MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

A HIGH SPEED WORD MATCHING ALGORITHM FOR HANDWRITTEN CHINESE CHARACTER RECOGNITION

Katsumi Marukawa, Masashi Koga, Yoshihiro Shima, and Hiromichi Fujisawa

Central Research Laboratory, Hitachi, Ltd.
1-280 Higashikoigakubo Kokubunji-shi Tokyo, 185 JAPAN
(Tel 0423-23-1111, E-mail: marukawa@hcrlgw.crl.hitachi.co.jp)

ABSTRACT

A new high speed word matching algorithm for handwritten Chinese character recognition is presented. A continuous string without delimiting space is recognized in real time by using this algorithm. Errors and rejects of an optical character reader are corrected to meaningful characters with the help of a dictionary. This algorithm uses a finite state automaton (FSA); the FSA's transition table is condensed by the representation device of the FSA to reduce the processing time of table generation and state transitions. Words can be extracted from any position in the continuous string. The recognition process is 6.8 times faster than a conventional automaton-type algorithm. The algorithm was run on 5,032 handwritten Japanese sample addresses. In the experiment, 96.6 percent of the recognition errors and rejects were corrected in real time.

1. Introduction

It is desirable for a data input system to have character recognition, speech recognition, and so on [1][2][3][4][5][6]. In a data input system, word matching plays an especially important role in automatically recognizing large input data. It has two objectives. First, in order to be practical, it must be much faster. The other objective is to resolve ambiguities of character recognition errors and rejections.

Some word matching algorithms have already been developed [7][8][9]. However, these were designed to recognize English letters. Word matching algorithms for Chinese character recognition are classified into two categories. One category is (a) a compound word matching [10], and the other category is (b) an automaton-type word matching [11]. In category (a), a string is compounded of candidate characters. If the string exists within a dictionary, the string is regarded as a candidate word. When a continuous string consists of multiple words, this algorithm creates huge strings. Therefore, it is very time-consuming and is not suitable as the algorithm that processes a continuous string. On the other hand, algorithm (b) uses an FSA [12][13]. The FSA is made from candidate characters. Words to input to the FSA are searched from within the dictionary according to candidate characters. As a result algorithm (b) is faster than algorithm (a). But in order to be practical it is necessary to improve algorithm (b) to process at still higher speeds.

This paper proposes a high speed word matching algorithm that improves algorithm (b). In the proposed algorithm, an FSA with penalties is used, and it is condensed by the representation device of the FSA to reduce the processing time for table generation and state transitions. Also, words at any position are extracted out of the continuous string in real time. The effectiveness of the proposed algorithm was ascertained by an experiment on 5,032 handwritten Japanese addresses.

2. Word matching problem for handwritten Chinese characters

2.1 Word matching target

Japanese write a sentence like one continuous string without delimiting spaces. Therefore it is important to extract words from any position in the string. Also, the Japanese language includes three kinds of character sets: Chinese characters, Hirakana and Katakana. There are about 8,000 characters.

A continuous string written on a voucher for example, is scanned. Up to K candidates for each character are output. K is about 20 in the case of handwritten Chinese character recognition. Then a candidate lattice is made from candidate characters as shown in Fig. 1. For length S of the string, it becomes a K•S matrix. The word matching algorithm needs to extract the right words from this lattice in real time with the help of a dictionary consisting of tens of thousands of words. Also, the character recognition dose not always output a right character into candidates. So, even if several correct characters don't exist among the candidates, the right words need to be extracted from this lattice.

2.2 Conventional word matching

As mentioned earlier, the approaches to word matching for handwritten Chinese characters fall into two categories. First, the compound algorithm makes a string to compound candidate characters. If the string exists in the dictionary, the string is regarded as a candidate word. 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 [10], this algorithm has no problem. But when K is approximately 20, it is very time-

Fig. 1. Candidate lattice example

consuming. Therefore, this algorithm is inadequate. In the automaton-type algorithm [5], on the other hand, as one Chinese character is represented by 2 bytes, a state transfers two times for one written character according to an upper-byte and a lower-byte of each character constructing a word as shown in Fig. 2. First, a state transfers from state P to one state of Q_1, Q_2, ••, Q_K, Q_{other} according to the upper-byte (U_1, U_2, ••, U_K), or other. Other is a byte separate from the upper-bytes. Secondly, a state transfers from the transferred state to state R according to the lower-byte (V_1, V_2, ••, V_K) or other. At the same time, the penalty(PL) is read from the FSA and accumulated. The FSA is represented by using a transition table and a penalty table. The transition table shows the next transition state. The penalty table shows the transition's penalty. These tables are $256(=2^8)$ raws by (K+1)•S columns as shown in Fig. 3. Each element is one byte.

