Context Processing to Read Text on Damaged Wooden Tablets

Akihito Kitadai, Kazu Nishijima, Kei Saito, Masaki Nakagawa, Hajime Baba,
Akihiro Watanabe

Context Processing to Read Text on Damaged Wooden Tablets

Akihito Kitadai   *Kazu Nishijima*   *Kei Saito*
Tokyo University of Agriculture and Technology
{ak, kazu, kei}@hands.ei.tuat.ac.jp

Masaki Nakagawa   *Hajime Baba*   *Akihiro Watanabe*
Tokyo University of   National Research Institute for
Agriculture and Technology   Cultural Properties, Nara

Abstract

This paper describes context processing to present candidates for damaged scripts on wooden tablets (mokkans). Since mokkans excavated from old strata have been damaged, even archeologists can hardly read scripts on mokkans. Very often, ink in several areas are faded out or completely lost, some characters might be misrecognized based on which other characters must be read. The context processing extends the Aho-Corasick method to allow self-transition and presents candidates even for scripts with lost ink and misrecognized characters. For evaluation, we employed 4,041 place names in Japan at the 8th century as the vocabulary. Each place name consists of 9 to 11 characters. Test keywords were prepared with 1 to 6 characters lost and 0 to 2 characters replaced by others from the vocabulary. Even for those with 5 characters lost and one character is replaced, the method nominates correct names in the top 10 candidates with 71.7% correctness.

Keywords: Context analysis, Historical document, Information retrieval method, Aho-Corasick method.

1. Introduction

Since reading and analyzing descriptions in historical documents extends our historical knowledge, archaeologists have been trying to read and decode historical documents all over the world. "Mokkan" is a Japanese generic name to call a wooden tablet on which text is written by a brush in India ink. Since wooden tablets were more accessible than other media to record handwriting and they were enough weatherproof, ancient people used many mokkans for recording and distributing the text information in Japan. We have now more than 170,000 mokkans excavated from the ruins of *the Heijo palace site*, the capital of Japan in the Nara period from A.D. 710 to 794. The number of excavated mokkans is increasing as excavation in the larger remaining part in the Heijo palace site and other ruins continues.

In the Heijo palace site, many mokkans have been found used for luggage tags of gifts, commodities, goods for tax, and so on. Decoding these mokkans is important since we can find the flow of materials, the relations among regions and the condition of economy at the period.

In spite of the archaeological value of mokkans, only a little amount of them have been read and most of them are remaining to be analyzed. Since most of mokkans coming from the underground have been stained, damaged and degraded, it is difficult even for experts to extract characters from badly blurred or missing ink on mokkans and read them

Recently, information technology (IT) is being employed to read historical documents and we can find several papers reporting historical document analysis [1]-[4]. Some propose image processing methods [1]-[3]. Others apply machine recognition of handwritten characters [4]. We also proposed image processing methods and a character recognition method to help experts to read mokkans [5]. However, the effect of these methods is limited when ink for a whole character pattern or more than one character pattern is missing completely.

In this paper, we consider context processing to conjecture missing characters on such mokkans. Since the size of each character pattern on a mokkan is uneven, however, we can hardly assume the number of missing characters by the size of missing ink area. Moreover, the recognition result of the remaining ink area might contain reading errors, which is not revealed until the decoding of the mokkan is completed. We extend the Aho-Corasick method to read incomplete character strings of place names that may contain missing characters and recognition errors.

Section 2 explains mokkans in some detail and problems in decoding them. Section 3 describes the design of the context processing. Section 4 presents evaluation and Section 5 draws the conclusion.

2. Information technologies to support archaeologists to read mokkans

To read scripts on a mokkan, archaeologists extract ink from the mokkan or its picture first. However, very often, ink has been blurred, damaged or missing because:
- Color of ink has been faded out or decolored.

- Color boundaries between ink area and the background (skin of wood with grain) have been vague since the surfaces of wooden tablets have become dark and stained.
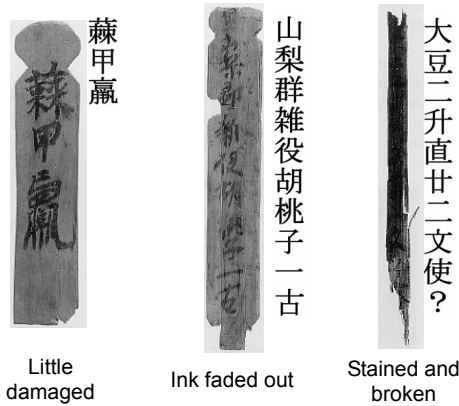- Some parts of a mokkan containing ink have been broken and lost.

