Experimental Comparison of ASCII Art Extraction Methods: a Run-Length Encoding based Method and a Byte Pattern based Method

Tetsuya Suzuki
Shibaura Institute of Technology, Japan
tetsuya@shibaura-it.ac.jp

ABSTRACT: *Pictures consisting of computer characters are called ASCII art. They are widely used in computer texts because they enrich expression in texts. To deal with texts including ASCII art objects, ASCII art extraction methods, which detect ASCII art objects in a given text, are now demanded. For example, such methods are needed to find ASCII art objects in texts for sentiment analysis based on ASCII art objects. Such methods are also needed to remove ASCII art objects from texts in preprocess for natural language processing because non-verbal texts of ASCII art makes texts noisy. Our research group and another research group independently proposed two different ASCII art extraction methods, which are a run-length encoding based method and a byte pattern based method respectively. Both of the methods use ASCII art recognizers constructed by machine learning algorithms, but they use different attributes of texts. In this paper, we compare the two methods by ASCII art extraction experiments where training texts and testing texts are in English and Japanese. According to our experimental results, the two methods are competitive if training texts and testing texts are in a same set of languages, but the run-length encoding based method works better than the byte pattern based method if training texts and testing texts are in different sets of languages.*

Keywords: ASCII Art, Information Extraction, Natural Language Processing, Pattern Recognition

Received: 3 January 2017, Revised 22 February 2017, Accepted 27 February 2017

1. Introduction

Pictures consisting of computer characters called *ASCII art* are widely used in computer texts such as Web pages, microbloggings, emails, and so on. To be exact, ASCII art means pictures consisting of ASCII code characters. It is, however, also used for pictures consisting of other character encoding methods such as Unicode.

This paper is based on the paper "Comparison of Two ASCII Art Extraction Methods: A Run-Length Encoding based Method and a Byte Pattern based Method" presented at the 6th IASTED International Conference on Computational Intelligence in 2015 [11]. Copyright © 2010 Permission to copy without fee all or part of the material printed in JMPT is granted provided that the copies are not made or distributed for commercial advantage.

ASCII art objects, which are instances of ASCII art, are roughly classified into two major categories in [14]: the structure-based ASCII art and the tone-based ASCII art. *The structure-based ASCII art* means pictures where outlines of objects are drawn by characters. Fig.1 shows a structure-based ASCII art object of a mascot called 'Monnar' in the most popular bulletin board service (BBS) '2channel' in Japan. *Emoticons*, which are facial expression represented by characters such as a smiley ':-)', are also categorized into the structure-based ASCII art. On the other hand, *the tone-based ASCII art* means gray scale pictures consisting of characters. Fig.2 and Fig.3 show an image of a tower and a tone-based ASCII art object of the image respectively.

We categorize ASCII art into two categories from a different point of view: the block ASCII art and the in-line ASCII art. *The block ASCII art* is ASCII art consisting of lines. The structure-based ASCII art object of Fig.1 and the tone- based ASCII art object of Fig.3 are examples of the block ASCII art. On the other hand, *the in-line ASCII art* is ASCII art which can be embedded in a line such as the smiley ':-)'.

To deal with computer texts including ASCII art objects, *ASCII art extraction methods*, which detect ASCII art objects in a given text, are now demanded. For example, such methods are needed to find ASCII art objects in texts for sentiment analysis based on ASCII art objects [4, 15]. Such methods are also needed to remove ASCII art objects from texts in preprocess for natural language processing because non-verbal texts of ASCII art makes texts noisy.

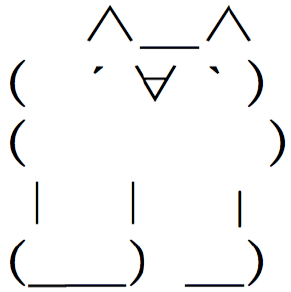

Figure 1. A structure-based ASCII art object called 'Monnar'

Our research group and another research group independently proposed two different ASCII art extraction methods for the block ASCII art. We proposed a run-length encoding (RLE) based method [9, 10, 11, 12]. The compression ratio of a text by RLE can be used to indicate how the text looks like the block ASCII art because same characters tend to successively occur in lines of the block ASCII art. On the other hand, Nakazawa et al. proposed a byte pattern (BP) based method [5]. The method uses the frequency distribution of byte values in each line of a text to tell if the line is a part of an ASCII art object or not.

