Extraction and Classification of Facemarks with Kernel Methods

Yuki Tanaka*
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku
Yokohama, JAPAN
yuki-t@lr.pi.titech.ac.jp

Hiroya Takamura
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku
Yokohama, JAPAN
takamura@pi.titech.ac.jp

Manabu Okumura
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku
Yokohama, JAPAN
oku@pi.titech.ac.jp

ABSTRACT

We propose methods for extracting facemarks (emoticons) in text and classifying them into some emotional categories. In text-based communication, facemarks have gained popularity, since they help us understand what writers imply. However, there are two problems in text-based communication using facemarks; the first is the variety of facemarks and the second is lack of good comprehension in using facemarks. These problems are more serious in the areas where 2-byte characters are used, because the 2-byte characters can generate a quite large number of different facemarks. Therefore, we are going to propose methods for extraction and classification of facemarks. Regarding the extraction of facemarks as a chunking task, we automatically annotate a tag to each character in text. In the classification of the extracted facemarks, we apply the dynamic time alignment kernel (DTAK) and the string subsequence kernel (SSK) for scoring in the k-nearest neighbor (k-NN) method and for expanding usual Support Vector Machines (SVMs) to accept sequential data such as facemarks. We empirically show that our methods work well in classification and extraction of facemarks, with appropriate settings of parameters.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Natural Language Processing; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms

Experimentation

Keywords

affect analysis, emoticon, facemark, kernel methods, SVM

*Currently Yuki Tanaka works at SHARP corporation.

IUI'05, January 9–12, 2005, San Diego, California, USA.

1. INTRODUCTION

Though the multi-modal communication has recently become available, the text-based communication such as e-mail, Internet Chat, B.B.S. is still quite popular and easier to use. In the text communication, people often use facemarks or emoticons to deliver their emotions more smoothly. Automated recognition and comprehension of facemarks thus improves affect analysis in human communication. Appearance of facemarks will help understand writer's emotions. Read-aloud systems for emails or web pages, which are currently obstructed by facemarks, on the contrary, can make good use of facemarks to produce more emotional voice by changing its tone.

In addition to that, facemarks sometimes must be removed out from text when we are focusing on language analysis on the conversation through e-mail or Internet Chat. Also in such a situation, recognition of facemarks is important as a preprocessing.

However, there are very few computer softwares that recognize where facemarks are and what facemarks mean. One of the reasons is the variety of facemarks. Users can create new facemarks, which cannot be recognized with a simple string matching. Moreover, it is also very difficult to estimate what the facemarks mean.

First, we are going to show how to extract facemarks in text by regarding the extraction of facemarks as a chunking task. We do not resort to "face recognition" methods, which usually require heavy computation. Chunking is a task of finding a chunk (that is, a continuous subsequence) from a sequence of objects. Named Entity extraction [8] and Base Noun Phrase detection [9] can be regarded as chunking tasks.

We will then classify the extracted facemarks into the 6 specific categories based on emotions and actions. Bag-of-characters features are not enough for capturing the differences of facemarks. Therefore, we use the dynamic time alignment kernel (DTAK) and the string subsequence kernel (SSK). Since those kernels take the information of character sequences into account, we expect they will show high performance in the classification. They are used as a similarity measure in the k-nearest neighbor (k-NN) method and the support vector machines (SVMs).

We conducted some experiments to evaluate the proposed methods.

2. FACEMARKS

In this section, we explain what the *facemarks* in this paper indicate. We then introduce 6 categories of facemarks based on their emotions or actions.

2.1 Facemarks

Facemarks are one of the expressions often used in Computer Mediated Communication. Some examples of Japanese facemarks are shown in Table 1. They express people's emotions and actions with the sequence of various kinds of letters and signs.