蘇甲贏

山梨群雑役胡桃子一古

大豆二升直廿二文使？

Little damaged　　Ink faded out　　Stained and broken

Figure 1. Mokkans excavated from the Heijo palace site.
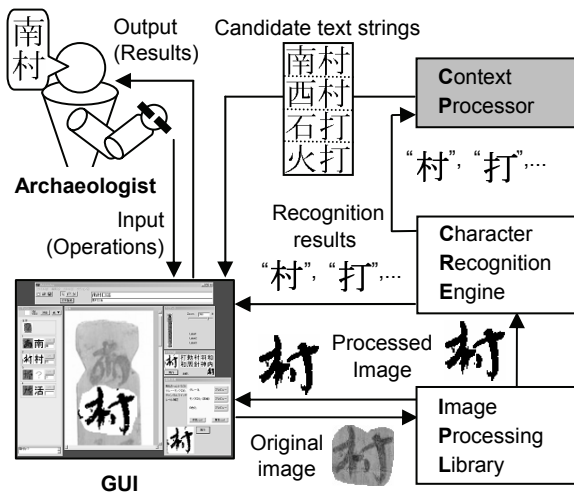

Figure 2. Architecture of the support system.

For these reasons, archaeologists have to make conjectures or hypotheses on the missing ink.

We have proposed a support system for archaeologists to read scripts on mokkans [5]. The system consists of an image processing library, a character recognition engine and a graphical user interface (GUI). We recently developed a context processor as shown in Figure 2.

The purpose of the image processing methods is to extract ink on the mokkans. Discriminant analysis to various color channels and color elements of mokkan images discriminates ink from dark and stained surface of mokkans. Also, our character recognition method outputs candidates even for degraded or partially missing character patterns and stimulates imagination of archaeologists. The GUI provides the above methods to

them. Through the GUI, they can operate most of the functions of the support system by using pen input devices.

3. Needs and problems of context analysis

3.1. Motivation and purpose of context analysis

Scripts of place names on mokkans used as luggage tags show sending places or receiving place. Generally, a place name in Japan in the Nara period has a structure consisting of a state name, a county name and a city/town name. Some place names have both city names and town names. In "*Wamyo-ruijyu-syo* (和名類聚抄)" that is an old book written about the Nara period and the beginning of the Heian period (from A.D. 794 to 1192), we can find more than 4,000 place names at that time.

Decoding scripts of place names with missing ink is difficult, but context analysis may be able to help it. Context analysis is commonly used to detect and correct recognition errors in the area of character and speech pattern recognition. It has a significant effect for address recognition [7] so that it is expected to complement missing characters in place names caused by missing ink on mokkans.

3.2. Problems in reading place names

First, the size of each character pattern on a mokkan is uneven. Therefore, it is difficult to guess the number of characters form the size of a blank space. Second, reading result of each mokkan might contain errors, which is not revealed until the mokkan is completely decoded. Third, scripts of place names sometimes do not follow the order of state/county/city/town names. Some of them might be omitted.
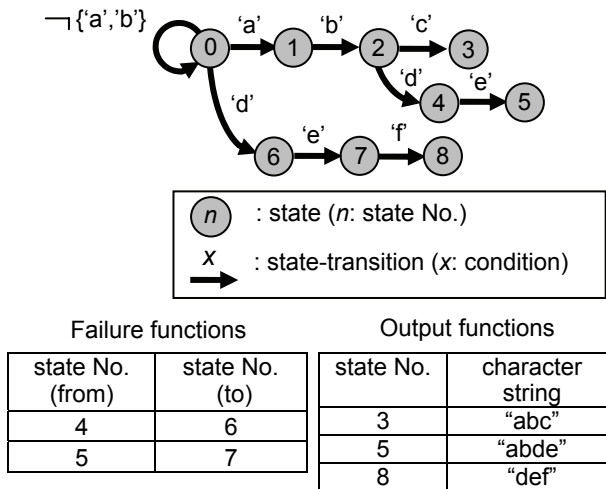
4. Design of context processing