In this paper, we experimentally compare the two ASCII art extraction methods because the two methods have not been compared yet by ASCII art extraction experiments using same text data sets.

The rest of the paper consists as follows. In section 2, we explain related work. In section 3 and 4, we explain the RLE based method and the BP based method respectively. In section 5, we explain our extraction experiments to compare the two methods. In section 6, we evaluate the two methods by the experimental results. We finally state our conclusion in section 7.

2. Related Work

We explain related work about ASCII art recognition and ASCII art extraction.

2.1 A Support Vector Machine-based ASCII Art Recognition Method

Tanioka et al. proposed a support vector machine (SVM)-based ASCII art recognition method, which tells if a given text is an ASCII art object or not, as a natural language processing technique. [3] Training data for SVM is a set of 262 dimension vectors. Each vector consists of two parts. The first 256 elements of the vector represent a *byte pattern*, whose i-th ($0 \leq i \leq 255$) element is the occurrence number of the byte value i in the byte stream of a UTF-8 encoded text. The authors categorized Japanese parts of speech into 6 groups. The rest 6 elements of the vector represent the occurrence numbers of the groups in the text. Because it is specific to Japanese language, this method will not work well for other languages.

2.2 In-line Emoticon Extraction

To extract in-line emoticons from a text, predefined sets of emoticons are used. Systems for automatic emoticon-based sentiment analysis of texts posted to a microblogging service Twitter use emoticon lexicons to extract in-line emoticons [15, 4]. The CAO system is an automatic affect analysis of in-line emoticons [7]. The system extracts emoticons from texts with reference to a database of emoticons.

Figure 2. An image of Tokyo Tower

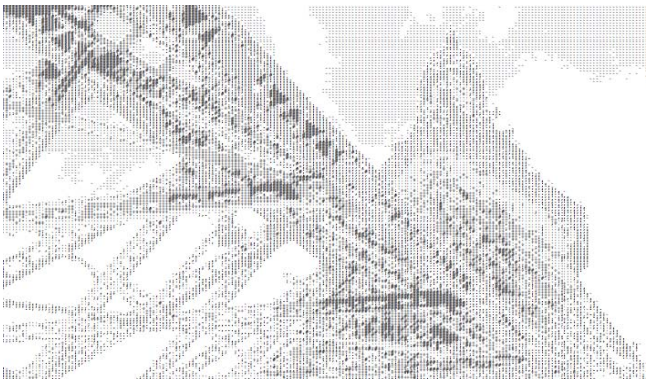

Figure 3. A tone-based ASCII art object of Tokyo Tower

2.3 ASCII Art Extraction Software

A freely distributed software "AA scan" [2] , whose purpose is support of ASCII art collection, recognizes articles with ASCII art in the BBS '2channel'. Though the author does not disclose the detail of the recognition method, he describes that the recognition method is a heuristic method specific to Japanese language in its document. For example, it uses occurrence rates of characters in texts which include not only the English alphabet but also Japanese characters such as Hiragana, Katakana, and Kanji.

3. A Run-Length Encoding based ASCII Art Extraction Method

In this section, we explain our RLE based ASCII art extraction method [9, 10, 11, 12]. We first explain two parts of the extraction method: a procedure called window scanning and a procedure called window reduction. We then explain the ASCII art extraction method. We finally explain an ASCII art recognizer used in the ASCII art extraction method. We assume that texts are encoded in UTF-8.

3.1 Window Scanning

We define a procedure called *window scanning*. It takes a window width k, a procedure P, and a text T as its inputs. The procedure P takes a text as its input. We use $P(x)$ to denote the output of P for a text x. The text T is represented by a sequence of lines. The window scanning procedure watches successive k lines on T and move the area from the beginning to the end of

T by one line. We call the successive *k* lines as *a window*. During the scanning, it applies the procedure *P* to each window. The output of this procedure is a set of pairs of a window *w* and *P*(*w*).

3.2 Window Reduction

We define a procedure called *window reduction* as follows. It takes a window and an ASCII art recognizer as its inputs. It then removes the following lines from the window, and outputs the resulting window.

- Successive lines from the starting line which are recognized as a non-ASCII art object.
- Successive lines from the end line which are recognized as a non-ASCII art object.

3.3 An ASCII Art Extraction Method

We define an ASCII art extraction method as follows. It takes a window width *k*, an ASCII art recognizer *M*, and a text *T* as its inputs. It outputs a set of windows in *T* as a set of ASCII art objects. The following is the procedure.

