Recognition of Numeric Strings with Notation Rules Using String Checking

Katsumi MARUKAWA, Kazutoshi TAKAKURA,
Taro HAYASHI, Masashi KOGA, and Yoshihiro SHIMA

Hitachi, Ltd., JAPAN
marukawa @ crl. hitachi. co. jp

Abstract

An algorithm for recognizing numeric strings with notation rules by using string checking has been developed. The proposed string check function removes extraneous characters from recognized character strings by using the notation rules. This function also determines whether to carry out recognition error correction. In this correction process, recognized characters are compared with strings in a dictionary. Errors in character strings are automatically corrected to meaningful letters by using the notation rules and dictionary. The string space of the dictionary to be compared is restricted based on the notation rules; this reduces processing time. The string check function improved the string recognition rate from 98.5 % to 99.7 %, and decreases the error rate by 98 %. Important meaningful strings, such as I.D. codes, written in a domain with neighboring characters can be recognized in real time by using this algorithm.

1. Introduction

In our information flooded society, paper processing has become a significant burden on companies and governmental agencies. Several approaches have been taken to converting information on paper into electronic form [1]-[3]. Optical character readers (OCRs) are able to recognize the contents of forms, such as an I. D. code, names, and addresses. Key data items are often used to control other data items. However, OCRs sometimes misread characters. Operators must therefore check for recognition errors and correct them, which can be an expensive process. A paper form processing system thus needs a function for correcting errors automatically with high reliability in real time.

Some of paper form processing systems already developed depend entirely on the OCR output. While some of these systems correct recognition errors for names and addresses [4]-[6], numeric strings such as I. D. codes, depend entirely on the OCR output. None of the developed systems have an error correction function for numeric strings. The string recognition rate is represented as P^L, where P is the recognition rate for characters and L is the string length. The string recognition rate decreases exponentially. Additionally, a conventional OCR can't recognize strings written in a domain with neighboring characters. These systems that recognize such strings as an I. D. code as a numeric string are thus unsuitable. These strings should be read as a word. The above tasks needs to be solved by utilizing sources of notation rules and a dictionary.

We propose a string check function that removes extraneous neighboring characters from recognized character strings based on notation rules. It also determines whether error correction is required. Correction is done by comparing recognized character strings with strings in a dictionary. This function was determined to be effective by conducting experiments using a sample set of 3,983 input pages.

2. Numeric String Recognition

2.1 Objectives

We developed our string check function with two objectives in mind. As shown in Fig. 1, a string is a set of characters and delimiters. When a string is regarded as a set of recognized characters, the recognition rate is represented as P^L. The recognition rate for strings decreases exponentially. However, some strings, such as I.D. codes, have an important meaning as a particular combination of characters. Our first target was therefore to improve the

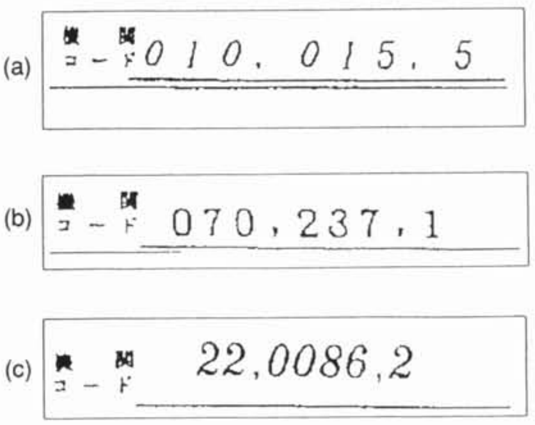


Figure 1: Samples of numeric strings in a domain with neighboring characters.

recognition rate for strings by regarding each string as one meaningful word.

A problem with conventional OCR is that other characters are written near the objective string, the recognition rate drops. Also, conventional OCR recognizes characters by using format information to specify character position. In short, conventional OCR has trouble recognizing strings written in a domain with neighboring characters or written at a slipped position. Our second target was to improve the recognition rate for these problem strings.