4.1. Aho-Corasick method

To design the context processing, we employ the basic idea of the Aho-Corasick (AC) method that is an efficient method of information retrieval to search multiple keywords in text [6]. First, an automaton that we call AC-automaton is generated from the keywords as shown in Figure 3. Common beginning characters among keywords are shared. By performing state-transitions in the AC-automaton to trace text, all the keywords are detected. It is notable that only a single pass of scanning text is necessary no matter how many keywords are given.
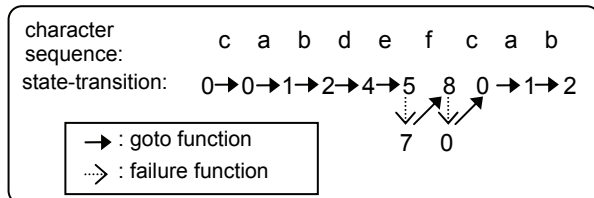
The AC-automaton shown in Figure 3 is generated from keywords "abc", "abde" and "def". The AC method defines states as well as goto functions, output functions and failure functions. A state shows to what point a keyword has been scanned. A goto function specifies state-transition and the condition to execute the transition. An output function outputs a keyword and it is defined at the state that the keyword has been scanned.

We call these states as end-states or *e*-states and others as middle-states or *m*-states in short. Especially, we call the starting state as the initial state (state no. 0) or *i*-state.

A failure function defines an exceptional state-transition. When a failure function coming out of a state is not shown explicitly, it is implicitly defined as the transition to the *i*-state in the AC-automaton. Failure functions are executed when no conditions of goto functions are satisfied or when the state-transition sequence has arrived at one of the *e*-states. Figure 4 shows the state-transition sequence in the AC-automaton when the text "cabdefcab" is given.

Failure functions		Output functions	
state No. (from)	state No. (to)	state No.	character string
4	6	3	"abc"
5	7	5	"abde"
		8	"def"

Figure 3. AC-automaton to detect the keywords "abc", "abde" and "def".

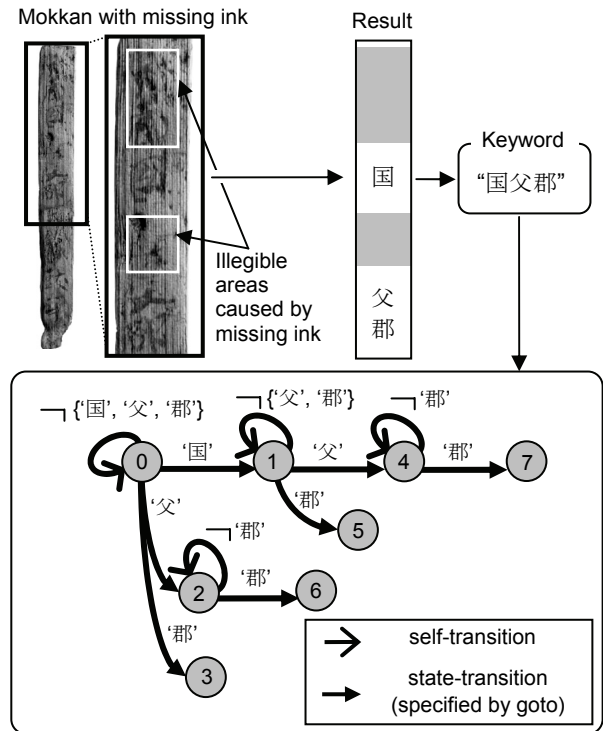**Figure 4**. State-transitions for a character sequence.

4.2. Extended AC method

The problem to apply the AC method is that text on a mokkan is incompletely decoded with some characters lost or replaced by others and the order of them changed. With simple extension, however, the AC method works to presents correct place names from given imperfect keywords. The extension is to allow self-transitions when the condition going to the next state is not satisfied and to prepare paths with arbitrary characters omitted from keywords in the AC-automaton. We call the method as Extended AC (EAC) method.

An automaton of the EAC method (EAC-automaton) is generated from the given keywords similarly as for the AC-automaton. A keyword may be a partially decoded sequence of characters or multiple alternative analyses.

