comm. Eng.* , Vol. J72-D-II, pp. 993-1000 (1989, in Japanese).

[7] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "Automaton-type Word matching for Character input systems, *"National Convention Record of Japanese Inst. of Electron. Infor. and Comm. Eng.* , 6-80, (1990, in Japanese).

[8] W. A. Wulf, M. Shaw, P. N. Hilfinger, and L. Flon, "Fundamental Structure of Computer Science, "ADDISON WESLEY (1981).

[9] A. V. Aho and M. J. Corasick, "Efficient String Matching, *"Com. of ACM*, Vol. 18, No. 6, pp. 333-340 (1975).

[10] K. Marukawa, M. Koga, Y. Shima and H. Fujisawa, "A High Speed Algorithm for Automaton-type Word Matching", *National Convention Record of Information Processing Society of Japan*, 2-135, (1990, in Japanese)

[11] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "A High Speed Word Matching Algorithm for Handwritten Chinese Character Recognition, *"Proc. Int. Workshop on Machine Vision Applications (MVA-90)*, pp. 445-449 (1990).

[12] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "A Word Extraction Method for Ambiguous Input String by Using Automaton-type Matching, *"National Convention Record of Japanese Inst. of Electron. Infor. and Comm. Eng.* , 6-110, (1992, in Japanese).

[13] Y. Iida and T. Sugimura, "A Study of word matching method with Dictionary for pattern recognition, *"Japanese Inst. of Electron. Infor. and Comm. Eng.* , PRL82-77, pp. 93-98, (1982, in Japanese).