Table 1: Examples of Japanese facemarks

Expression	Facemark
Smiling	(^ ^)
Crying	(T o T)
Disgusted	(;´Д`)
Surprised	(￣|￣;)!!
Action (writing)	φ(..)
Smiling Wryly	(;￣ー￣A

There is another kind of facemarks called emoticons or smilies, as shown in Table 2. These "emoticons" are used mainly in the countries where language is expressed with 1-byte alphabets. In contrast, the "facemarks" used in other countries, where language is expressed with 2-byte alphabets, are mainly expressed with 2-byte characters as well as 1-byte characters. In this paper, by "facemarks", we mean the facemarks with 1- and 2-byte characters as in Table 1. The reason why we choose these facemarks as our target is that the 2-byte characters can generate various kinds of facemarks and automatic processing methods for those facemarks are desired.

Table 2: Examples of 1-byte emoticons

Expression	Emoticon
Smiling	:-)
Crying	:'(
Getting angry	:-(
surprised	:O
Wearing glasses	8-)
Using a headphone	[:)
Wearing a helmet	(|:-)

2.2 Category Definition

Generally, facemarks are classified into two coarse categories. One is the emotion of writers, and the other is the action of writers. Category "emotion" is split further into five fine categories; happy, sad, angry, surprised, and wry smile, which are modifications of classification by Ekman [3]. We classify facemarks into 6 categories in total.

Some examples of the facemarks in each category are shown in Table 3.

Table 3: The 6 categories for facemarks

Category	Examples
Happy	(^^), !(∂▼∂)!
Sad	(T o T), (、>_<)ノ
Angry	(▼、▼メ), (ーー;)
Surprised	(;･◆･), (° o°)
Wry smile	f(^^;, (;￣ー￣A
Actions	(^人^), m(__ __)m

3. EXTRACTION OF FACEMARKS

In this section, we explain how we extract facemarks in text.

Regarding extraction of facemarks as a chunking task, we chunk facemarks by classifying each character into the beginning, the inside or the outside of a facemark. Morphological information is used as features in chunking. In the next two subsections, we will explain how to annotate features on characters and give a brief introduction of the annotation and the chunking.

3.1 Feature annotation

As the first step, we automatically assign POS tags to the characters in each sentence. *ChaSen* [7] is used in the morphological analysis. After the analysis, the names of the POSs and the positions in a POS shown in Table 4 are annotated to each character. For example, if a word consists of only one character, we give S tag to the character. B is given to the first character in a multi-character word. E is given to the last character. I is given to the other characters. These tags are used as features for the chunking in the second step.

In addition, we annotate the POS types of characters. Nouns, verbs, adjectives, auxiliary verbs and signs are some of the examples of types. The intuition behind the use of these kinds of features is that we a priori know, for example, that signs tend to be a part of facemarks and auxiliary verbs tend to precede facemarks. Please note that all these annotations are done automatically.

Table 4: Position of POS tags

Tag	Description
S	Characters in words consisting of a single character
B	The first characters in words consisting of multi characters
E	The last characters in words consisting of multi characters
I	The other characters in words consisting of multi characters

3.2 Chunking with SVMs

With the features of the characters, we identify the beginning characters of facemarks. Character-based chunking of Named Entities is proposed by Asahara et al. [1]. As in their method, we extract facemarks by tagging a BIO-tag (Table 6) to each character in text.

Table 5: An example of features for facemark extraction; characters at positions i to $i+2$ correspond to a facemark. Analysis proceeds from left to right on text (from top to bottom in this table)

Position	Character	POS Type	Position in POS	facemark tag
i-3	で (de)	AUX	B	O
i-2	す (su)	AUX	E	O
i-1	。	Sign-Period	S	O
i	(	Sign-Bracket-Open	S	B
i+1	^	Sign-Other	S	
i+2	^	Sign-Other	S	
i+3	)	Sign-Bracket-Close	S	

We use *yamcha* [4][1], an SVM-based chunker. In the SVM-based chunking, our goal is to derive hyper spaces for dividing a set of training data. Assume that there is a set of training examples x_i with labels y_i :

$$(x_1, y_1), \cdots, (x_N, y_N),\ x_i \in R^n, y_i \in \{+1, -1\}. \quad (1)$$

An SVM gives the decision function $f(x) = sgn(g(x))$ for a new example x:

$$f(x) = sgn(W \cdot x + b) \quad (2)$$

$$= sgn(\sum \alpha_i y_i K(x_i, x) + b) \quad (3)$$

Weights α_i and bias b are parameters of SVMs. $K(x_i, x)$ is called a kernel function which works as a similarity function between two examples. We used the polynomial function of degree 2 as a kernel for extraction of facemarks.