4.2.1. Implication of self-transition

The purpose to define the self-transition functions is to skip characters written in illegible areas on a mokkan. For instance as shown in Figure 5, when a partially decoded keyword is "国父郡", the place name "武蔵国秩父郡", where "武蔵国" is a state name and "秩父郡" is a county name, is processed by the sequence of state-transitions shown in Figure 6. Note that the missing characters '武', '蔵' and '秩', are scanned by self-transitions.

Figure 5. EAC-automaton for a partially decoded keyword.
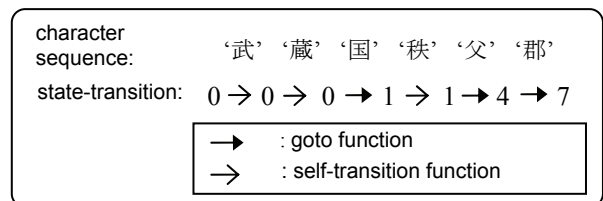
Figure 6. State-transitions when the keyword lacks characters due to lost ink.

4.2.2. Implication of skipping paths

We prepare paths with arbitrary characters omitted from keywords in the AC-automaton. We call these paths as skipping paths. The reason of adding skipping paths is that characters in the keywords can be the result of misreading.

Figure 7 shows the EAC-automaton for the keyword "国母郡". Path from state 1 to state 5, one from 0 to 2, another from 0 to 3 are skipping paths. When the same
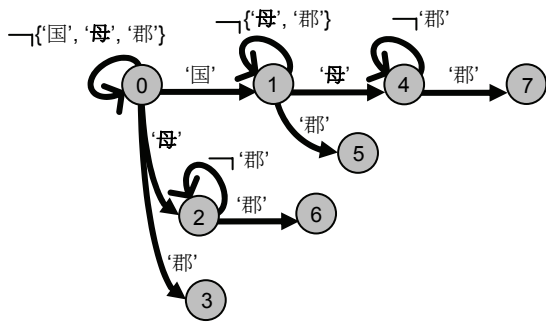
Figure 7. EAC-automaton for the keyword containing a replaced character.

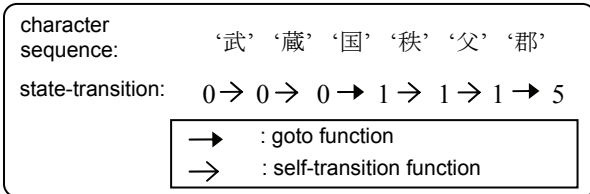| character sequence:	'武' '蔵' '国' '秩' '父' '郡'
state-transition:	$0 \to 0 \to 0 \to 1 \to 1 \to 1 \to 5$
	$\to$: goto function
	$\to$: self-transition function

Figure 8. State-transitions when the keyword contains a replaced character.

place name "武蔵国秩父郡" as the above is processed by the EAC-automaton, the sequence of state-transitions is shown in Figure 8.

In the above example, the character '母' in the keyword is the result of misreading. The characters '秩' and '父' are processed by the self-transitions at the state 1 and the character '郡' is scanned by the state-transition from the state 1 to 5. The skipping path from 0 to 2 works when the character '国' is the result of misreading and that from 0 to 3 works when both the '国' and '母' are the results of misreading. The self transitions of the state 2 and 4 work as skipping functions when the last character '郡' is the result of misreading.

When the number of the characters composing a keyword is l, the i-state has basically l next states to make state-transition and the state, to which the k-th character of the keyword is the condition to come, has basically l-k states as the next states.

4.2.3. Scoring

Since the EAC method is the context analysis method, it should output the evaluation score showing the similarity between keywords and place names in the vocabulary. The EAC method assigns scores to the state-transitions specified by goto functions. First, the score 1 = 2^0 is assigned to each state-transition coming out from the i-state and specified by goto functions. Second, given a state to which the state-transition to come is assigned 2^n as its score, the score 2^{n+1} is assigned to all the state-transitions coming out from the state and arriving to the next states. On the other hand, we don't assign any scores to self-transitions. The similarity between the keyword and a place name in the vocabulary is the sum of the scores on the state-transition sequence. Figure 9 shows the scores of an
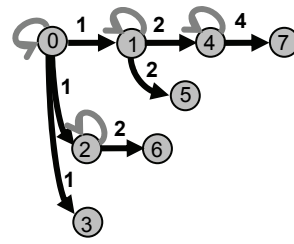

Figure 9. Scores assigned to state-transitions.