3. High speed word matching algorithm

3.1 System architecture

The proposed system consists of a character recognition part, an automaton generator, a high speed word matching part, a word dictionary, a word search controller and a candidate file as shown in Fig. 4. The general processing flow consists of four stages. First, a handwritten string is scanned, and a character recognition part outputs up to K candidate characters for each character. A candidate lattice is made from candidate characters. In the second step, the automaton generator generates a high speed FSA from the candidate lattice. In the third stage, the word search controller searches words included in a word dictionary based on candidate characters, and they are input to the high speed word matching part. Then, word matching is carried out by using a word extracting method. Finally, only candidate words are output to the candidate file.

3.2 Principle of algorithm

This algorithm transfers once for one character according to the code that has compressed a 2-byte code. The processing time of the table generation and state transitions is reduced by the representation device of the FSA and how a candidate word is estimated. Also, this algorithm extracts words at any position of the continuous string.

3.2.1 Representation of FSA

The high speed FSA is made from the candidate lattice. A state transfers once for each character constructing an input word. The FSA has (written characters+1) states and (candidate characters+1) paths as shown in Fig. 5. The penalty and the compressed code (U) 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 state P, a state transfers from state P to state Q through path U_i. At the same time the

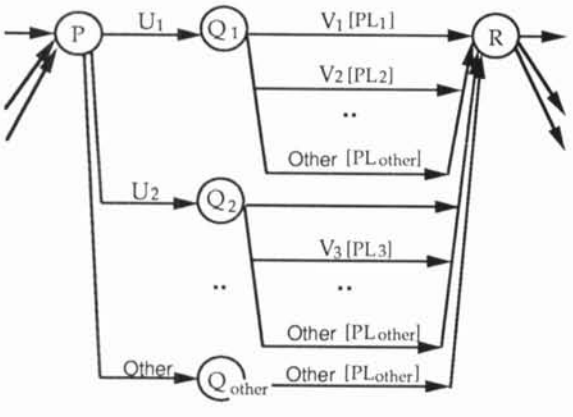


Fig. 2. Conventional FSA

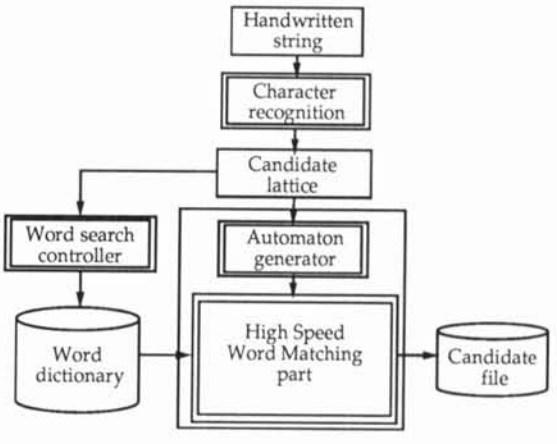
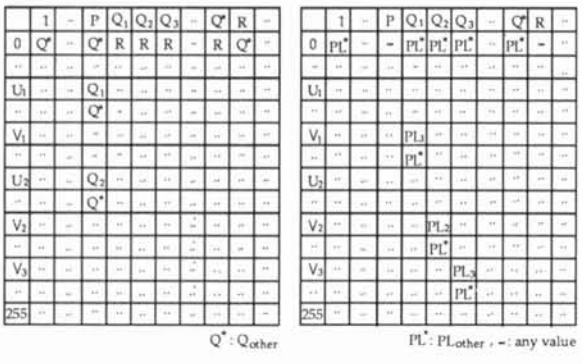


