平成28年度卒業論文

テンプレートマッチを用いた画像認識による顔文字の検出

情報・通信工学科　コンピュータサイエンスコース

1211072　佐々木透

指導教員　寺田 実　准教授

提出日 2017年 1月31日

概要

目的

顔文字とは文字列に視覚的情報を持たせる手法の一つであり, 自然言語処理では文書の感情判定に用いる研究がある. 顔文字の感情判定には必ず顔文字を検出するステップが存在し, 文字コードによる検索手法が用いられている. しかし, Unicode は毎年文字が新規追加され, 顔文字にも未知の記号が増えていくと文字コードによる検出手法では検出精度が低下する. 本研究では, 顔文字に使われる文字が未知の記号であったとしても人間には顔文字として認識出来ることに着目し, 画像認識を用いた顔文字の検出法を実装した.

方法

本手法では顔文字が含まれる文字列を画像として描画し, 顔文字に出現する頻度の高い記号をテンプレート画像とするテンプレートマッチ法を用いてマッチする座標が連続している箇所を顔文字として検出している.

結論

2010 年に存在した顔文字をもとに作成した記号をテンプレートとし, 当時存在しなかった特殊顔文字に対して本手法を適用したところ, 先行研究が用いた文字コードによる手法よりも再現率を上回ったが, 英単語や記号の連続を誤検出して適合率は下がる傾向にあった. 先行研究の顔文字の感情分類では, 検出した顔文字に出現するパターンから感情値を抽出しているが, パターンをテンプレートとして本手法を用いることにより感情分類へ応用することが可能である.

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

計算機においてテキストとはバイナリ列を文字コードに沿って対応させ人間に理解しやすく表示したものであるが, 近年 Web 上のコミュニケーションサービスの発展に伴いユーザはテキストに視覚的な作用を持たせて文字以上の意味を付加してきた. その代表が顔文字やアスキーアートである. 特に顔文字については自然言語処理の分野ではドキュメントの感情判定に利用されている.[1] [2] 文書中の顔文字の検出については正規表現, もしくは顔文字の構成要素をキーとする部分一致検索が用いられてきたが, 現在主流の Unicode は毎年数千から数万の新規記号が追加されており, それに伴う形で顔文字の種類も年々爆発的に増え続けている.

1.1.1　文字コードによる顔文字検出の問題

文字コードを用いた検出手法では上記のような未知の記号を持つ顔文字を検出する事は原理的に不可能であるとともに, 次のような問題が発生する. 図 1. 1 に示す 2 つの顔文字についてユーザによる視覚的な情報は左側がほぼ一致, 右側は完全一致しているが, 構成する文字コードは異なる顔文字であるため, 自然言語処理による検出手法では両者がともに検出可能である保証はない.
四角枠内は Unicode における CodePoint と文字名である.