EAC-automaton for a keyword consisting of 3 characters.

The scoring policy the EAC-method is to assign higher scores as state-transitions are farther away from the i-state. The purpose is to give higher scores to the place names that contain more characters of the keyword with keeping their order.

In the case shown in Figure 6, the score is 1+2+4=7. Also, it becomes 1+2=3 in the case shown in Figure 8.

4.2.4. Output function

In the AC method, a goto function specifies state-transition and the condition to execute the transition. In the EAC-method, however, the similarity between a keyword and a place name must be evaluated even if the state-transition sequence for the place name does not arrive to any e-states. Therefore, we define output functions to all the states in EAC-automatons.

Figure 10 shows the EAC-automaton for the keyword " 国 父 県 " where the last character is misrecognized from '郡'. On the EAC-automaton, a place name "武蔵国秩父郡" does not reach to any e-state (Figure 11). However, the EAC-automaton outputs 3 (=1+2) as the result of the evaluation by defining the output function to the state 4.

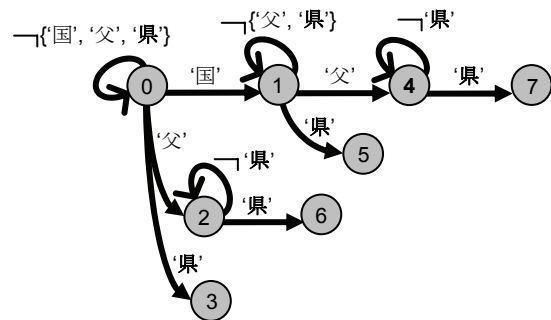**Figure 10**. EAC-automaton for the keyword where the last character is misrecognized.

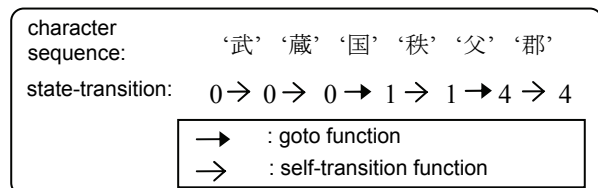| character sequence:	'武' '蔵' '国' '秩' '父' '郡'
state-transition:	$0 \to 0 \to 0 \to 1 \to 1 \to 4 \to 4$
	$\to$: goto function
	$\to$: self-transition function

Figure 11. State-transitions when the keyword where the last character is misrecognized.

4.2.5. Failure function

Although there is no difference in the definition of failure functions between the AC method and the EAC method, those in the latter work to accept keywords of wrong state/county/city/town order.

In the AC method, a failure function defines an exceptional state-transition. When it is not explicit, it is implicitly defined as the transition to the i-state. In the EAC-method, however, self-transition may also occur in m-states. We break the tie by suspending the transition to the i-state until the next character makes the transition to the next states from the i-state, otherwise making the self-transition at the m-states.

Figure 12 shows the EAC-automaton for the keyword "父蔵国". It may be the partial decoding result of the wrongly ordered place name "秩父郡武蔵国" instead of "武蔵国秩父郡" since "父" is a part of the county name "秩父郡", "蔵国" is a part of the state name "武蔵国". Although all the characters of the keyword are contained in the place name, the similarity between the keyword and the place name should be reduced because the order of the state and county names is opposite. We show the state-transition sequence in Figure 13.
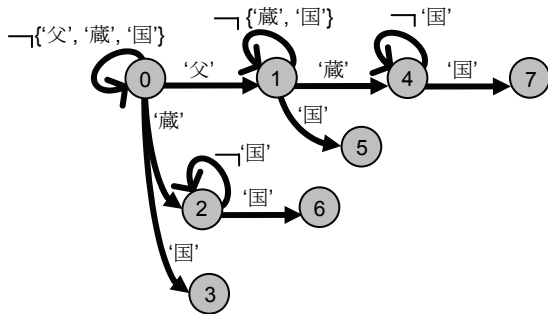

Figure 12. EAC-automaton for the keyword with different order of characters.
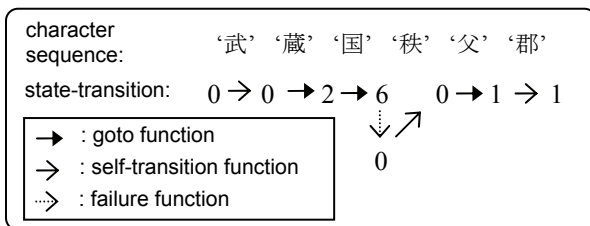

Figure 13. State-transitions when order of characters is changed in keyword.

