文字ベースのコミュニケーションにおける顔文字に関する研究

田中 裕紀[†]　高村 大也[‡]　奥村 学[‡]

Research on Face Marks used in a Text-based Communication

Yuki TANAKA[†]　Hiroya TAKAMURA[‡]　Manabu OKUMURA[‡]

1　はじめに

近年利用機会の増えたコンピュータを介したコミュニケーションのうち, インターネットチャット, 電子掲示板, 電子メール等のような文字ベースのコミュニケーションにおいては, 文字だけでは表しにくい意味や意図を表すために, 顔文字がよく用いられている. しかし現在, 顔文字を含む文字ベースのコミュニケーションをコンピュータ上で扱う際に, 顔文字が有効に処理されている例はまだ少ない [5]. この理由には, 第一に顔文字の多様性が挙げられる. 現在用いられている顔文字の種類は既に多種多様にある上に, ユーザである人間は, 新たに自由に顔文字を創り出し続けるためである. 第二に, 顔文字の表す感情的な情報を適切に得る事が困難であるためである.

そこで本研究では, 顔文字を含む文章から顔文字を抽出し, その顔文字の表す感情的な情報を適切に判断する事を目的とする. これにより現在ではノイズとなっている要素を取り除けるばかりか, 文章からだけでは得られなかった, 顔文字からの情報をも新たに用いることが出来るため, 話者の意図をより適切に理解できる. まず初めに, 顔文字を含む文章から顔文字の出現する位置を, 形態素解析とチャンキングを組み合わせて特定する手法を提案する. 次に顔文字を, その表す感情に応じて数種類のカテゴリに分類する手法として, Dynamic Time Alignment Kernel (DTAK)[1] と, String Subsequence Kernel (SSK)[2] を用いた手法を提案する. そしてこれらの手法を用いた Nearest Neighbor (NN) や Support Vector Machine(SVM) により, 顔文字を, その表す感情に基づいて分類し, その分類精度について評価する.

2　顔文字

顔文字は, 表 1 に挙げるような, 文字列を用いた人間の顔の表情や動作の表現の一つである.

さらに顔文字には「＼（＾＾＼）♪（／＾＾）／」の様に顔を表す要素が複数連なったものや, 「（＾＾）ニコ」の様に顔の要素に言葉が連なるものもある. 本研究における顔文字の定義では, これらを全て単一の顔文字とする. これは複数の要素をまとめて取り扱っても, 必要であれば後処理によって, それらの要素を分離することも可能だからである.

またここで取り上げる顔文字と類似するものに, 主に欧米などで用いられている, エモティコン (emoticon), またはスマイリー (smiley) と呼ばれる表現がある. これらもまた, ここで言う顔文字と同様に顔の表情を表すものも多いが, 使用される文化や言語が異なることもあり, その利用方法は必ずしも同一とは言えない. そこで本研究では, エモティコン・スマイリーは, 日本で用いられている顔文字とは異なる種類のものであるとし, これを含めない事とする.

表 1: 代表的な顔文字

表す意味	顔文字
笑っている様子	（＾＾）
泣いている様子	（ＴｏＴ）
困っている様子	（;´д｀）
驚いている様子	（￣□￣；）!!
文章を書く様子	φ（..　）
苦笑いしている様子	（；￣ー￣Ａ

次に顔文字の表現する意図について考える. 顔文字の表現する意図には, 大きく分けて人間の感情と, 人間の動作の二種類がある. 本研究ではこのうち感情を表す顔文字を, さらに「喜んでいる」「悲しんでいる」「怒っている」「驚いている」「苦笑いをしている」の 5 つに分類する. この 5 分類に, 動作を表す顔文字の 1 分類を加えた, 合計 6 分類を顔文字の分類として定義し, 後に述べる顔文字の分類において利用する. その 6 分類の例を, 表 2 に挙げる.