We extend SVMs, a binary classifier, to an n-class classifier. The *one-vs-rest* method and the *pairwise* method are popular for the extension. The one-vs-rest method requires n binary classifiers, each of which distinguishes one class from others. The pairwise method requires ${}_nC_2$ binary classifiers, each of which distinguishes one class from another class. We use the pairwise method because of the high efficiency in learning.

We chunk facemarks by classifying each character with the annotated features. Tags shown in Table 6 are used for chunking. Table 5 shows an example. The features that we use are characters, POS types and position tags for POSs in a fixed window size, as well as preceding facemark tags. For example, the features in the solid box of Table 5 are used for estimating facemark tag B at position i.

Table 6: Tags for chunking

Tag	Description
B	The first character of a facemark
I	The other characters in a facemark
O	The characters outside of any facemark

4. CLASSIFICATION OF FACEMARKS

In this section, we introduce some methods for the classification of facemarks.

In text classification, we often focus on which words occur in a text, ignoring the order of words. This simple feature indexing is called bag-of-words method. If we regard a facemark as a short text consisting of characters (instead of words), facemarks can be classified by bag-of-characters features. However, in the classification of facemarks, we have to take the sequence of characters into account, because each character often has high ambiguity in facemarks. Therefore, it is essential for improving the performance of classification to use the information of character sequence.

For this task as well, we use SVMs. When we would like to handle character sequences in SVMs, we have only to provide an appropriate kernel function for that purpose.

We will use the following kernel functions for facemarks, which are actually sequences of characters. Each input instance for these kernels is a sequence of characters denoted by S or T, whereas each input instance for Equation (2) was a vector x in R^n.

Please note that while the facemark extraction is reduced to character classification into BIO-tags, the facemark classification is reduced to character sequence classification.

4.1 Dynamic Time Alignment Kernel (DTAK)

Dynamic Time Alignment Kernel (DTAK) [11], which is also called Dynamic Time Warping Kernel (DTWK) [2], is one of the kernels that are used to apply the original SVMs for variable length data sequences. In DTAK, the characters in two facemarks are aligned to compute the similarity of the facemarks.

Assume that we have two data sequences:

$$S = (s_1, s_2, \cdots, s_{|S|}), \quad T = (t_1, t_2, \cdots, t_{|T|}).$$

If those two sequences are equal in length, ($|S| = |T|$), we can easily obtain the inner product between S and T:

$$S \cdot T = \sum_{k=1}^{|s|=|t|} s_k \cdot t_k.$$

However, if they are different in length, we cannot calculate the inner product directly. In DTAK,the characters are aligned using dynamic programming (DP), where *insertion*, *omission* and *replacement* are allowed with penalty.

Define that $d(s_i, t_j)$ is the distance between s_i and t_j. If $s_i = t_j$, $d(s_i, t_j) = 0$, or if $s_i \neq t_j$, $d(s_i, t_j) = p$, where p is the cost of replacement. If the distance between subsequences $S = (s_1, s_2, \cdots, s_i)$ and $T = (t_1, t_2, \cdots, t_j)$ is given by $g_{S,T}(i, j)$, then $g_{S,T}(i, j)$ is recursively calculated using

[1]Available at `http://chasen.org/~taku/software/yamcha/`.

DP as follows:

$$\begin{aligned} g_{S,T}(0,0) &= 0 \\ g_{S,T}(i,0) &= g_{S,T}(i-1,0)+r, i=1,2,\cdots,|S| \\ g_{S,T}(0,j) &= g_{S,T}(0,j-1)+q, i=1,2,\cdots,|T| \end{aligned}$$

$\forall i \in \{1,2,\cdots,|S|\}$
$\forall j \in \{1,2,\cdots,|T|\}$

$$g_{S,T}(i,j) = \min \left\{ \begin{array}{l} g_{S,T}(i-1,j)+r \\ g_{S,T}(i-1,j-1)+d(i,j) \\ g_{S,T}(i,j-1)+q \end{array} \right. \quad (4)$$

$$D(S,T) = g_{S,T}(|S|,|T|) \quad (5)$$

where q is the cost of insertion, r is the cost of omission. We apply $D(S,T)$ for the kernel in SVMs.