When the state-transition sequence arrives at the e-state 6, the failure function defined to the state executes the transition to the i-state. As the result, each character of the keyword satisfies a condition of one of the goto functions. The total score becomes 4 (=1+2+1) that is lower than the case shown in Figure 6 but higher than the case of Figure 8.

Figure 14 shows another example when the keyword is "父蔵口" caused by the misreading of '国' to '口'.
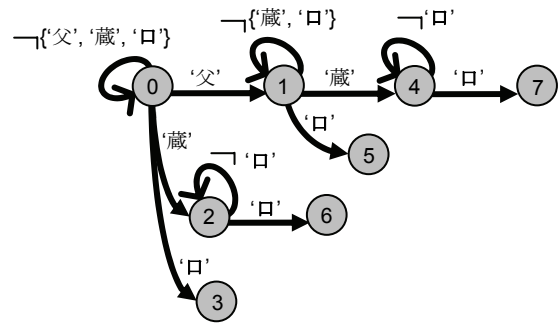

Figure 14. EAC-automaton for the keyword with misreading and different order of characters.
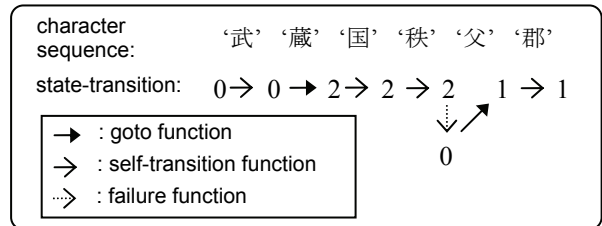

Figure 15. State-transitions when the order of characters is changed in the keyword.

Figure 15 shows the state-transition sequence. Note that both the self-transition function and the failure function are executable at the state 2. The next character '父' breaks the tie and makes the state transition to go back to the i-state and then proceed to the state 1.

4.2.6. Tying of state-transitions

When multiple state-transitions coming out from the same state have the same character as the conditions for the goto functions, they are tied to avoid the redundancy. Such redundant state-transitions are due to the same character contained twice or more times in a keyword. (Figure 16).
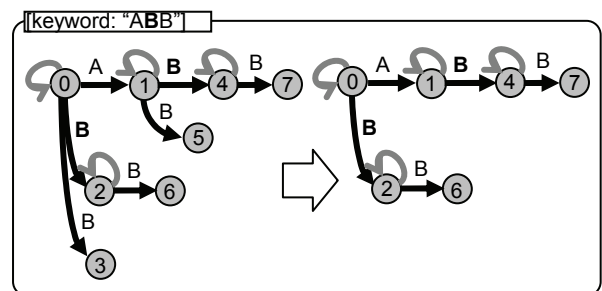

Figure 16. Tying of state-transitions.

5. Experimental result

We prepared the database consisting of 4,041 place names described in "*Wamyo-ruijyu-syo*" to evaluate the effect of the EAC method. We employed all the place names of the database as the vocabulary.

To prepare imperfect keywords for the experiment, we employed 3,993 place names consisting of 9-11

characters registered to the database and randomly removed 0-6 characters. Removal of characters simulates missing characters caused by illegible areas on mokkans. We call this set of imperfect keywords as "test keyword set *A*". Moreover, we replaced one character and two characters of every test keyword in the test keyword set *A* to generate mis-decoded keywords. We call the set of the test keywords with one character replaced as "test keyword set B_1" and the other set of test keywords with two characters replaced as "test keyword set B_2".

The EAC method generates the EAC-automaton for each test keyword, matches all the place names in the vocabulary to the EAC automaton and ranks the similarity between each place name and the keyword of the EAC automaton. When the original place name of the keyword comes in the top ten by the similarity, we consider that the estimation of the EAC method is correct.