1. It calls the window scanning procedure with the window width *k*, the ASCII art recognizer *M*, and *T*. It means that the procedure applies *M* to windows in the scanning.
2. It then collects windows recognized as ASCII art objects from the results.
3. For each chunk of overlapped windows in the collected windows, it merges the overlapped windows into a window as a roughly extracted ASCII art object.
4. For each roughly extracted ASCII art object, it applies the window reduction procedure with *M*.
5. It finally outputs the results of the window reduction procedure as a set of ASCII art objects. The reason why it roughly extracts ASCII art objects at first is to avoid splitting an ASCII art object into parts.

Fig.4 shows an example of the input text *T* for the RLE based extraction method, where an ASCII art object exists between English texts. Fig.5 shows a roughly extracted ASCII art object at the step 3 where there are redundant lines before and after the ASCII art. Fig.6 shows the resulting ASCII art object extracted at the step 4.

3.4 An ASCII Art Recognizer

We use an ASCII art recognizer in the ASCII art extraction method, which is constructed by a machine learning algorithm. It takes a set of text attributes as its input and outputs whether true or false. The true value and the false value represent that the text is an ASCII art object and that it is not respectively.

We construct training data for the machine learning as follows.

1. We prepare a set of ASCII art objects and a set of non-ASCII art texts.
2. We extract text attributes from them using the window scanning with window width *i* (= 1, 2, 3, . . ., *k*) where *k* is a window width used in the ASCII art extraction method.

The extracted text attributes are *R*, *L* and *S*. The attribute *R* is an attribute based on data compression ratio by RLE. Given a text *T* consisting of *n* lines, the attribute is defined as follow.

$$\mathrm{R} \equiv \frac{\sum_{i=1}^{n} |RLE(l_i)|}{|T|} \tag{1}$$

where $|x|$ denotes the length of a string *x*, $RLE(x)$ denotes a string encoded from the string *x* by RLE, and l_i is the *i*-th line of *T*. The attributes *L* and *S* are the number of lines and the length of the text data respectively.

Before we calculate the attributes of a given text, we normalize the text [10]. There exist two kinds of white spaces in Unicode, whose code points are U+0020 and U+3000. We replace each U+3000 white space with two U+0020 white spaces in texts because the font width of U+3000 is the double of that of U+0020. According to our ASCII art extraction experiments in [10], this text normalization improves the *F*-measure of the precision and the recall.