We should note that DTAK is not a valid kernel, because DTAK lacks semi-definiteness. However, DTAK practically performs well in some tasks such as handwriting recognition [2].

4.2 String Subsequence Kernel (SSK)

String Subsequence Kernel (SSK) proposed by Lodhi et al. [6] is also one of the kernels for sequential data. In SSK, the similarity of two strings are measured by the weighted number of all the common substrings. If two strings have many substrings in common, they are regarded as similar. Although it seems difficult to compute the number of all the common substrings, this computation is realized as explained below.

Define Σ as a class of all the characters including alphabets, Japanese letters, and signs. String $S=(s_1,s_2,\cdots,s_{|S|})$ is a sequence of the characters in Σ. $|S|$ is defined as the length of the string S. String $U=(u_1,u_2,\cdots,u_{|U|})$ is defined similarly. If there are indices $\mathbf{i}=(i_1,i_2,\cdots,i_{|U|})$ and $u_j=s_{i_j}$ on the condition of $1 \le i_1 <,\cdots,< i_{|U|} \le |S|$, we call that U is a subsequence of S and express as $U=S[\mathbf{i}]$. The length of subsequence U is defined as $l(\mathbf{i})=i_{|U|}-i_1+1$.

The feature $\phi_U(S)$ of a string S for U is defined as

$$\phi_U(S) = \sum_{\mathbf{i}:U=S[\mathbf{i}]} \lambda^{l(\mathbf{i})} \quad (6)$$

for $\lambda < 1$. λ is the discount factor used to give weights to subsequences according to their lengths. If the component characters of a substring are positioned closely to each other, the substring is heavily weighted. If the gap between component characters is large, the substring is less weighted.

The string kernel of length n (the weighted sum of the number of all the common substrings of length n), is written as:

$$\begin{aligned} K_n(S,T) &= \sum_{U\in\Sigma^n} \langle \phi_U(S)\cdot\phi_U(T)\rangle \\ &= \sum_{U\in\Sigma^n} \sum_{\mathbf{i}:U=S[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:U=T[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{U\in\Sigma^n} \sum_{\mathbf{i}:U=S[\mathbf{i}]} \sum_{\mathbf{j}:U=T[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \end{aligned}$$

For defining a recursive algorithm, we define this function as below;

$$K_i^{'}(S,T) = \sum_{U\in\Sigma^i} \sum_{\mathbf{i}:U=S[\mathbf{i}]} \sum_{\mathbf{j}:U=T[\mathbf{j}]} \lambda^{|S|+|T|-i_1-j_1+2} \quad (7)$$

K_n is derived with a recursive calculation as:

$$\begin{aligned} K_0^{'}(S,T) &= 1, \forall S, \forall T, \\ K_i^{'}(S,T) &= 0, if\ \min(|S|,|T|) < i, \\ K_i(S,T) &= 0, if\ \min(|S|,|T|) < i, \end{aligned}$$

$\forall i \in \{1,\cdots,n-1\}$

$$\begin{aligned} K_i^{'}(Sx,T) &= \lambda K_i^{'}(S,T) \\ &+ \sum_{j:t_j=x} K_{i-1}^{'}(S,T[1:j-1])\lambda^{|T|-j+2} \end{aligned} \quad (8)$$

$$\begin{aligned} K_n(Sx,T) &= K_n(S,T) \\ &+ \sum_{j:t_j=x} K_{n-1}^{'}(S,T[1:j-1])\lambda^2 \end{aligned} \quad (9)$$

By taking summation with respect to n, we obtain the string kernel:

$$K_{str}(S,T) = \sum_n K_n(S,T). \quad (10)$$

To remove the bias caused by the length of strings, we normalize the kernel:

$$\hat{K}_{str}(S,T) = \frac{K_{str}(S,T)}{\sqrt{K_{str}(S,S)K_{str}(T,T)}}. \quad (11)$$