Fig. 4. System architecture



Q*: Q other

PL*: PL other , -: any value

Fig. 3. FSA table of conventional automaton-type

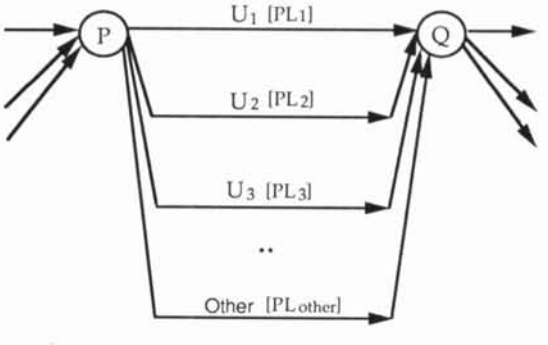


Fig. 5. High speed FSA

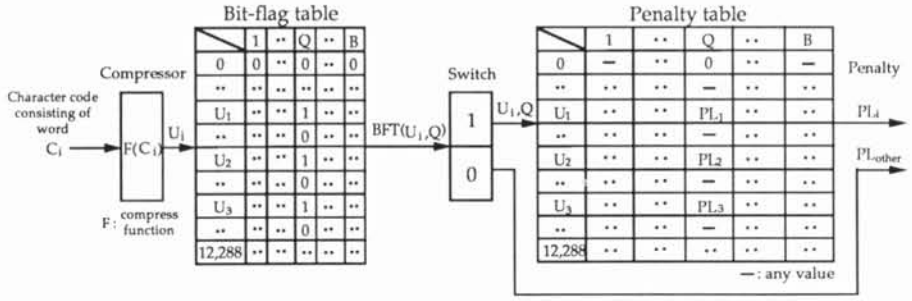


Fig. 6. Processing flow of penalty calculation

penalty PL_i is read. This penalty is assigned according to the order of the outputs of the character recognition. The 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, a state transfers from the starting state to the next state after each compressed code constructing 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. Only when the accumulated penalty is smaller than some value, is the word regarded as a suitable word and output to the candidate file. When the accumulated penalty becomes larger than the value, the word matching is cancelled. The larger the value set, the higher the reliance of the word matching. However, processing time is prohibitive and the number of unsuitable words increases. So, the value needs to be set by a trade off between the processing time and recognition rate.

This algorithm's FSA uses a bit-flag table which improves the transition table and the penalty table. The bit-flag table (BFT(i,j)) has the compressed code (i) as row and position number (j) as column. The bit-flag table and the penalty table are represented as shown in Fig. 6. The general process of the word matching consists of a table generation and a penalty calculation. The table generation process consists of 2 steps. First, all elements of the bit-table are cleared. Second, flags are set only at elements where candidate characters exist. For example, when there are compressed codes $U_1, U_2, \bullet\bullet, U_K$ of candidate characters at position Q, flags of $BFT(U_1,Q)$, $BFT(U_2,Q)$, $\bullet\bullet$, $BFT(U_K,Q)$ are set. As each element is represented by 1 or 0, this table is represented by a bit and condensed. Therefore this table is cleared by a used register at the same time. The penalty table (PLT(i,j)) is generated by a byte. As described below, as this table is accessed according to the flag of the bit-flag table, it does not need to clear it. The processing time of the table generation is expressed as follows.

$$T1 \bullet S \qquad (1),$$

where T1 is the processing time per written character.

Also, the process of penalty calculation is described as follows.

IF $BFT(U_i,Q) = 1$ THEN

$PLT(U_i,Q)$ is access and $PLT(U_i,Q)$ is accumulated.

ELSE

PL_{other} is accumulated. Namely U_i does not exist among the candidates at position Q.

That is, the penalty is accessed from the penalty table only when the flag of the bit-flag is set as shown in Fig. 6. The processing time of the penalty calculation is expressed as follows.

$$T2 \bullet L \bullet N \bullet K \qquad (2),$$

where T2 is the processing time per written character and N is the number of words searched per candidate character.

3.2.2 Word extracting method based on shifting word matching position

This method extracts words at any position of the continuous string by using the high speed word matching algorithm. The principle of the method is that a starting address controls the state which starts the word matching as shown in Fig. 7, and the word matching is carried out. Thus, it is possible to extract words written at any position. This method has two procedures. First, the starting address is set to an initial state. The high speed 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. The latter procedure is repeated until the starting address reaches the end of the state. Only when the following condition is satisfied, is the word matching carried out. The condition is that the length of an input word is smaller than the number of states that the word matching has not yet started. So the time (T) to process the continuous string is expressed as follows.

$$T=T1 \bullet S+T2 \bullet L \bullet N \bullet K \bullet S \qquad (3)$$

3.3 Word search method using bit-flag table

We propose a word search method to improve the conventional method [8]. The conventional method searches words from within a large dictionary by using candidate characters. The dictionary is constructed from index tables and a word table. Words included in the word table are sorted by the same key character and searched by using index tables. It is supposed that the Pth character of words is