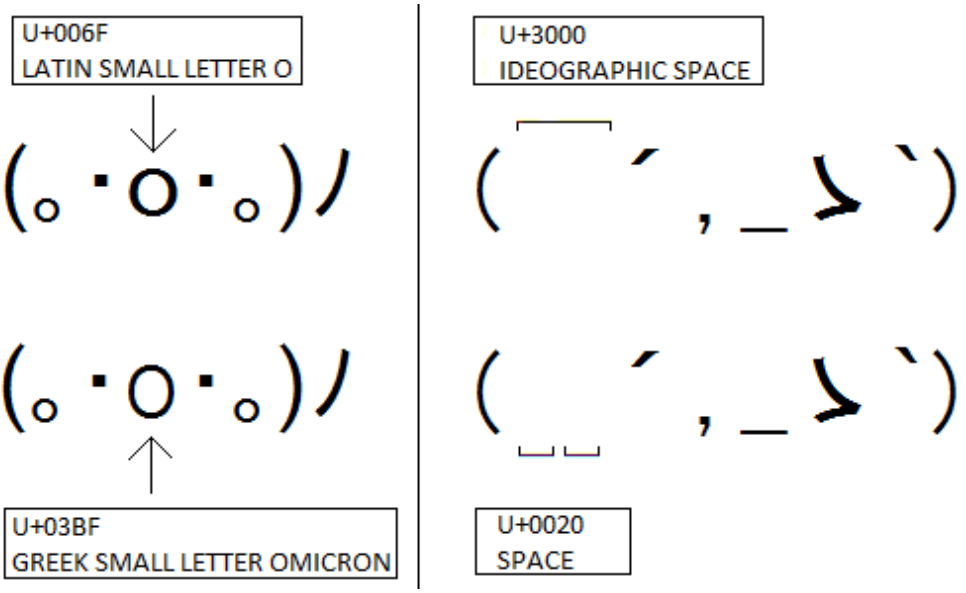

図 1.1: 視覚的には同じ顔文字でも文字コードが異なる例

1.2　着眼点

前項にて記述した文字コードの違いによる問題を解決する方法として, フォントの描画結果を画像として解析する方法が有効であると考えた. また画像による検索手法では次の例のような利点も発生する. 例えば検索キーとして e（U+0065 LATIN SMALL LETTER E) を持っていた場合, 次のような記号群が検索が可能である.

図 1.2: e を部分的に含む文字の表

CodePoint	character	name
U+00E8	è	LATIN SMALL LETTER E WITH GRAVE
U+00E9	é	LATIN SMALL LETTER E WITH ACUTE
U+00EA	ê	LATIN SMALL LETTER E WITH CIRCUMFLEX
U+00EB	ë	LATIN SMALL LETTER E WITH DIAERESIS
U+0113	ē	LATIN SMALL LETTER E WITH MACRON
U+0115	ĕ	LATIN SMALL LETTER E WITH BREVE
U+0117	ė	LATIN SMALL LETTER E WITH DOT ABOVE
U+0119	ę	LATIN SMALL LETTER E WITH OGONEK
U+011B	ě	LATIN SMALL LETTER E WITH CARON

上は一例であるが, Unicode には上記のように一つの文字に対して付加記号がついているという文字が非常に多い. そのためテキストによる検索手法では検出できない文字も画像認識では検出が可能である.

1.3　目的

本研究で提案する顔文字の検出手法は入力をテキストとし, テキスト中に含まれる顔文字部分を検出することを目的とする. なお, 顔文字には西洋顔文字と東洋顔文字があるが, 本研究で対象とするものは東洋顔文字である. 西洋顔文字は英文で用いられており, 顔の向きが行に対して平行に表現されている顔文字を指す. 対して東洋顔文字は顔が垂直に表現されている顔文字を指す. 現在東洋顔文字はほとんど日本語の文章中で使われており, 本研究も日本語の文章から顔文字を検出する事を想定している.

表 1.1: 西洋顔文字と東洋顔文字の例

感情	西洋顔文字	東洋顔文字
喜び	:-)	(@ ^ ▽ ^ @)
怒り	:-(	（｀□´）　(｀３´)
悲しみ	:-/	(;_;)　(x_x；)
驚き	:-O	＼（゜o゜）　/

1.4　論文構成

本章では序論として研究の背景と目的について述べた.
第二章ではテンプレートマッチングのアルゴリズムについての説明を述べる.
第三章では本研究に関連する研究について述べる.
第四章では本研究の提案システムについて述べる.
第五章では本研究の提案システムの実装について述べる.
第六章では評価実験について述べる.
第七章では結論と今後の課題について述べる.

第2章　関連研究

2.1　CAO: A Fully Automatic Emoticon Analysis System[1]

2.1.1　概要

Ptaszynski らは目と口の形状の組み合わせに感情が表れるという理論のもと, 入力文書内から顔文字を自動で検出し, 感情判定を行うシステムである CAO を実装した. CAO では Web 上に存在する顔文字辞書から感情ラベルがついている顔文字を収集し, 10,137 の顔文字から構成されるデータベースを構成して以下のステップで検出した顔文字の感情分類を行っている.

1. 検出した顔文字をデータベース上の顔文字と完全一致検索
2. 1 で検索失敗した場合は目と口の組み合わせのみを検索

2.1.2　本研究との関連

入力文書内の顔文字の検出のアルゴリズムについては, 収集した顔文字に出現する頻度の高い 400 の記号が 3 つ以上含まれる行に対して上の 2 ステップの検索を行っている. しかし, 記号を含む行からどのようなアルゴリズムで検索を行っているかは論文中に明らかにされていないので, 本手法と比較する際は入力に対して頻度の高い 400 の記号が出現した場合を検出したものとして比較する.