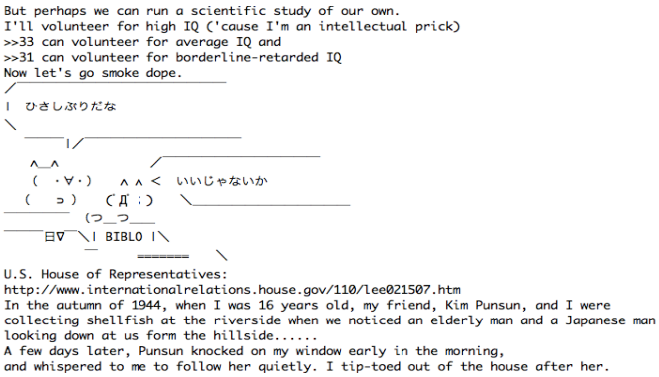

Figure 4. An input text for the RLE based ASCII art extraction method

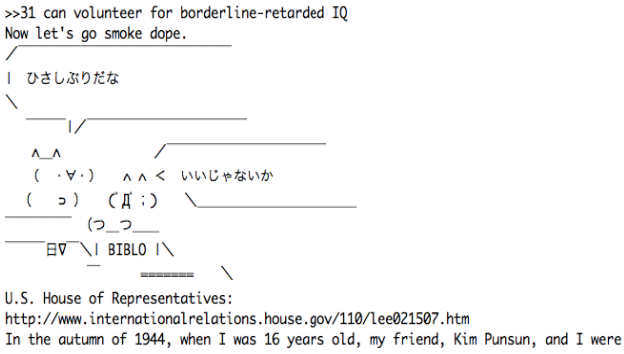

Figure 5. A roughly extracted ASCII art object

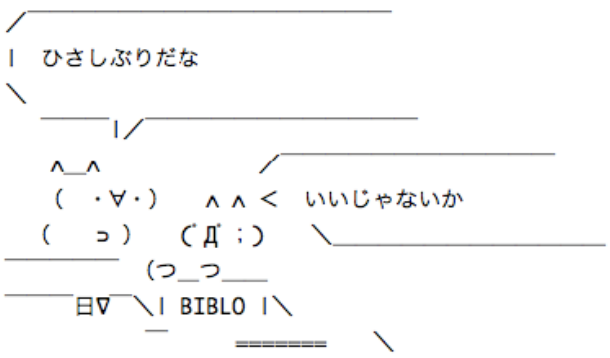

Figure 6. A result of the RLE based ASCII art extraction

4. A Byte Pattern based ASCII art Extraction Method

Nakazawa et al. proposed a BP based method [5]. It scans a UTF-8 encoded text by one line and detects areas of ASCII art objects in the text. In advance of ASCII art extraction, an SVM model is constructed by a learning algorithm as an ASCII art recognizer. Each training data consists of a class and an extended byte pattern. A class is either a class of ASCII art or a class of non-ASCII art. An extended byte pattern of a line is a concatenation of byte patterns of the line, the previous *N*-lines, and the next *N*-lines. If, for example, $N = 1$, it is a concatenation of byte patterns of three lines, which is a 768 dimension vector.

To avoid splitting an ASCII art object into parts in ASCII art extraction, the BP based method uses the smoothed ASCII art possibility. If the smoothed ASCII art possibility of a line is greater than or equal to 50%, the line is recognized as a part of an ASCII art object.

The smoothed ASCII art possibility of a line is calculated from the ASCII art possibilities of $2M+1$ lines, which are the line, the previous M-lines, and the next M-lines. The expression (2) shows the detail where p_i and w_i are the ASCII art possibility and the weight of i-th line respectively. The ASCII art possibilities of a line is calculated from an extended byte pattern of the line using the constructed SVM model.

$$\frac{w_i p_i + \sum_{k=1}^{M} w_{i-k} p_{i-k} + \sum_{k=1}^{M} w_{i+k} p_{i+k}}{w_i + \sum_{k=1}^{M} w_{i-k} + \sum_{k=1}^{M} w_{i+k}} \quad (2)$$

5. Experiments

We compare the two ASCII art extraction methods by extraction tests as follows.

5.1 Training and Testing Data Sets

We used two sets of texts E and J encoded in UTF-8 for the experiment. The set of texts E consists of English texts with 289 ASCII art objects and 290 non-ASCII art objects, whose lines range from 1 to 118. The set of texts J consists of Japanese texts with 259 ASCII art objects and 299 non-ASCII art objects, whose lines range from 1 to 39.

We constructed training data sets A and B as follows. We divided the set of text data E and J into two groups A and B where $A = E_A \cup J_A$, $B = E_B \cup J_B$, $E = E_A \cup E_B$, $E_A \cap E_B = \phi$, $J = J_A \cup J_B$ and $J_A \cap J_B = \phi$. Table 1 shows the number of ASCII art objects and the number of non-ASCII art objects in them.

We then made 800 text data as testing data from each of A and B. Each of the 800 text data consists of three parts X, Y , and Z where X and Z are randomly selected non-ASCII art objects from A (B) and Y is randomly selected ASCII art objects from A (B). Each of X, Y , and Z is either an English text or a Japanese text. There exist eight combinations of languages for X-Y -Z as follows: (1) English-English-English, (2) English-English-Japanese, (3) English-Japanese-English, (4) English-Japanese-Japanese, (5) Japanese-English-English, (6) Japanese-English-Japanese, (7) Japanese-Japanese-English and (8) Japanese-Japanese-Japanese.

	E		***J***	
	E_A	E_B	J_A	J_B
The number of ASCII art objects	145	144	130	129
The number of non-ASCII art objects	145	145	150	149

Table 1. The number of objects in the sets of texts E, E_A, E_B, J, J_A, and J_B

	Training data set	**Testing data set**
E-J	E_X	Japanese-Japanese-Japanese generated from J_Y
J-E	J_X	English-English-English generated from E_Y
E-E	E_X	English-English-English generated from E_Y
J-J	J_X	Japanese-Japanese-Japanese generated from J_Y
EJ-EJ	$E_X \cup J_X$	Union of English-English-Japanese, English-Japanese-English, English-Japanese-Japanese, Japanese-English-English, Japanese-English-Japanese, and Japanese-Japanese-English generated from E_Y and J_Y

Table 2. Combinations of the training data and testing data. X and Y are a group name and its counter group name respectively. It means that $X, Y \in \{\mathrm{A}, \mathrm{B}\}$ and $X \neq Y$

Fig.4 shows an example of the combination (3) English-Japanese-English. For each of the eight combinations, we made 100 text data. As a result, we got 800 text data in total from each of *A* and *B*.