2.2 Notation rules

We use notation rules to specify the number of characters included in a string and how to handle such delimiters as hyphens. Two example rules are shown in Fig. 2. Using notation rules improves the recognition rate and increases the processing speed. As discussed above and as shown in Fig. 1, a string can be written close to other characters (Fig. 1 (a)) or written at a slipped position (Fig. 1 (c)). If strings have notation rules, extraneous neighboring characters are easily searched for and removed by using the notation rule delimiters. The string recognition rate is thus improved. If a set of strings is divided according to the notation rules, as shown in Fig. 3, dictionary strings can be divided into groups based on the notation rules. The number of dictionary strings to be processed is thus reduced, which reduces the processing time.

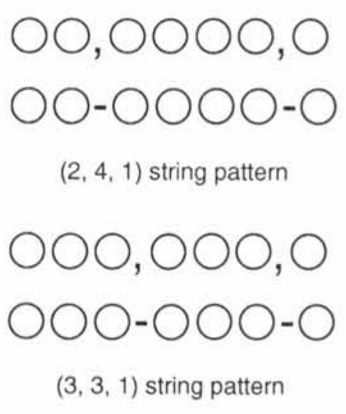


Figure 2: Example notation rules.

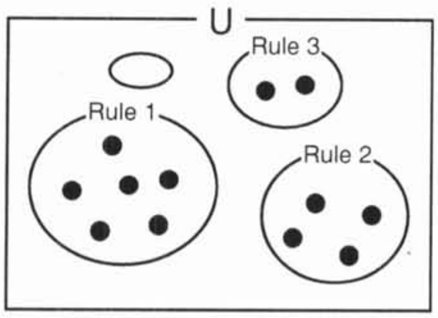


Figure 3: Sets of notation rules.

3. Numeric String Recognition Using Notation Rules

The proposed system automatically corrects recognition errors to meaningful letters by using notation rules and a dictionary. The system has four stages (Fig. 4). First, the OCR scans the page and extracts a partial image, including the numeric string. The characters in the extracted image are then segmented. Next, each segmented character image is recognized and the recognized result is output. In the final stage, the string check function removes extraneous neighboring characters and corrects any recognition errors in the recognized string by using the notation rules and the dictionary. The recognized string or reject information is then output.

4. String Check Function

4.1 Outline

The string check function consists of three modules (Fig. 5). The first module removes any extraneous characters from the string based on the notation rule. If the recognized string does not have the right notation, the string check function rejects the page. If it has the right notation, the second module is activated; it determines whether there are any recognition errors by searching for the recognized string in the dictionary. If the recognized string is found in the dictionary, the string check function accepts the string. If the recognized string is not found in the dictionary, the final module is activated to correct the errors. The final module corrects the errors by using string matching.

4.2 Removal of extraneous characters

Extraneous characters are determined by the relative positions of the characters and delimiters based on the notation rules. There are three steps in this process (Fig.

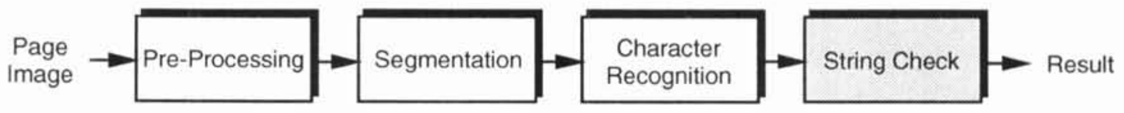


Figure 4: Outline of numeric string recognition system.