論文中では ameba ブログをコーパスとしたテストセットに対して, 顔文字の検出率は 97. 6%と非常に高い検出率を示しているが, しかしこの数値は, 論文発表時の 2010 年当時の顔文字であり, 2017 年現在の顔文字にはこの 400 の記号の中に含まれない記号が非常に多い.

その為, 序論に述べた文字コードによる検索の問題が発生し, 流動的な顔文字に対するシステムを対応させるには新しい顔文字が出る都度データベースに追加していく以外に方法がない.

2.2　ツイートに出現する顔文字等の文字と記号に着目した感情分類 [2]

2.2.1　概要

三好らは Web 上コミュニケーションサービスである Twitter について, ツイートに含まれる顔文字を用いて感情抽出, および顔文字の感情分類を利用して，ツイートを肯定（Positive）と否定（Negative）に分ける PN 分類を実装している. 顔文字辞書等から顔文字を抽出して 5 つの感情体系に分類し, それぞれの感情に分類された顔文字に出現する頻度の高い文字列をパターンとして保持して未知の顔文字の感情分類を行う.

2.2.2　本研究との関連

三好らは Perl Compatible Regular Expression による正規表現によって顔文字を定義し抽出している. 定義は次の通りである.

```
((?!C{3,}).){2,}
```

上記の C は英数字, ひらがな, カタカナ, または漢字の 1 文字を表す. この正規表現は, 次の条件を満たす文字列を半角括弧で囲んだ文字列にマッチする.

- C が 3 文字以上連続しない
- 2 文字以上の文字列

この顔文字の抽出手法は, 未知の文字に対する欠点を正規表現によって解決している反面, 顔の輪郭に相当する括弧を含まないような顔文字は抽出することができないという例を三好らは指摘している.

文字コードの指定がないので, 本論文で比較する際には UTF-8 に従って次のように定義する.

```
[0-9A-Za-zぁ-ヶ一-龠]
```

[一-龠] は日本語に用いられる漢字にマッチする.

表 2.1: 論文中より顔文字として抽出した文字列の例

	顔文字	顔文字でない
1	(-_-)zzz	(14)】
2	(´Д)」	?(ピー)
3	(〃’ ’〃) ☆	…(直喩)
4	＼(＾O＾)／	(英語)

表 2.2: 論文中より抽出できなかった顔文字 6 件

1	＾x＾*
2	ノハ゜ ウ゜）＜
3	`･ω･
4	´ω`
5	＾＾
6	＾＾♪

第3章　提案手法

3.1　概要

本手法は日本語の文章を入力とし, 顔文字に使用される頻度の高い記号が連続している領域は顔文字である可能性が高いとして, 次のようなステップにより顔文字を検出する.

1. 顔文字の収集と記号の画像化
2. 入力テキストを画像化
3. マッチ
4. 座標の出力と顔文字領域の検出

3.2　顔文字の収集と記号の画像化

顔文字の収集については, 顔文字 Cafe[1]をはじめとする, Web 上に存在する顔文字を集約した顔文字サイトに収録されている顔文字を用いる. 収集した顔文字に出現する記号を数え上げ, 出現回数が一定回数を超える記号を画像化し, 後述するテンプレートマッチングに用いる.
ただし本論文の評価実験では先行研究との比較の為, 関連研究に示す CAO が検索に用いた記号を用いて顔文字の検出を行う.

3.3　入力テキストの画像化

顔文字の検索対象となる日本語の文章を前項の記号の画像化と同様の条件で画像化する. この入力画像は後述するテンプレートマッチングに用いる. 以下は次の実際のツイートから顔文字を含む文章を画像化した例である.

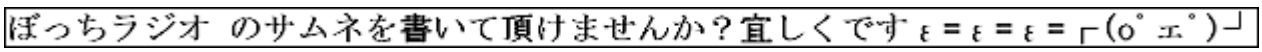

図 3.1: 顔文字を含む入力文字列を画像として描画した例

[1] http://kaomoji-cafe.jp/

3.4　マッチング

3.3 によって画像化された入力文字中に, 3.2 によって画像化された顔文字に含まれる記号が含まれる座標を検出する. 検出には入力文字を被探索画像, 記号をテンプレート画像としたテンプレートマッチング法を用いる.