4.3 Additional application of kernels

The kernel functions such as polynomial kernels and RBF kernels can be used on top of DTAK or SSK:

$$K_{str+poly}(S,T) = (\hat{K}_{str}(S,T)+1)^d, \quad (12)$$

$$\begin{aligned} K_{str+RBF}(S,T) &= \exp\{-(\hat{K}_{str}(S,S)-2\hat{K}_{str}(S,T) \\ &+\hat{K}_{str}(T,T))/2\sigma^2\}. \end{aligned} \quad (13)$$

We expect that with this kind of additional use of kernels makes clearer the similarity and the dissimilarity of facemarks. We should note that additional kernels cannot change results of k-NN, as long as the additional kernels are monotonic functions with respect to the values of argument kernels: $\hat{K}_{str}$. In contrast, SVMs can yield different results, when additional kernels are used.

5. EXPERIMENTS

In this section, we evaluate the proposed methods for extraction and classification of facemarks. The 10-fold cross validation was conducted for each experiment.

First, we annotated facemark tags on the 12261 sentences that we obtained from some Japanese B.B.S.s in the Internet. 913 instances of facemarks appeared on this corpus. We use the annotated sentences in the evaluation of the facemarks extraction. For comparing with pattern-matching extraction using a facemark list, we collected 14 facemark dictionaries from the Internet. As a result, we obtained a list consisting of 5796 different facemarks. We use this list for pattern-matching extraction. As evaluation measures for extraction, we use precision and recall. Precision is defined as the number of the correctly extracted facemarks divided by the number of all the extracted facemarks. Recall is defined as the number of the correctly extracted facemarks divided by the number of all the facemarks in the dataset.

We prepared another dataset consisting of 1075 different facemarks extracted from the B.B.S. corpus. We manually classified the facemarks into 6 categories; “Happy (435)”, “Sad (184)”, “Angry (71)”, “Surprised (102)”, “Action (152)” and “Wry Smile (131)”. We use these 1075 facemarks in the evaluations of the facemarks classification. Please note that these 1075 facemarks are different from each other. As an evaluation measure for classification, we use F-measure, which is defined as

$$\text{F-measure} = \frac{2 \times recall \times precision}{recall + precision}. \quad (14)$$

5.1 Evaluation of Extraction

We first evaluated recall of pattern-matching extraction method. By the method, we extracted 635 facemarks out of 913. In the evaluation, if there are multiple matching patterns, we select the longest pattern. The precision and recall of the pattern-matching extraction are respectively 84.0% and 69.6%.

The simple pattern-matching method fails to extract, for example, facemark （＞д＜） , although the facemark list contains facemark （＞@＜） . As indicated by this error, the pattern-matching method cannot be flexible enough to extract new facemarks, even if the list has similar facemarks.

We then evaluate our method for facemark extraction. In our method, the performance of the facemark extraction can be affected by window-size. In order to estimate the best window-size, we conducted the same facemark extraction task with different window-sizes, from 2 to 10 characters in forward and backward window-sizes. The results are shown in Figure 1. the horizontal axis corresponds to the width of each window-size, and the vertical axis corresponds to the value of precision and recall. As a result of the experiment, we found that 4 is the best window size for our extraction task.

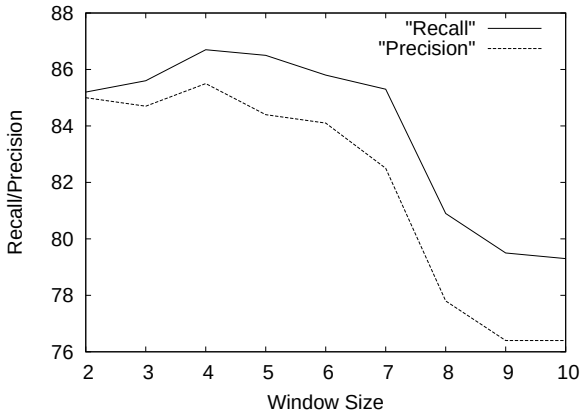

Figure 1: Results with different window-size (horizontal axis : window-wize, vertical axis : performance)

Many of the errors in the extraction are related to lack or excess of some neighbor characters of facemarks shown in Table 7. However, the number of the errors is much smaller than the one of pattern-matching method. Fatal errors except them are few. Most of the errors are mischunking of symbols with brackets such as “(Thu)”. The precision and the recall of our method are respectively 85.5% and 86.7% in the highest, which are much better than those of the pattern-matching method.