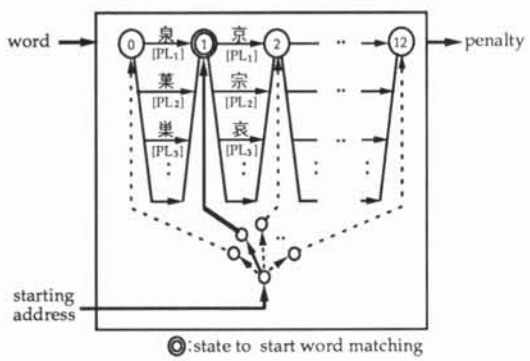


Fig. 7. Word matching control for word extraction

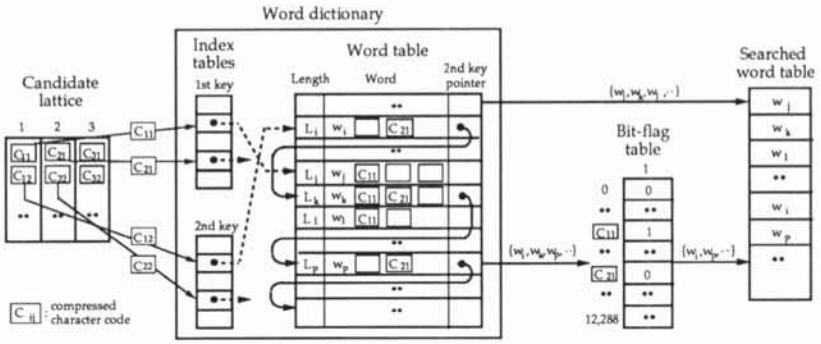


Fig. 8. Dictionary representation for word search

regarded as the key. When the Pth character is included in candidate characters, the right word is obtained. In the conventional method, there are cases that the same words are searched by plural keys at different points. When such a syntax analysis is postprocessed, it is undesirable to include the same words in the candidate file.

The proposed word search method avoids this by using the bit-flag table shown in Fig. 8. The word table is sorted by the 1st key. The words having the same Pth key are linked by the Pth key pointer. The method consists of the following 4 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 {w_j, w_k, w_l, ••} are searched by using the 1st index table, and are loaded into a searched word table. In the second step, words {w_i,w_k,w_p, ••} are still searched by the 2nd index table. In the third stage, these words are checked by using the bit-flag table whether they have been already searched or not. The check process is described as follows.

IF BFT(F(C) , 1) = 1, THEN

Search the next word. The word has already been searched.

ELSE

Load the word. The word has not been searched yet.

Here, F is a compress function, and C shows the 1st character of the word that is searched by the 2nd key.

The reason why the bit-flag table is used, is that the compressed code of the 1st character of the word that is searched by the 1st key corresponds to the set flag. This is because the bit-flag table is made from candidate characters and words that are searched by the 1st key are based on candidate characters. Finally, only words {w_i, w_p, ••} that haven't been searched yet are loaded to the searched word table. By repeating this procedure until the end of a state, it is easy to apply the word extracting method.

4. Experimental results

We implemented this high speed word matching algorithm in C language at a workstation. The workstation used in the experiments, has a 32bit cpu (MC68020, 20MHz) and a 16 Mbyte main memory.

4.1 Processing time

We measured the table initialization/creation time and the penalty calculation time per written character for the conventional automaton-type and the high speed word matching as shown in Table 1. Also, our measurements indicated that about 42 words (N) are searched per candidate character and the length (L) of the words is 3 characters on the average for Japanese addresses. We supposed that 32 characters (S) are written as a continuous string and K is 15. We estimated the ratio of the processing time of the conventional automaton-type to that of the high speed word matching. As mentioned above, the processing time is expressed as in equation (3). In the case of the conventional automaton-type, t1 and t2, as shown in Table 1, are substituted in T1 and T2 of equation (3). Its processing time is 5020.2 msec. Also, that of the high speed word matching is 743.7 msec to substitute t3 and t4 in T1 and T2 of equation (3). Thus, the ratio is estimated at 6.8, that is, the high speed word matching algorithm is 6.8 times faster than the conventional automaton-type algorithm.

4.2 Application to handwritten Japanese addresses

We verified the effects of the proposed algorithm when applied to handwritten Japanese addresses. A Japanese address consists of geographical names arranged in a hierarchical structure listing prefecture, city, and town in order. The dictionary consists of four hierarchical tables and two index tables for each. The hierarchical tables consist of geographical names and links. The ambiguity of a candidate string is calculated by using the penalty of extracted words. The written address is recognized by this ambiguity value. The algorithm was run on 5,032 sample addresses. The average length of the address was 12 characters. As a result, the average processing time was 341.52 msec. This processing time was near the estimated value based on equation (3). Also Fig. 9 shows the example that "が"(ga) of a written city

Table 1. Processing time per written character

procedure	conventional automaton-type	high speed word matching
table initialization/ creation	30.25 (t1)	0.56 (t3)
penalty calculation	0.067 (t2)	0.012 (t4)

(unit : msec)