3.4.1　テンプレートマッチングについて

概要

テンプレートマッチングとは被探索画像, 及び被探索画像中から検出したいテンプレート画像を用意し，被探索画像上でテンプレート画像を走査させることによって類似度が高い座標を出力として返すアルゴリズムである. 類似度が最大の座標一点のみを出力する方法と, 一定の閾値を超える座標を複数出力する方法があるが, 今回は被探索画像の顔文字中に同一の記号が出現する可能性があるので, 後者の類似度が一定の閾値を超える座標を出力として返すものとする.

類似度の計算

各座標における入力画像とテンプレートの類似度は以下の計算式で算出される (OpenCV 公式ドキュメントより).[2]

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y')$$

$I(x,y)$: 被探索画像の座標 (x,y) における輝度値

$T(x',y')$: 走査中のテンプレート画像の輝度値

被探索画像とテンプレート画像はグレイスケール化し, 輝度値は 1 次元で,0 から 255 の値をとる. 本論文ではこの R(x, y) のスコアを次のように正規化し-1 から 1 の範囲を取るようにして閾値を超える点を検出する.

$$R(x,y) = \frac{\sum_{x',y'} T(x',y') \cdot I(x+x', y+y')}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x', y+y')^2}}$$

[2] http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

3.5　座標の出力と顔文字領域の検出

顔文字領域とは, テンプレートにマッチした領域が 3 回以上連続している箇所と定義し, 顔文字領域の始点と終端を出力する. 以下は本手法によって出力された始点と終端をもとに赤枠で囲った例である.

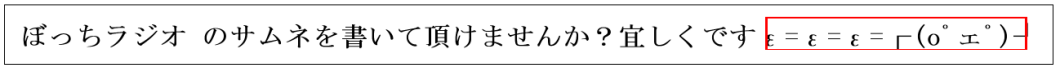

図 3.2: 描画文字列中から顔文字の検出例

第4章　実装

4.1　開発環境

本システムにおける画像の描画は Java Standard Edition 7 の Graphics2D クラスを用いていおり, テンプレートマッチングは Python2. 7 によって実装している.

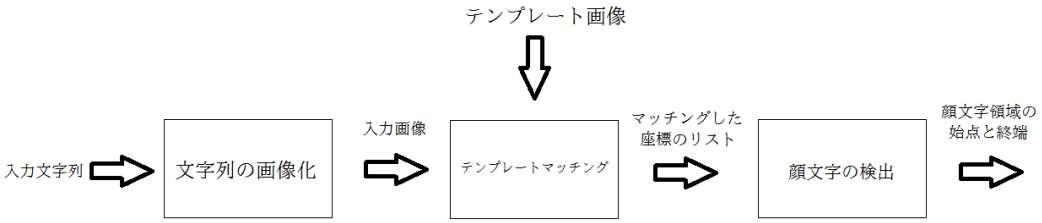

図 4.1: 入力文字列から顔文字の検出フロー

4.2　描画

Java Graphics2D クラスの drawString() メソッドにより描画する. 白い長方形領域にフォントを MSP ゴシック, 大きさ 16pt で描画する. フォントと大きさを上記にした理由は現在普及率が最も高いウェブブラウザである google 社の Chrome の日本語版がデフォルトにしている設定であり, より人間が顔文字を見たときの描画に近いと考えられるためである. また描画の際, アンチエイリアス処理を施すものとする.

4.2.1　入力文字列

入力文字列は縦 19 ピクセルの白色画像に左詰めで 1 行に描画する. 縦幅は描画後の文字の上下にパディングとして 1 ピクセルの白色ドットが入るように描画している.

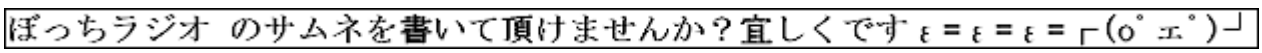

図 4.2: 顔文字を含む入力文字列を画像として描画した例 (2 倍に拡大して表示)

4.2.2　テンプレート画像

テンプレート画像は記号の黒色部分をのまわりに 1 ドットだけパディングピクセルを入れて切り詰めたものとする. 以下はテンプレートに用いられる記号の一部を画像化した例である.