5.2 Conditions

We conducted ASCII art extraction experiments using the two methods under the following conditions.

We measured the average of precision, the average of recall and the average of *F*-measure in ASCII art extraction by 2-fold cross validation in the five cases E-J, J-E, E-E, J-J, and EJ-EJ. Each combination of E and J represents languages used in the training data and the testing data. In the case E-J, for example, the training data in English and the testing data in Japanese were used. Table 2 shows the detail of the combinations.

In extraction by the RLE based ASCII art extraction method, we changed the window width from 1 to 10 by 1.

In extraction by the BP based ASCII art extraction method, we set *N* with 1 for extended byte patterns as in their extraction experiment [5]. We also set *M* with 1 in the expression (2) though the value of *M* is not clearly stated in their experiment [5]. As a result, we used the following expression (3) to calculate the smoothed ASCII art possibility.

$$\frac{w_i p_i + w_{i-1} p_{i-1} + w_{i+1} p_{i+1}}{w_i + w_{i-1} + w_{i+1}} \quad (3)$$

We changed the weight w_i in this expression (3) from 0.0 to 1.0 by 0.1 under the condition that $w_i + w_{i-1} + w_{i+1} = 1$ and $w_{i-1} = w_{i+1}$.

We implemented the two extraction methods in Java.[1] We used decision trees as ASCII art recognizer for the RLE based method. The decision trees were constructed by the C4.5 machine learning algorithm [8] implemented in the data mining tool Weka [6, 13] with the default parameters.

On the other hand, we used a library for support vector machines LIBSVM [1] to implement the BP based method. We regularized byte patterns by feature scaling in this experiment. For each combination of the training data and the weight w_i, we conducted grid search to tune two parameters *C* and γ of SVM. In the grid search, *C* and γ ranged over $\{2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}, 2^{13}, 2^{15}\}$ and $\{2^{3}, 2^{1}, 2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-11}, 2^{-13}, 2^{-15}\}$ respectively.

5.3 Results

5.3.1 Extraction in the Case E-J

Table 3(a) and Table 4(a) show the results by the RLE based method and the BP based method respectively under the case E-J. In the case of the RLE based method, the highest average of *F*-measure is 0.897 when the widow width is 4. In the case of the BP based method, the highest average of *F*-measure is 0.630 when the weight is 0.0. The highest average of *F*-measure of the BP based method is 70.2% of that of the RLE based method.

5.3.2 Extraction in the Case J-E

Table 3(b) and Table 4(b) show the results by the RLE based method and the BP based method respectively under the case J-E. In the case of the RLE based method, the highest average of *F*-measure is 0.990 when the widow width is 9. In the case of the BP based method, the highest average of *F*-measure is 0.831 when the weight is 0.0. The highest average of *F*-measure of the BP based method is 83.9% of that of the RLE based method.

5.3.3 Extraction in the Case E-E

Table 3(c) and Table 4(c) show the results by the RLE based method and the BP based method respectively under the case E-E. In the case of the RLE based method, the highest average of *F*-measure is 0.996 when the widow width is 4. In the case of the BP based method, the highest average of *F*-measure is 0.965 when the weight is 0.4 and 0.5. The highest average of *F*-measure of the BP based method is 96.9% of that of the RLE based method.

5.3.4 Extraction in the Case J-J

Table 3(d) and Table 4(d) show the results by the RLE based method and the BP based method respectively under the case J-J.

[1] *The implementation of our run-length encoding based ASCII art extraction method is available on our GitHub repository* https://github.com/tslab-sit/asciiart-extractor .

In the case of the RLE based method, the highest average of *F*-measure is 0.953 when the widow width is 6. In the case of the BP based method, the highest average of *F*-measure is 0.944 when the weight is 0.3. The highest average of *F*-measure of the BP based method is 99.1% of that of the RLE based method.

5.3.5 Extraction in the Case EJ-EJ

Table 3(e) and Table 4(e) show the results by the RLE based method and the BP based method respectively under the case EJ-EJ. In the case of the RLE based method, the highest average of *F*-measure is 0.972 when the widow width is 5, 6 and 7. In the case of the BP based method, the highest average of *F*-measure is 0.955 when the weight is 0.2. The highest average of *F*-measure of the BP based method is 98.3% of that of the RLE based method.