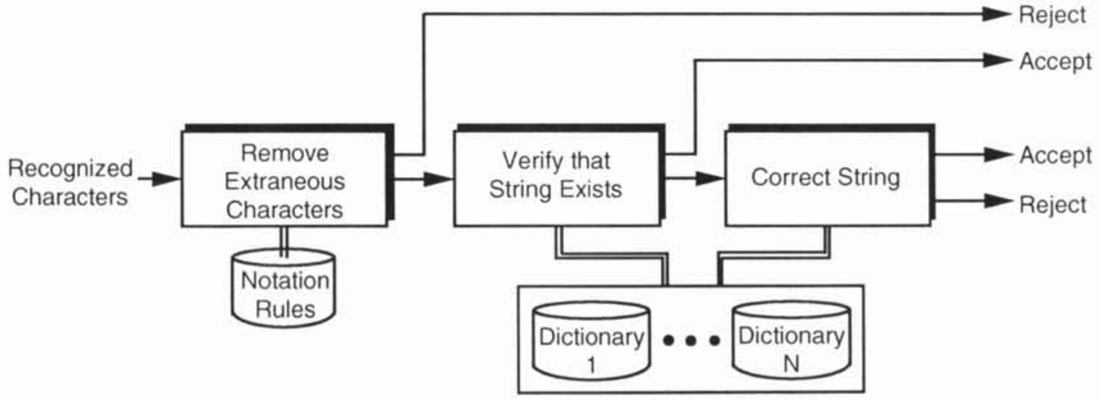


Figure 5: Architecure of string check function.

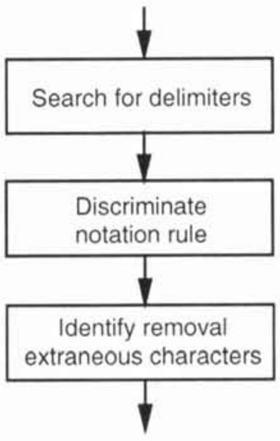
6). First, the delimiters are identified. Next, the notation rule is discriminated according to the relative positions of the characters and delimiters. The notation rule is discriminated according to the number of delimiters and the number of characters between delimiters. In the final step, the extraneous characters are identified based on the notation rule and removed. If the recognized string does not have the right notation, the string check function rejects the page. If the recognized string has the right notation, the second module is activated.

4.3 Verification of string existence

The second module verifies string existence in order to improve processing speed. It determines if it is necessary to run the procedure to correct recognition errors by using the dictionary. When strings are represented by several notation rules, the dictionary strings are grouped according to the notation rules, as shown in Fig. 7. The module searches for the recognized string only in the dictionary section corresponding to the recognized string's notation rule. In other words, the dictionary consists of several string tables, and the appropriate string table is determined according to the discriminated notation rule and the recognized string is searched for in the divided string table. If the recognized string is found in the string table, the string is regarded as a suitable string. The string check function then finishes and accepts the string. When the recognized string is not found in the string table, the string is regarded as a non-suitable string. The final module is then activated to correct the recognition errors.

4.4 String correction

The last module corrects recognition errors by using string matching. The recognized strings are compared with strings in the dictionary. The string matching is done in the divided string table space according to the recognized string's notation rule, as explained above. The degree of matching between a string and each string in the dictionary is calculated character by character. A finite state automaton (FSA) is generated from the recognized characters. It has (the number of characters+1) states and two paths ("a



Search for delimiters
Discriminate notation rule
Identify removal extraneous characters

Figure 6: Removal of extraneous characters.

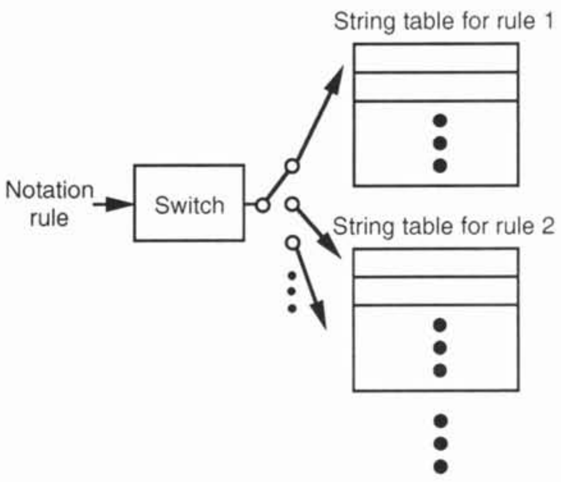


Figure 7: Selection of string table using notation rules.