表 2: 顔文字の分類カテゴリ

分類名	例
喜んでいる	（＾＾）, !（∂▼∂）!
悲しんでいる	（ＴｏＴ）, (, ＞＿＜）ノ
怒っている	(▼､▼メ), （ーー；）
驚いている	(;･◆･), (°ｏ°)
動作を表す	（＾人＾）, ｍ（＿＿）ｍ
苦笑い	ｆ（＾＾;, （；￣ー￣Ａ

3　顔文字の抽出

本研究では, 文章中の未知語位置の特定などにも用いられている [4], SVM を用いたチャンカーである *Yamcha*[3] を用い, 以下の様な手順で顔文字抽出を行う.

3.1　文章の形態素解析

顔文字を含んだ文章を形態素解析器「茶筅」[6] により形態素解析し, その形態素情報を得る. 顔文字を含んだ文章においては, 顔文字の付近において正確な形態素

[†]東京工業大学大学院 総合理工学研究科
Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology　yuki-t@lr.pi.titech.ac.jp
[‡]東京工業大学 精密工学研究所
Precision and Intelligence Laboratory, Tokyo Institute of Technology　{takamura,oku}@pi.titech.ac.jp

表 3: チャンキングに用いる素性

位置	文字種	品詞	形態素中での位置	顔文字タグ
i-3	で	助動詞	B	O
i-2	す	助動詞	E	O
i-1	。	記号-句点	S	O
i	（	記号-括弧開	S	B
i+1	^	記号-その他	S	
i+2	^	記号-その他	S	
i+3	）	記号-括弧	S	

解析が行えていない場合もあるが, その様な場合の形態素情報も例外なく全て用いる.

3.2 文字単位への形態素情報のタグ付け

文章を文字単位に分割し, 各文字に対して, その文字の属する形態素の種類や形態素中での位置, またその文字種をタグ情報として付与する. 形態素中での位置を表すタグには, 表 4 に示すタグを用いる.

表 4: 形態素中での位置情報を表すタグ

タグ	タグが付与される文字種
S	一文字で形態素を構成する文字
B	二文字以上の形態素中の最初の文字
E	二文字以上の形態素中の最後の文字
I	二文字以上の形態素中の 最初でも最後でもない文字

3.3 パターンの学習及び文字のチャンキング

各文字に付与されたタグ情報を素性として, 学習時には *Yamcha* を用いて顔文字の出現パターンを学習する. 抽出時には, その学習結果を用いて顔文字の構成要素となりやすい文字を推定し, その連続した文字列のチャンキング (くくり出し) を行う.

Yamcha は SVM を利用した分類器を用いて, 文章中の各文字を, タグによって付与された素性に基づいて文字単位でチャンキングしていく. チャンキングの結果, 表 5 のタグが, 各文字に対して順に付与される.

表 5: 顔文字のチャンキングに用いるタグ

タグ	タグが付与される文字種
B	顔文字を構成する最初の文字
I	顔文字を構成する, 最初以降の文字
O	顔文字以外の文字

表 3 にチャンキングの例を示す. この例においては, i 番目の文字である「**(**」のタグを決定するために, 枠で囲まれた前後 3 文字のタグ情報を素性として用いている. その結果, **B** というタグが「**(**」に対して付与される.

4 顔文字の分類手法

あるデータ列の類似性を評価する際に用いる手法の代表として, 含まれる素性の数に注目するもの (bag-of-words) がある. しかし顔文字においては, 文字の順序も重要な情報である. そこで, このようなデータの並びを考慮に入れるために考えられた 2 つの手法について説明する.

4.1 Dynamic Time Alignment Kernel (DTAK)

2 つの文字列 $S = (s_1, \cdots, s_{|S|}), T = (t_1, \cdots, t_{|T|})$ があるとする. この時 2 つの文字列同士の類似度を $D(S,T)$, それぞれの文字列中の文字同士の類似度を $d(s_i, t_j)$ と表すとき, $d(s_i, t_j)$ から $D(S,T)$ を導く.

$s_i = t_j$ の時の類似度を $d(s_i, t_j) = p$, $s_i \neq t_j$ の時の類似度を $d(s_i, t_j) = q$ とし, またギャップ $*$ との類似度を $d(s_i, *) = r$ とする。これらを用いて文字列 S と T を最終的に一致させるまでに得られる類似度の総和のうち, 最大のものを S と T との類似度 $D(S,T)$ とする.