6. Evaluation

Table 3(a), Table 4(a), Table 3(b), and Table 4(b) show the results where the training texts and the testing texts are in different sets of languages. In these cases, the highest average of *F*-measure by the BP based method is at most 83.9% of that by the RLE based method.

On the other hand, Table 3(c), Table 4(c), Table 3(d), Table 4(d), Table 3(e), and Table 4(e) show the results where training texts and testing texts are in a same set of languages. In these cases, the highest average of *F*-measure by the BP based method is at least 96.9% of that by the RLE based method.

According to these results, the two methods are competitive if training texts and testing texts are in a same set of languages but the RLE based method works better than the BP based method if training texts and testing texts are in different sets of languages.

The reasons are as follows. Table 5 shows the occurrence percentages of byte values in texts used in our experiments. In English texts of both ASCII art and non-ASCII art, more than 87% of byte values are in the range between 00h and $7F_h$. In Japanese texts of both ASCII art and non-ASCII art, more than 81% of byte values are in the range between 80h and FF_h. It is due to the UTF-8 encoding method [16]. ASCII art objects and non-ASCII art objects in a language can not be distinguished well by recognizers generated from byte patterns of texts in a different language because of the difference.

On the other hand, Table 6 shows the percentages of compression ratios by RLE in texts used in our experiments. In the texts of ASCII art in both English and Japanese, more than 87% of compression ratio are in the range between 0.0 and 1.6. In the texts of non-ASCII art in both English and Japanese, more than 97% of compression ratio are in the range between 1.6 and 2.0. ASCII art objects and non-ASCII art objects can be roughly distinguished by even one test of compression ratio regardless of their languages because of the difference.

Window Width	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
1	0.896	0.774	0.811
2	0.889	0.918	0.892
3	0.868	0.931	0.886
4	0.883	0.938	0.897
5	0.839	0.942	0.873
6	0.787	0.944	0.840
7	0.795	0.948	0.844
8	0.770	0.941	0.826
9	0.750	0.948	0.809
10	0.742	0.941	0.799

(a) The case E-J

Window Width	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
1	0.987	0.915	0.947
2	0.981	0.950	0.964
3	0.993	0.975	0.983
4	0.986	0.977	0.981
5	0.985	0.979	0.982
6	0.982	0.981	0.981
7	0.987	0.989	0.987
8	0.985	0.992	0.988
9	0.985	0.995	0.990
10	0.984	0.996	0.989

(b) The case J-E

Window Width	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
1	0.990	0.922	0.954
2	0.998	0.976	0.986
3	0.999	0.989	0.994
4	1.000	0.993	0.996
5	0.995	0.995	0.995
6	0.992	0.997	0.994
7	0.992	0.997	0.994
8	0.991	0.997	0.993
9	0.991	0.997	0.993
10	0.988	0.997	0.991

(c) The case E-E

Window Width	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
1	0.923	0.787	0.836
2	0.947	0.917	0.922
3	0.957	0.930	0.932
4	0.958	0.951	0.947
5	0.955	0.958	0.949
6	0.951	0.968	0.953
7	0.940	0.972	0.949
8	0.938	0.969	0.946
9	0.927	0.973	0.939
10	0.919	0.970	0.933

(d) The case J-J

Window Width	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
1	0.958	0.855	0.895
2	0.973	0.946	0.954
3	0.975	0.954	0.959
4	0.973	0.969	0.967
5	0.976	0.975	0.972
6	0.975	0.976	0.972
7	0.971	0.980	0.972
8	0.962	0.976	0.964
9	0.960	0.977	0.964
10	0.953	0.979	0.960

(e) The case EJ-EJ

Table 3. The results by the RLE based ASCII art extraction method. Maximum averages of *F*-measure in each table are underlined

Weight	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
0.0	0.492	0.996	0.630
0.1	0.458	0.996	0.602
0.2	0.458	0.994	0.601
0.3	0.457	0.994	0.601
0.4	0.457	0.992	0.601
0.5	0.458	0.992	0.602
0.6	0.457	0.992	0.601
0.7	0.457	0.991	0.601
0.8	0.457	0.990	0.601
0.9	0.458	0.990	0.601
1.0	0.458	0.990	0.601