Table 7: Errors in extraction

あったんですね～φ（__ __）メモメモ ((I) haven't known that. φ（__ __） taking notes)	
correct	φ（__ __）メモメモ
wrong	（__ __）メモメモ
ちなみにσ（ﾟ__ﾟ；は、上溝の住人です。 (incidentially, σ（ﾟ__ﾟ； live in Kami-mizo.)	
correct	σ（ﾟ__ﾟ；
wrong	（ﾟ__ﾟ；
「...したいよ（；ｏ；）」とゆーのは... (“(I) want to do ･･･（；ｏ；）”because･･･)	
correct	（；ｏ；）
wrong	（；ｏ；）」
ございましたｍ（＿＿） ｍえ～っと、材料･･･ ((Thank you) very much. ｍ（＿＿） ｍ Well, stuffs･･･)	
correct	ｍ（＿＿） ｍ
wrong	ｍ（＿＿） ｍえ～っと、

5.2 Evaluation of Classification

In the classification of facemarks, we used mainly two approaches. One of the two is called the k-nearest neighbor (k-NN) method. In this method, as a similarity measure, we use DTAK and SSK. In 1-NN, the category of the training facemark that is most similar to a given test facemark is assigned to this test facemark. In 10-NN, the category of a test facemark is predicted by majority voting of the 10 most similar training facemarks.

The other approach is SVM classification. We evaluate the performance of SVM classifiers with DTAK and SSK, as well as bag-of-characters.

- *k-NN classification*
  In the left half of Table 8, we show the results of k-nearest neighbor (k-NN) classification with DP-matching scoring (that is, DTAK).

  10-NN with DTAK technique showed higher performance than 1-NN with DTAK. However, it is still inferior to the 1-NN with SSK technique and SVM classifiers as shown in the next subsection.

  It is also interesting that 10-NN is superior to 1-NN in DTAK while 1-NN is superior to 10-NN in SSK.

- *SVM classification*
  The results of SVM classifications with DTAK, SSK, and the linear kernel (bag-of-characters) are shown in the left half of Table 8.

  Both DTAK and SSK showed higher performance than the bag-of-characters feature.

  Without additional polynomial or RBF kernels, the result of SVM+DTAK is superior to the result of k-NNs+DTAK, while 1-NN+SSK is a bit superior to the one of SVM+SSK.

  We then checked the performance of the three SVM methods with additional kernels. We used polynomial and RBF kernels as the additional kernels. With polynomial kernel, SVMs take into account the combinations of features. With RBF kernel, effect of more

Table 8: Results of classifications

	k-NN				SVM					
	DTAK		SSK		Bag-of-characters		DTAK		SSK	
	$k=1$	$k=10$	$k=1$	$k=10$		+poly.		+poly.		+RBF
Happy	19.0	83.3	89.9	60.4	76.9	92.5	91.1	94.6	84.4	94.0
Sad	46.1	71.2	86.5	10.8	77.3	88.2	82.5	92.0	89.8	92.1
Angry	18.0	57.1	77.0	15.3	45.6	85.7	72.4	88.3	83.5	87.9
Surprised	30.6	58.2	77.3	9.3	73.0	86.0	74.1	84.9	79.3	85.7
Action	61.2	61.7	61.4	1.4	58.1	71.7	68.4	79.4	57.0	78.3
Wry Smile	27.2	38.9	78.8	37.4	49.0	89.5	81.4	91.1	82.0	94.0
micro average [10]	32.0	67.0	82.6	45.5	70.1	87.8	82.6	90.4	81.9	90.3

similar training examples is exponentially larger than that of less similar examples.

As the table shows, the additional polynomial kernel improves the performances of all the three kernels. For SSK, RBF improves its performance most.