図 4.3: テンプレート画像の例 (4 倍に拡大して表示)

なお, 説明のためにテンプレート画像領域の外側の黒枠で囲っている.

4.3　テンプレートリスト

以下の表は先行研究である CAO が顔文字検出に用いた記号からひらがなや漢字などの, 日本語本文中に出現しやすい記号を除いた 325 の記号であり, 本手法で用いるテンプレートのリストである.

表 4. 3 テンプレート画像のリスト

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152
153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184
185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216
217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232
233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248
249	250	251	252	253	254	255	256
257	258	259	260	261	262	263	264
265	266	267	268	269	270	271	272
273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288
289	290	291	292	293	294	295	296
297	298	299	300	301	302	303	304
305	306	307	308	309	310	311	312
313	314	315	316	317	318	319	320
321	322	323	324				

4.4　テンプレートマッチング

テンプレートマッチングは Intel 社によって開発された画像処理ライブラリの OpenCV2. 4. 13[1] を Python2. 7 から利用しており, 次のメソッドによる返り値が 3. 2 で示したテンプレートマッチングによる類似度となる.

```
cv2. matchTemplate(img_gray, template, cv2. TM_CCOEFF_NORMED)
```

ここで第 1 引数は被探索画像となる入力文字を描画したもの, 第 2 引数はテンプレート画像, 第 3 引数は R(x, y) の計算式を適用する為の定数である. 返り値には R(x,y) が各座標について二次元リスト形式で格納されており, リストのインデックスがそのまま x, y 座標の値になる. 閾値を越える R(x,y) を持つ要素のインデックスを全て配列オブジェクトに格納して後述する 4.4.2 で出力している.

4.4.1　適用例

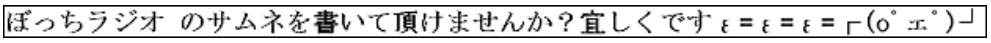

図 4.4: 顔文字を含む入力文字列を画像として描画した例

テンプレートマッチ適用例として, 上記の画像に対しマッチした箇所を赤枠で囲んで示す.

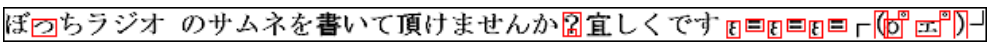

図 4.5: R(x, y) ≧ 0.98 の設定でテンプレートマッチを適用した例

顔文字外にマッチしている記号に”っ”, ”？”があるが, 前者は顔文字の手として, 後者も顔文字の口に出現する頻度が高い記号としてテンプレートに入っているためマッチしている.

[1] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html

4.4.2　座標の出力

マッチした座標は四角形の左上の座標を次の形式で出力している.

`template 番号, x 座標 (pixel), テンプレート画像の幅 (pixel)`

前項の例では次のような出力になる.

```
5,563,6
5,589,6
23,554,11
32,455,8
32,481,8
32,507,8
33,598,8
34,548,8
50,355,5
72,467,11
72,493,11
72,519,11
75,574,15
117,18,15
274,352,11
```

文字列の左側から見て最初にマッチングした記号は, テンプレート番号 117 の”っ”であり以下の出力により, x=18 ピクセルの座標にマッチしたことが分かるようになっている.

4.5　検出アルゴリズム

上の出力を受けて 3.5 で述べた顔文字領域の始点と終端を次の手順に従って判定している.

1. 出力のリストを, x 座標にしたがってソート,
2. (x 座標+テンプレート幅+24 ピクセル) 以内に次のマッチングした座標の有無を判定,
3. 2 が True ならカウントを進めて次の座標に対して 2 を実行,
4. 2 が False かつ, カウントが 3 未満の場合はカウントを初期化して次の座標に対して 2 を実行,
5. 2 が Flase かつカウントが 3 以上の場合は探索を終了し, 最初のカウントした x 座標を始点, と最後にカウントした (x 座標+テンプレート幅) を終点として出力,

マッチングした座標をカウントをしているのは, マッチングした座標が連続しているかどうかを確認であり, 顔文字領域外の文章の記号が単発でマッチングしたとしても, 顔文字として誤検出しないようにする為である. 2. において 24 ピクセルとしているのはフォントサイズが 16 ポイントであり, その 1. 5 文字分以内に次のマッチングした座標を有無を判定している. これにより, 顔文字中にテンプレートに無い未知の記号が含まれていたとしても 2 文字以上未知の顔文字が連続していなければ検出が可能である.