S の先頭から i 番目までの部分文字列 $(s_1, \cdots, s_i)$ と, T の先頭から j 番目までの部分文字列 $(t_1, \cdots, t_j)$ との類似度を $g(i,j)$ とする. この $g(i,j)$ は, Dynamic Programming の最適性原理を用いて, 以下の漸化式で表される:

$$g(0,0) = 0 \quad (1)$$

$$g(i,0) = g(i-1,0) + q, i = 1, 2, \cdots, |S| \quad (2)$$

$$g(0,j) = g(0,j-1) + q, i = 1, 2, \cdots, |T| \quad (3)$$

For $i = 1, 2, \cdots, |S|, \quad j = 1, 2, \cdots, |T|,$

$$g(i,j) = \begin{cases} g(i-1,j) + q \\ g(i-1,j-1) + d(i,j) \\ g(i,j-1) + q \end{cases} \quad (4)$$

$$D(S,T) = g(|S|, |T|). \quad (5)$$

4.2 String Subsequence Kernel (SSK)

SSK においては, 対象となる文字列から生成される部分文字列の一致具合を利用する. 例えば, **(^o^)** と **(^ー^)** の例では, 表 6 に挙げるような部分文字列が生成され, その一致度は, 文字同士の距離に対する重み $\lambda \leq 1$ を用いて,

$$2(\lambda^2+\lambda^4)^2+\lambda^{5\times2}+\lambda^{3\times2} = \lambda^{10}+2\lambda^8+5\lambda^6+2\lambda^4 \quad (6)$$

表 6: 部分文字列の一致具合の例

	(^	(o	()	^ o	^^	^)
(^ o ^)	$\lambda^2+\lambda^4$	λ^3	λ^5	λ^2	λ^3	$\lambda^2+\lambda^4$
(^ − ^)	$\lambda^2+\lambda^4$		λ^5		λ^3	$\lambda^2+\lambda^4$

	o ^	o)	(−	^ −	− ^	−)
(^ o ^)	λ^2	λ^3				
(^ − ^)			λ^3	λ^2	λ^2	λ^3

と表すことが出来る.

任意の n 文字までの文字の一致までに拡張する事もできる. この時 S から得られる部分文字列 U は, インデックス集合 $i=(i_1,i_2,\cdots,i_{|u|})$ $(1 \le i_1 <,\cdots,< i_{|u|} \le |s|)$ を用いて, $u_j = s_{i_j}$ と表す事ができる. これを, $U = S[i]$ と表す. この時の部分文字列の先頭から末尾までの距離 $l(i)$ を, 飛ばした部分も含めて, $l(i) = i_{|U|} - i_1 + 1$ で表す.

この時, 部分文字列 U に関する文字列 S, T の一致度 $\phi_U(S)$ は, それぞれ以下の様に表される:

$$\phi_U(S) = \sum_{i:U=S[i]} \lambda^{l(i)}, \quad \phi_U(T) = \sum_{i:U=T[i]} \lambda^{l(i)}. \quad (7)$$

よって, U に関する S, T の共通する一致度は, $\phi_U(S)$ と $\phi_U(T)$ の積で表される. この時 S, T における, $l(i) = n$ までの部分文字列の一致を全て考慮に入れた一致度 $K_n(S,T)$ は, Σ を, 文字列を構成する全ての文字種として, 下の式で求められる:

$$K_n(S,T) = \sum_{U\in\Sigma^n} \langle \phi_U(S)\cdot\phi_U(T)\rangle \quad (8)$$

$$= \sum_{U\in\Sigma^n} \sum_{i:U=S[i]} \lambda^{l(i)} \sum_{j:U=T[j]} \lambda^{l(j)} \quad (9)$$

$$= \sum_{U\in\Sigma^n} \sum_{i:U=S[i]} \sum_{j:U=T[j]} \lambda^{l(i)+l(j)}. \quad (10)$$

Lodhi ら [2] の提案する高速な計算法により, n が大きくても十分な速度で計算する事ができる.