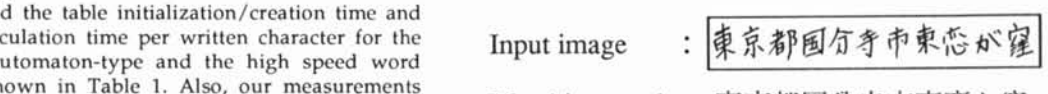


Fig. 9. Corrected example using this algorithm

name "東恋が窪"(higashi-koigakubo) was corrected to "ケ" of a dictionary word " 東恋ケ窪 " by the word matching algorithm. Fig. 10 shows the relation between a correction rate of the recognition errors and rejects and the oder of candidate characters. This figure indicates that the correction rate is independent of the character recognition rate. 96.6 percent of the recognition errors and rejects were corrected for each processed candidate character.

5. Conclusion

We proposed a high speed word matching algorithm for handwritten Chinese character recognition. This algorithm can recognize a continuous string of handwritten words without delimiting space in real time with the help of a word dictionary. The recognition process is 6.8 times faster than a conventional automaton-type algorithm. Also, 96.6 percent of the character recognition errors and rejects from 5,032 handwritten Japanese sample addresses were corrected in real time.

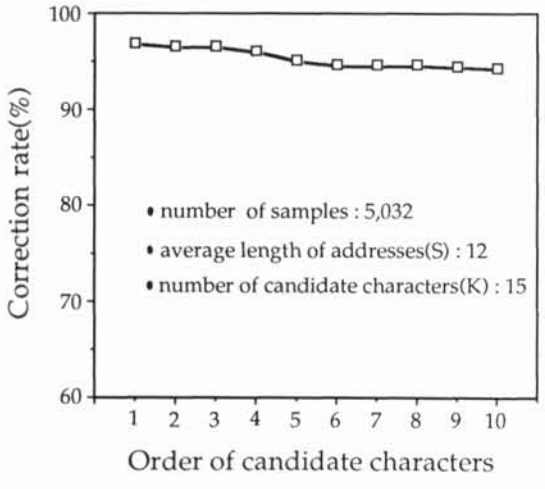


Fig. 10. Correction rate for Japanese addresses

REFERENCES

[1]J.Higashino, H.Fujisawa, Y.Nakano and M.Ejiri, " A knowledgebased segmentation method for document understanding",*Proc.8th Int. Conf. Pattern Recognition*, pp.745-748 (1986)

[2]Y.Shima, T.Murakami, J.Higashino, Y.Nakano and H.Fujisawa,"A Segmentation Method of Color Document Images for Multimedia Contents Retrieval", *Proc. RIAO 88*, Use-oriented Content-based Text and Image Handling (AFIPS), Cambridge, vol.2, pp.1001-1008 (March 1988)

[3]H.Fujisawa, H.Yashiro, T.Murakami, Y.Shima and Y.Nakano,"Document Analysis and Decomposition Method for Multimedia Contents Retrieval", *Proc. the Second Int. Symposium on Interoperable Information System (ISIIS '88)*, pp.231-238 (Nov. 1988)

[4]H.Yashiro, T.Murakami, Y.Shima, Y.Nakano and H.Fujisawa,"A New Method of Document Structure Extraction Using Generic Layout Knowledge", *Proc. Int. Workshop on Industrial Applications of Machine Intelligence and Vision (MIV-89)*, pp.282-287 (April 1989)

[5]Y.Shima, T.Murakami, M.Koga, H.Yashiro and H.Fujisawa,"A High Speed Algorithm for Propagation-type Labeling based on Block Sorting of Runs in Binary Images", *Proc. the 10th Int. Conf. on Pattern Recognition (10th ICPR)*, vol.1, pp.655-658 (June 1990)

[6]H.Fujisawa and Y.Nakano,"A Top-Down Approach for the Analysis of Document Images", *Proc. IAPR on Syntactic & Structural Pattern Recognition* , pp.113-122 (June 1990)

[7]E.M.Riseman and A.R.Hanson,"A contextual Postprocessing system for error correction Using binary n-Grums", *IEEE Trans. on computer*, vol.C-23, No.5, pp.480-493 (May 1974)

[8]E.Tanaka, T.Kohasiguchi and K.Shimamura,"High Speed String Correction for OCR", *Proc. 8th Int. Conf. on Pattern Recognition* (October 1986)

[9]W.H.Cushman, P.S.Ojha, and C.M.Daniels,"Usable OCR : What are the minimum performance requirements ?", *CHI '90 Proc. of ACM*, pp.145-151 (April 1990)

[10]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)

[11]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)

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

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

[14]Y.Iida and T.Sugiyama,"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)