尚, 入力画像の文字中に顔文字が含まれていない場合はステップ 5 に到達せず, その場合は終点を 0 として出力する.

第5章　評価実験

5.1　テンプレートに存在する記号によって構成される顔文字に対する検出率

本手法に用いられているテンプレートはCAOが検出に用いた記号群の部分集合によって構成されており, CAOがこの記号群を作成するにあたって用いられた顔文字が, 本手法によって検出可能でることを確認する.

5.1.1　実験方法

CAOが保持している顔文字は全部で10,131あり, このうち無作為に選んだ1000の顔文字について本手法と正規表現による手法を比較する. なお, CAOは保持している顔文字のデータベースからこれら1000の顔文字を完全一致で必ず検出することが可能なので検出率は100%とする. 本来ならば2010年以前のCAOが用いていない顔文字を収集して比較するべきであるが, 顔文字の発生が2010年以前であると断定できるのはCAOが用いている顔文字のみである. 正規表現による抽出は半角括弧の顔文字にしか抽出できないが, 本実験では全角括弧で囲われている顔文字も抽出できるように拡張する.

5.1.2　実験結果

表 5.1: 2010年の顔文字に対する検出率

検出手法	検出数	検出率
テンプレートマッチング	997/1000	99.7%
正規表現	885/1000	88.5%

5.1.3　考察

検出に用いたテンプレートは 324 と CAO が検出に用いた 400 のうちの 8 割程度であるが, 顔文字の検出には 99%以上の検出率を出している. 一方正規表現は括弧で囲われていない顔文字は全て検出に失敗している. 次に, 本手法によって検出が失敗した顔文字 3 つを, 検出したテンプレートと共に以下に示す.

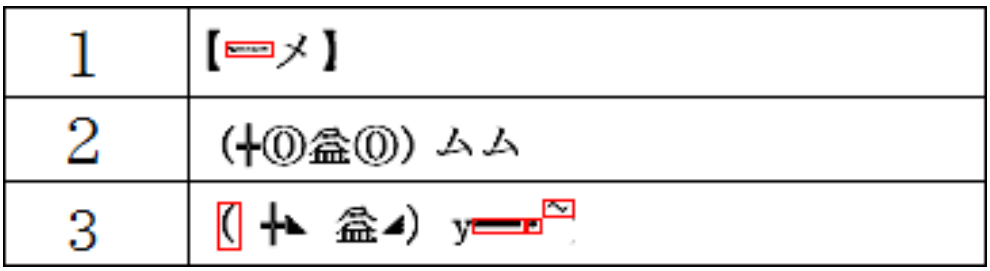

図 5.1: 検出に失敗した顔文字

1 は顔文字の構成要素が極端に少ないために検出が失敗している. 2, 3 は構成要素がほとんどテンプレートに含まれていないため失敗しているが, テンプレートに含まれている括弧も検出に失敗している. 怒りを表す”＋”記号, 及び顔文字の目が括弧のピクセルに接近しているが, テンプレートの画像は記号の描画の上下左右に 1 ドット白いピクセルを入れてあるため, テンプレート画像を走査させたときに隣の記号の一部も類似度計算に含まれてしまい, 類似度が低下したことが原因である.

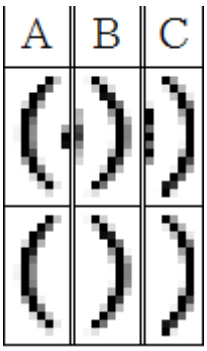

図 5.2: テンプレートに含まれている記号が検出に失敗した括弧
A は 2 の顔文字の左括弧 (上) と対応するテンプレート (下)
B は 2 の顔文字の右括弧 (上) と対応するテンプレート (下)
C は 3 の顔文字の右括弧 (上) と対応するテンプレート (下)

5.2　特殊文字顔文字に対する検出率

2010 年には存在しない顔文字で現在広く使われる顔文字として, 特殊顔文字があげられる. 特殊顔文字の厳密な定義は存在しないが, 顔文字を集約している媒体では共通に特殊顔文字という呼称を用いており, 本論文では環境依存文字を含む顔文字を特殊顔文字と定義する. 今回はこの特殊顔文字に対して本手法, CAO が用いた検出手法, 正規表現による検出手法, の 3 つによって検出率を算出する.