- *Influence of parameter λ of SSK*
  Although SSK improved the classification results for 1-NN and SVM, the above results are obtained only with an appropriate setting of parameter λ. To clarify the influence of this parameter, we show the results with different λ in Tables 9 and 10.

Table 9: k-NN+SSK classification result with different λ

λ	0.05	0.15	0.25	0.35	0.45
F-measure (10-NN)	54.6	48.8	48.5	48.0	45.9
F-measure (1-NN)	82.7	82.4	82.4	82.5	82.5
λ	0.55	0.65	0.75	0.85	0.95
F-measure (10-NN)	46.0	44.3	44.6	45.5	51.1
F-measure (1-NN)	82.7	82.8	82.6	82.6	81.1

Table 10: SVM+SSK classification with different λ, the linear kernel

λ	0.05	0.15	0.25	0.35	0.45
F-measure	68.6	67.8	67.8	67.8	75.6
λ	0.55	0.65	0.75	0.85	0.95
F-measure	77.3	79.9	80.3	81.9	78.5

We thus conclude that our classification methods work well, though an appropriate setting of parameters is required in practice.

In our experiments, SVM classifications both with DTAK plus polynomial kernel and SSK plus RBF kernel show the best performance of all the methods we introduced in this paper.

We succeeded in classifying facemarks with high accuracy. However, there are some errors caused by ambiguous usage of letters. A substring " ^^ " usually suggests the facemark belongs to "Happy", because this substring expresses smiling eyes. Counter-examples are

(^∧^) (praying something with hands clasped),

and

(^ o ^)。＿o∠=※ (setting off a fire cracker).

Their actual category is "Action", but the predicted category is "Happy".

In cases of emotion-based categories "Happy", "Sad", "Angry", "Surprised", and "Wry smile", the performances of the two SVM methods are high enough. However, compared with the high performances in these categories, the performance in category "Action" is relatively low. We think that it is because this category has more variety than the other categories have.

6. CONCLUSION

We proposed a method for extracting facemarks from sentences with morphological informations and SVM-based chunking.

We also proposed to use DTAK and SSK combined with k-NN and SVM, which can effectively deal with sequential data, to classify facemarks into 6 categories according to their emotions or actions.

We evaluated the proposed extraction and classification methods, and showed that these methods have a high capacity in the task.

However, in order to obtain good classification results, we require an appropriate setting of parameters. We regard this problem as future work. We believe that a simple empirical method for parameter settings such as cross-validation would perform well.

A promising and interesting extension to our method is incorporation of similarity between characters. For example, we know that ":" and ";" are similar, but the current methods cannot capture this similarity. With this extension, the method would get more robustness to new facemarks.

Although we focused on extraction and classification in this paper, selecting or creating an emoticon suitable for a document [5] is also an interesting research direction related to facemarks.

7. REFERENCES

[1] M. Asahara and Y. Matsumoto. Japanese named entity extraction with redundant morphological analysis. In *Proceedings of Human Language Technology and the North American Chapter of the Association for Computational Linguistic (HLT-NAACL 2003)*, pages 8–15, 2003.

[2] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines - a kernel approach. In *Proceedings of the 8th*

International Workshop on Frontiers in Handwriting Recognition (IWFHR), pages 49–54, 2002.

[3] P. Ekman. Facial expression of emotion: New findings, new questions. In *Psychological Science*, volume 3, pages 34–38, 1992.

[4] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 192–199, 2001.

[5] H. Liu, H. Lieberman, and T. Selker. A model of textual affect sensing using real-world knowledge. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces (IUI 2003)*, pages 125–132, 2003.

[6] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. J. C. H. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[7] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, and M. Asahara. *Japanese Morphological Analysis System ChaSen version 2.2.1*, 2000.

[8] E. F. T. K. Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Computational Natural Language Learning*, pages 155–158, 2002.

[9] E. F. T. K. Sang and S. Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of Conference on Computational Natural Language Learning*, pages 127–132, 2000.

[10] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[11] H. Shimodaira, K. ichi Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Proceeding of the Neural Information Processing (NIPS2001)*, pages 921–928, 2002.