5 評価実験

5.1 顔文字の抽出

まず初めに, 抽出の際の適切なウィンドウサイズを調べるため, ウィンドウサイズを変化させた時の各精度を比較した. その結果が図 1 である. その結果, ウィンドウサイズが前後 4 文字の場合が, 精度, 再現率ともに最も高く, それ以上ウィンドウサイズを広げて素性を増やしても, 逆に抽出精度が下がってしまう事が確認された. なお, ウィンドウサイズの前後の幅を異なる値にしたり, 解析方向を文末から文頭へと変えた場合についても調べたが, ウィンドウサイズ前後 4 文字の時よりも良い結果が得られる事はなかった.

次に, ウィンドウサイズ前後 4 文字のもとで, 12261 文中に出現する 913 文字をそれぞれ訓練データとテストデータに分割し, 訓練に用いたデータサイズの大きさが, 抽出性能に与える影響の評価を行った. その結果が表 7 である.

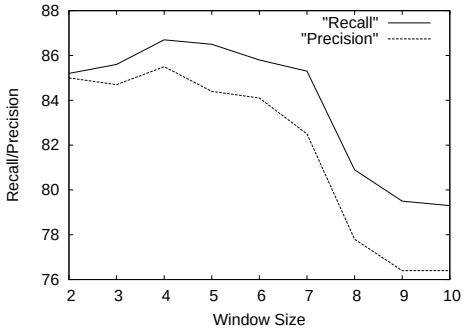

図 1: ウィンドウサイズと抽出性能との関係

表 7: 訓練データサイズと抽出性能との関係

訓練事例数	102	183	229	305	457
再現率	77.2	81.3	83.7	85.3	86.5
精度	72.9	78.2	81.0	86.9	86.7

傾向として, 訓練数が 400 程度以上になると, 精度, 再現率ともに上昇の幅は小さくなっている. よって, 913 のデータ全てを訓練データとして用いた場合は, そのサイズは十分妥当であると考えられる.

抽出に失敗した代表的な誤りの多くは, 開き括弧「(」以前や, 閉じ括弧「)」以降の文字列を顔文字としてチャンキングするかどうかの判断の誤りによるものであった. これには, 括弧内部では比較的素性のパターンが限られているのに対し, 括弧外部に出現しうる素性のパターンが多かった事と, 括弧外部にも要素を持つ顔文字の種類がそれほど多くなかった事などが考えられる.

5.2 顔文字の分類

事前に収集した 1075 種類の異なる顔文字を, 人手により表 2 で挙げた 6 つのカテゴリに予め分類し, 訓練データ, テストデータとして用いた. 1075 種の顔文字は, それぞれ「喜んでいる」に 435 種, 「悲しんでいる」に 184 種, 「怒っている」に 71 種, 「驚いている」に 102 種, 「動作を表す」に 152 種, 「苦笑い」に 131 種にそれぞれ分類された. このデータを, Nearest Neighbor 分類器 (NN) と SVM を用いて分類する. 実験は 10 分割交差検定により行った. 各手法による結果を, 表 8 に

表 8: 各手法によって得られた分類結果

カテゴリ	NN				SVM					
	DTAK		SSK		bag-of-words	DTAK			SSK	
	10-KK	NN	10-KK	NN	linear	Poly		Poly		RBF
喜んでいる	76.2	91.0	62.3	89.9	76.9	92.5	91.1	95.7	84.4	94.0
悲しんでいる	69.3	89.0	10.8	87.4	77.3	88.2	82.5	91.7	89.8	92.1
怒っている	48.5	81.4	27.2	77.0	45.6	85.7	72.4	88.2	83.5	87.9
驚いている	39.2	86.3	55.8	77.3	73.0	86.0	74.1	90.6	79.3	85.7
動作を表す	57.3	79.5	25.3	62.3	58.1	71.7	68.4	83.1	57.0	78.3
苦笑い	20.3	80.7	28.3	77.9	49.0	89.5	81.4	92.7	82.0	94.0
合計	64.0	86.6	44.3	82.8	70.1	87.8	82.6	92.1	81.9	90.3

示す. なお, 結果は F 値を用いて表す.