5.2.1　実験方法

データセットには顔文字 Cafe の特殊顔文字一覧[1]に載っているものを用いる.

5.2.2　実験結果

表 5.2: 特殊顔文字に対する検出率

検出手法	検出数	検出率
テンプレートマッチング	418/500	83.6%
CAO	390/500	78.0%
正規表現	402/500	80.2%

[1] http://kaomoji-cafe.jp/author/kaomoji/

5.2.3　考察

本手法, およびテキストによる検索を用いた CAO も 2010 年の顔文字よりも検出率が下がっているが, 本手法が CAO を上回っている. このような結果が得られたのは特殊文字と類似度の高いテンプレートがマッチしたためである. テストケースの内 CAO が検出に失敗しテンプレートマッチングが検出に成功した顔文字の例を以下に示す.

未知の記号に対してテンプレートがマッチした例

図 5.3: 未知の記号に対してテンプレートがマッチした例

上の例で顔文字の目の部分は CodePoint U+275B の HEAVY SINGLE TURNED COMMA QUOTATION MARK ORNAMENT という特殊文字であり,2010 年の顔文字には存在しない記号であるが, 本手法ではテンプレート番号 5 の U+30FB KATAKANA MIDDLE DOT がマッチし, 検出に成功している.

未知の記号の構成要素にテンプレートがマッチした例

図 5.4: 未知の記号の構成要素にテンプレートがマッチした例

顔の構成要素である目と口は合わせて一つの記号であり, CodePoint U+141B の CANADIAN SYLLABICS NASKAPI WAA という特殊文字である. この記号も CAO の顔文字 10,131 の中には一切出現しない記号であるが, 目に該当する部分にテンプレートの番号 50 の U+FF0E FULLWIDTH FULL STOP がマッチしたことにより検出に成功している.

5.2.4　正規表現について

正規表現による検出が 80%の検出に成功し実験 1 の 2010 年の顔文字よりも検出率が上がっているが, これは実験 2 のテストデータに括弧で囲われている顔文字の割合が偶然多かったためであり, 上記のパターンを含んでいれば未知の記号に関わらず検出が可能であるという特徴がある.

5.3　顔文字を含まない日本語文に対する誤検出

正規表現による方法と本手法を顔文字を含まない日本語文に対して行い False positive の出現率を確かめる. CAO はアメーバブログをコーパスとした場合に誤検出は一つもないとあったので 0%とする. 本論文は代わりに Twitter のツイートより日本語のツイートを用いて行う. その際, ハッシュタグ RT 記号, リプライ, そして正解である顔文字を含む行を取り除いたものをデータセットとして用いる.

5.3.1　実験結果

表 5.3: 顔文字を含まない日本語文字列に対する検出率

検出手法	検出数	検出率
テンプレートマッチング	44/500	8.8%
正規表現	17/500	3.4%

5.3.2　考察

本手法が日本語文字列を False positive と検出したものは次の 3 パターンに分けられた.

1. ”！”や ”？”など, 同一記号の連続 23/44
2. テンプレートが偶然密集する 12/44
3. 英単語, 数字 9/44

2 に関しては現状対処の方法がないが, 1 と 3 に関しては描画ステップに渡す入力文字列に正規表現でフィルタリングをかけることによって文字列の描画前に除去が可能であると考えられる.

5.3.3　適合率と再現率について

実験 5.2 の結果と合わせて適合率と再現率を計算する. テンプレートマッチについては前項目で指摘した 1, 2, 3 を含めたものと, 2 以外を除いたもの 2 通りについて算出する.