(a) The case E-J

Weight	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
0.0	0.948	0.805	0.831
0.1	0.946	0.798	0.826
0.2	0.957	0.778	0.818
0.3	0.956	0.774	0.815
0.4	0.956	0.771	0.814
0.5	0.945	0.763	0.805
0.6	0.939	0.759	0.801
0.7	0.938	0.753	0.797
0.8	0.927	0.744	0.787
0.9	0.919	0.734	0.781
1.0	0.920	0.723	0.773

(b) The case J-E

Weight	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
0.0	0.982	0.919	0.945
0.1	0.973	0.958	0.964
0.2	0.975	0.957	0.964
0.3	0.975	0.957	0.964
0.4	0.975	0.958	0.965
0.5	0.975	0.958	0.965
0.6	0.975	0.958	0.964
0.7	0.975	0.958	0.964
0.8	0.975	0.958	0.964
0.9	0.975	0.958	0.964
1.0	0.974	0.958	0.964

(c) The case E-E

Weight	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
0.0	0.958	0.931	0.938
0.1	0.965	0.933	0.943
0.2	0.962	0.935	0.942
0.3	0.963	0.936	0.944
0.4	0.962	0.936	0.943
0.5	0.961	0.935	0.943
0.6	0.960	0.933	0.941
0.7	0.958	0.933	0.940
0.8	0.957	0.931	0.938
0.9	0.956	0.931	0.938
1.0	0.956	0.931	0.938

(d) The case J-J

Weight	Avg. of Precision	Avg. of Recall	Avg. of *F*-measure
0.0	0.974	0.929	0.945
0.1	0.969	0.947	0.954
0.2	0.969	0.947	0.955
0.3	0.968	0.948	0.954
0.4	0.968	0.948	0.954
0.5	0.968	0.948	0.954
0.6	0.968	0.948	0.954
0.7	0.967	0.948	0.954
0.8	0.965	0.948	0.953
0.9	0.964	0.947	0.952
1.0	0.963	0.947	0.952

(e) The case EJ-EJ

Table 4. The results by the BP based ASCII art extraction method. Maximum averages of *F*-measure in each table are underlined

7. Conclusion

We conducted ASCII art extraction experiments using English texts and Japanese texts to compare two ASCII art extraction methods. The two methods are a run-length encoding (RLE) based method and a byte pattern (BP) based method. The two methods use ASCII art recognizers constructed by machine learning algorithms. The recognizers take attributes of a text as their inputs and tell if the text is an ASCII art object or not. The RLE based method uses the compression ratio of a text by run-length encoding as one of the attributes of the text. On the other hand, the BP based method uses the frequency distribution of byte

values in a text as the attribute of the text. According to our experimental results, the two methods are competitive if training texts and testing texts are in a same set of languages, but the RLE based method works better than the BP based method if training texts and testing texts are in different sets of languages. We confirmed that the distribution of compression ratios of texts is distinct between ASCII art objects and non-ASCII art objects while the frequency distribution of byte values is distinct between English texts and Japanese texts. This shows that the RLE based extraction method is more language-independent than the BP based method.

Byte range	English		Japanese	
	AA (%)	Non-AA (%)	AA (%)	Non-AA (%)
00_h - $0F_h$	0.0	0.0	0.0	0.0
10_h - $1F_h$	0.0	0.0	0.0	0.0
20_h - $2F_h$	69.1	20.2	9.8	3.9
30_h - $3F_h$	4.7	1.2	4.9	2.9
40_h - $4F_h$	4.1	2.6	0.1	0.4
50_h - $5F_h$	4.7	1.4	0.9	0.4
60_h - $6F_h$	2.5	50.1	1.0	3.4
70_h - $7F_h$	2.7	24.4	1.6	2.0
80_h - $8F_h$	6.6	0.0	39.1	29.9
90_h - $9F_h$	0.3	0.0	3.6	8.6
$A0_h$ - AF_h	0.2	0.0	2.6	10.8
$B0_h$ - BF_h	1.1	0.0	9.0	8.8
$C0_h$ - CF_h	0.0	0.0	0.2	0.1
$D0_h$ - DF_h	0.0	0.0	0.0	0.0
$E0_h$ - EF_h	4.0	0.0	27.0	28.9
$F0_h$ - FF_h	0.0	0.0	0.0	0.0

Table 5. The occurrence percentages of byte values in the two sets of texts *E* and *J*