To compare with the EAC method, we have implemented a simple method (product method) that counts how many characters in the keyword is contained in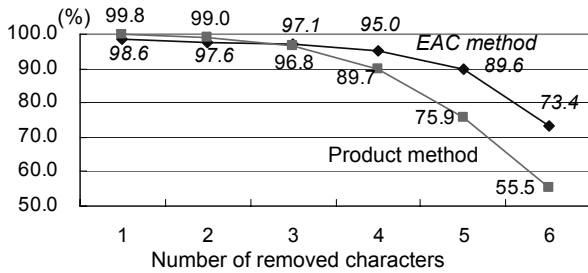 each place name. The product method is also adaptable to keywords containing missing and replaced characters. The rates of the correct estimated place names are shown in Figure 17, 18 and 19.

Although the product method does not evaluate the order of characters in keywords, the EAC method evaluates it. This difference seems to give an advantage to the EAC method when the number of lost characters becomes large. When one or two characters are replaced, however, the advantage of the EAC method is smaller as the number of lost characters increases. It is hard to guess original 9 to 11 place names even for people from 3 to 5 remaining characters with 1 or 2 characters replaced to others.

6. Conclusion

This paper describes the EAC method to presents candidates for damaged scripts on mokkans. By employing the AC method as the basic idea and extending its definition with allowing self-transition, the Extended AC method presents candidates even for incomplete character sequences in which some characters are lost or replaced by others and the order of them is changed. The remaining work is to evaluate the method in real situations used by archaeologists. Also, the scoring method

Acknowledgement
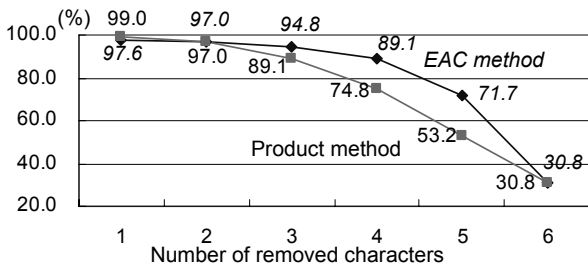
Figure 17. Estimation rates for the test keyword set *A*.

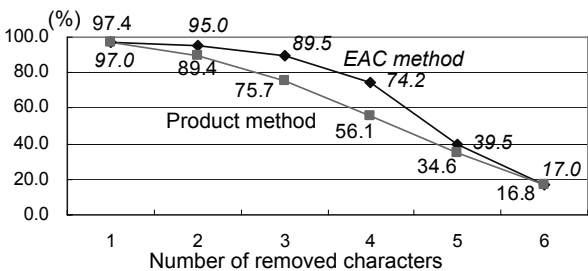**Figure 18**. Estimation rates for the test keyword set B_1.

Figure 19. Estimation rates for the test keyword set B_2.

References

[1] B. Gatos, I. Pratikakis, and S.J. Perantonis, "An Adaptive Binarization Technique for Low Quality Historical Documents", Proc. 6th DAS, Florence, Italy, pp.102-113, Sep. 2004.

[2] C. Yan and G. Leedham, "Decompose-Threshold Approach to Handwriting Extraction in Degraded Historical Document Images", Proc. 9th IWFHR, Tokyo, Japan, pp.239-244, Oct. 2004.

[3] Z. Shi and V. Govindaraju, "Historical Document Image Enhancement Using Background Light Intensity Normalization", Proc. 17th ICPR, Cambridge, UK, Aug. 2004, 2aP. Mo-ii.

[4] M.S. Kim, K.T. Cho, H.K. Kwag and J.H. Kim, "Segmentation of Handwritten Characters for Digitalizing Korean Historical Documents", Proc. 6th DAS, Florence, Italy, pp.114-124, Sep. 2004.

[5] A. Kitadai, K. Saito, D. Hachiya, M. Nakagawa, H. Baba and A. Watanabe, "Support System for Archeologists to Read Scripts on Mokkans", Proc. 8th ICDAR, vol.2, pp.1030-1034, Aug. 2005.

[6] A.V. Aho and M.J. Corasick, "Efficient string matching: An aid to bibliographic search", Commu. of ACM, vol. 18, no. 6, pp.333-340, 1975.

[7] C.-L. Liu, M. Koga and H. Fujisawa, "Lexicon-Driven Segmentation and Recognition of Handwritten Character Strings for Japanese Address Reading" IEEE Trans. PAMI, vol. 24, no. 11, pp.1425-1437, Nov. 2002.