5.2.1 NN による分類

NN による分類では, 最も類似するパターンのみを考慮した分類 (NN) と, 類似度の高い 10 件の多数決による分類 (10-NN) の両方の, DTAK と SSK それぞれを用いた場合の精度について調べた. なお, DTAK においては p, q, r, SSK においては λ の各パラメータによって得られる結果は変化する. 表 8 は, パラメータ p, λ を変化させて, 最も結果が良かった時の値である. その結果, 10-NN と NN の両方で, SSK よりも DTAK を用いた方が, 高い分類精度を示した. また, DTAK と SSK の両方で, 10-NN よりも NN の方が精度は高い事が分かった.

5.2.2 SVM による分類

SVM を, 通常の bag-of-words のみを考慮するカーネルを用いた場合と, DTAK, SSK を用いた場合について調べた. DTAK, SSK においては, NN の時と同様に p, λ を様々に変化させ, 最も結果が良くなる値を用いた. その結果, DTAK と SSK のどちらを用いても, 通常の場合よりも結果は向上した. DTAK と SSK とを比較すると若干 DTAK の方が高かったが, その差はわずかであった.

次に通常の SVM に対して, Polynomial カーネル

$$K(X, X') = (K_n(X, X') + 1)^2$$

を追加すると, 結果は向上した. これは Polynomial カーネルにより, 素性の順序が考慮されたためだと考えられる. 同様に DTAK と SSK に対しても Polynomial カーネルを追加すると, 得られる結果は向上した. 但し SSK においては, Polynomial カーネルよりも RBF カーネル

$$K(X, X') = \exp(-s||K_n(X, X) - 2K_n(X, X') + K_n(X', X')||)$$

を追加した時の方が, さらに結果は良かった. DTAK に RBF カーネルを追加した場合には, 得られる結果は Polynomial カーネルを追加した場合や, 何も追加しなかった場合よりも低かった.

6 まとめ

本研究では顔文字を抽出する手法として, SVM を用いたチャンカーである *Yamcha*[3] を利用した手法を用い, また顔文字の表す意図を理解するためには, 様々な手法を用いて, 顔文字を 6 種類のカテゴリに分類した.

顔文字の抽出実験においては, 再現率で 86.7%, 精度で 85.5%であった. 顔文字の分類実験においては, DTAK と SSK と呼ばれる, 素性の配列を考慮したカーネルを用いた. 結果, 分類精度が向上した. この事から, 顔文字の類似性を判定する目的においては, 文字の並び順を考慮する事が重要である事を確認した. また最も良い手法においては, 92.1%という高い分類精度を得ることができた. この結果, 従来ほとんど有効に扱えていなかった顔文字の持つ情報を, 新たに文章の意味や意図の理解に役立てる土台ができた. また顔文字の持つ情報についても, 表す感情や動作に基づく分類が高い精度で可能である事が確認され, 文章の書き手の意図や感情を顔文字から推測できる可能性ができた.

参考文献

[1] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines - a kernel approach. In *Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 49–54, 2002.

[2] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianiti, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, Vol. 2, pp. 419–444, 2002.

[3] 工藤拓, 松本裕治. Support vector machine を用いた chunk 同定. 自然言語処理, Vol. 9, No. 5, pp. 3–22, 2002.

[4] 浅原正幸, 松本裕治. 形態素解析とチャンキングの組み合わせによる日本語テキスト中の未知語出現箇所同定. *SIGNL-154*, March 2003.

[5] 中村純平, 池田剛, 乾伸雄, 小谷善行. 対話システムにおける顔文字の学習. 第 154 回 自然言語処理研究会, 2003.

[6] 奈良先端科学技術大学院大学 松本研究室. 形態素解析システム 『茶筌』 version 2.2.5 使用説明書, 2001.