Compression ratio	English		Japanese	
	AA (%)	Non-AA (%)	AA (%)	Non-AA (%)
0.0 - 0.2	2.7	0.0	1.1	0.0
0.2 - 0.4	15.8	0.0	8.7	0.2
0.4 - 0.6	21.3	0.0	15.0	0.2
0.6 - 0.8	22.6	0.0	18.5	0.1
0.8 - 1.0	16.8	0.0	16.8	0.2
1.0 - 1.2	9.8	0.2	12.1	0.6
1.2 - 1.4	5.7	0.2	9.0	0.5
1.4 - 1.6	3.1	0.8	6.4	1.0
1.6 - 1.8	1.1	4.5	4.1	6.6
1.8 - 2.0	1.1	94.2	8.4	90.6

Table 6. The percentages of compression ratio by RLE in the two sets of texts *E* and *J*

References

[1] Chang, Chih-Chung., Lin, Chih-Jen.(2011) LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2 (27) 1–27. 27. Software available at http://www.csie.ntu. edu.tw/~cjlin/libsvm.

[2] EGG. AAscan. http://www.gigafree.net/tool/aasupo/aascan.html (Retrieved on March 25, 2017).

[3] Tanioka, Hiroki., Maruyam, Minoru. 2005). Ascii Art Pattern Recognition using SVM based on Morphological Analysis. *Technical report of IEICE. PRMU*, 104 (670) 25–30, 2005. 0218.

[4] Hogenboom, Alexande., r Bal, Daniella., Frasincar, Flavius., Bal, Malissa., de Jong, Franciska., Kaymak, Uzay (2013). Exploiting emoticons in sentiment analysis. *In:* Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, p. 703–710, New York, NY, USA, 2013. ACM.

[5] Nakazawa, Masami., Matsumoto, Kazunori., Yanagihara, Tadashi., Ikeda, Kazushi., Takishima, Yasuhiro., Hoashi, Keiichiro. (2010). Proposal and its Evaluation of ASCII-Art Extraction, *In:* Proceedings of the 2nd Forum on Data Engineering and Information Management (DEIM2010), C9-4.

[6] The University of Waikato.Weka 3 - Data Mining with Open Source Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/ (Retrieved on March 25, 2017).

[7] Ptaszynski, Michal., Araki, Kenji., Dybala, Pawel., Rzepka, Rafal., Maciejewski, Jacek (2010). CAO: A Fully Automatic Emoticon Analysis System Based on Theory of Kinesics. *IEEE Transactions on Affective Computing*, 1 (1) 46–59, 2010.

[8] Quinlan, Ross. J. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[9] Suzuki, Tetsuya (2010). A Decision Tree-based Text Art Extraction Method without any Language-Dependent Text Attribute. *International Journal of Computational Linguistics Research*, 1 (1) 12–22, 2010.

[10] Suzuki, Tetsuya (2011). A Comparison of Whitespace Normalization Methods in a Text Art Extraction Method with Run Length Encoding. *In:* Hepu Deng, Duoqian Miao, Jingsheng Lei, and FuLee Wang, editors, *Artificial Intelligence and Computational Intelligence*, volume 7004 of *Lecture Notes in Computer Science*, p. 135–142. Springer Berlin Heidelberg, 2011.

[11] Suzuki, Tetsuya.(2015). Comparison of Two ASCII Art Extraction Methods: A Run-Length Encoding based Method and a Byte Pattern based Method. *In:* Proceedings of the 6th IASTED International Conference on Computational Intelligence. ACTA Press, 2015. 827-026.

[12] Suzuki, Tetsuya., Hayashi, Kazuyuki (2010). Text Data Compression Ratio as a Text Attribute for a Language-Independent Text Art Extraction Method. *In*: Proceedings of the 3rd International Conference on the Applications of Digital Information and Web Technologies, p. 513–518.

[13] Witten, Ian H., Frank, Eibe (2005). Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann.

[14] Xu, Xuemiao., Zhang, Linling., Wong, Tien-Tsin . Structure-based ascii art. *ACM Transactions on Graphics* (SIGGRAPH 2010 issue), 29 (4) 52. 1–52. 9, July 2010.

[15] Yamamoto, Yuki., Kumamoto, Tadahiko., Nadamoto, Akiyo (2014). Role of emoticons for multidimensional sentiment analysis of twitter. *In*: Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services, iiWAS '14, p. 107–115, New York, NY, USA, 2014. ACM.

[16] Yergeau, F. UTF-8, a transformation format of ISO 10646. https://tools.ietf.org/html/rfc3629 (Retrieved on March 26, 2017).