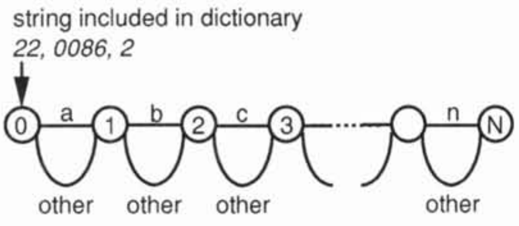


Figure 8: String matching process.

recognized character" and "other"), as shown in Fig. 8. The state moves from the starting state to the next state after each character has been input to the FSA. The degree of matching is calculated at each state transfer. When the input character is not a recognized character, the state moves along the path denoted "other". This process continues until the last character in the string is input. The degree of matching represents the ambiguity of the input string. The string with the highest value is taken as the correction, if its value is above the threshold. If it is below the threshold, reject information is output.

5. Experiment

5.1 Method

We implemented this algorithm on a personal computer with software written in the C language. The computer had a 32-bit CPU (clock speed of 20 MHz). Two kinds of notation rules were used and 3,983 input pages were tested. The dictionary included 1,238 strings; each string consisted of seven characters and two delimiters. The strings were written in domains with neighboring characters.

5.2 Results

As shown in Table 1, using the proposed function increased the number of correct strings from 3,922 to 3,969, reduced the number of error strings from 48 to 1, and increased the number of rejected strings from 4 to 13. As a result, the string recognition rate improved from 98.5 % to 99.7 %. The string error rate dropped from 1.4 % to 0.1 %, a decrease of 98 %. The string reject rate increased from 0.1 % to 0.2 %. The processing time was 9.2 msec per page, which is fast enough for practical use. This algorithm thus substantially decreased processing time and improved the recognition rate.

6. Conclusion

A recognition algorithm was proposed for numeric strings with notation rules that uses a string check function. This function removes extraneous characters from recognized strings based on the notation rules. Errors in optical character recognition are automatically corrected to meaningful letters by using the notation rules and a dictionary. Testing on 3,983 pages showed that the recognition rate for strings improved from 98.5 % to 99.7 %, and the error rate was decreased 98 %.

The tested strings were I. D. codes, such as these used by companies and governmental agencies. Future research should be directed to enhancing the error correction algorithm to deal with other strings, such as personal codes.

Table 1: Effectiveness of using string check function.

unit: string

String check	Correct	Error	Reject	Total
No	3,931 (98.5%)	48 (1.4%)	4 (0.1%)	3,983
Yes	3,969 (99.7%)	1 (0.1%)	13 (0.2%)	3,983

References

[1] K. Wong, R. Casy, and F. Walh, "Document analysis system," *IBM Journal of Research and Development*, Vol. 26, No. 6, pp. 647-656 (1982).

[2] R. Casy, D. Ferguson, K. Mohiuddin, and E. Walach, "Intelligent Forms Processing System," *Machine Vision and Applications*, Vol. 5, No. 3, pp. 143-156 (Summer 1992).

[3] S. L. Taylor, R. Fritzson, and J. A. Pastor, "Extraction of Data from Printed Forms," *Machine Vision and Applications*, Vol. 5, No. 3, pp. 211-222 (Summer 1992).

[4] K. Marukawa, K. Nakashima, M. Koga, Y. Shima, and H. Fujisawa, "A Paper Form Processing System with an Error Correcting Function for Reading Handwritten Kanji Strings," *Proc. 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 469-482, (1994).

[5] 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).

[6] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "A Post-processing Method for Handwritten Kanji Name Recognition Using Furigana Information," *Proc. ICDAR 93 Int. Conf. on Document Analysis and Recognition*, pp. 218-221 (1993).