表 5.4: 適合率と再現率

検出手法	適合率	再現率	F 値
テンプレートマッチング	90.5 (418/462)	83.6 (418/500)	87.2
テンプレートマッチング (1, 3 を除く)	97.2 (418/430)	83.6 (418/500)	90.0
CAO	100 (390/390)	78.0 (390/500)	87.6
正規表現	95.9 (402/419)	81.6 (408/500)	88.2

第6章　まとめと今後

6.1　本手法の利点

画像認識による顔文字の検出は, 見かけ上顔文字として出現頻度の高い記号であれば文字コードの違いに関わらず検出することが可能であるため, 再現率は高くなる. 序論では使われる記号とその組み合わせである顔文字が年々増え続けていることを指摘した. このことは, テキストによる検出手法で新規の顔文字に対応するためには検索に用いるキーを増やすという手間を必要とするが, 画像認識を用いた検索ではそのような負担を軽減することにつながると考えられる.

6.2　本手法の欠点

適合率は文字コードによる検出方法よりも低くなる傾向になる. また実験 5.2 で, 入力文字が隣接する記号と接近しテンプレートとの類似度が下がるケースが僅かに存在したが, 画像認識特有の欠点と言える. また, 本手法はテキストによる検出よりも計算時間を多く要する. 顔文字の研究利用には静的な解析しか用いられていないが, ストリーミングサービスに顔文字の検出を用いる場合があれば, パフォーマンスの観点から本手法を用いるべきか検討しなくてはならない.

6.3　今後

6.3.1　検索アルゴリズムの改良

テンプレート画像のクラスタリング

今回はテンプレートについて, マッチした座標とその連続回数に基づいて顔文字の検出をしているが, 同一の座標に複数のテンプレートが反応して, 回数のカウントが 2 回進む場合がある. 例えば 4.3 にて記載したテンプレートのリストで番号 27 の ”。” と番号 28 の ”゜” など似通っているテンプレートが存在するためである. 改善策として, 似ているテンプレート同士をクラスタリングして一つのテンプレートに共通化する方法が考えられる.

テンプレートの内包関係の解消

あるテンプレートに別のテンプレートが存在するケースがある. 例えば, 番号 275 の ”？” の下部分の点に番号 79 の ”.” がマッチしたケースなど, この場合も回数のカウントが 1 回余分に進むので, このような内包関係を洗い出してカウントを正常に進める改善策が考えられる.

記号の連続, 英数字の除去

5.3.2 で示した通り誇張表現などで記号を連続したり, 日本語文でも英単語や数字が入るケースがある. そのようなケースは, 同一テンプレートが連続して出力されている場合は無視する, 正規表現によって描画前に英単語, 数字列を除去するなどして適合率を高める方法などが考えられる.

6.3.2　顔文字の感情分類への応用

顔文字に関する研究は顔文字の感情分類が主流であり, そして顔文字の感情分類は顔文字中に出現するパターンをもとに行う場合が多い, 例えば三好ら [2] は論文中に連接する記号のパターンが持つ感情値を算出し, 一部の例を以下の表にしている.

表 6.1: [2] より顔文字として抽出した文字列の例

	怒り	喜び	悲しみ	照れくささ	驚き
ｰ ー ｰ	0.0009	0.0055	0.0000	0.0048	0.0007
o ≧∇	0.0000	0.0004	0.0000	0.0000	0.0000
ｰ д ｰ	0.0009	0.0000	0.0000	0.0000	0.0007
@@ ;	0.0000	0.0000	0.0000	0.0000	0.0007
T △ T	0.0000	0.0000	0.0010	0.0000	0.0000

本手法を用いて顔文字の感情分類に応用をする場合, この連接する記号を本手法のテンプレートとして顔文字中から検出することにより, 先行研究で算出された感情値をそのまま適用することが可能である. このような記号に対する感情値の重み付けは CAO でも同様に行われている.

謝辞

本研究は, 電気通信大学情報理工学部情報・通信工学科コンピュータサイエンスコースの寺田研究室において, 寺田実准教授の指導のもとで卒業研究として行われました. 寺田実准教授には卒業研究のアイデアや卒業論文の書き方などの様々なご指導を頂きました. 心より御礼申し上げます. また, 顔文字のデータベースを New BSD Lincense で全て公開してくださった北海道大学の Ptaszynski 氏にも深い感謝を申し上げます.

参考文献

[1] Michal Ptaszynski, “AO: A Fully Automatic Emoticon Analysis System Based on Theory of Kinesics”IEEE Transactions on Affective Computing, vol. 1, no. 1, pp. 46-59, 2010. 1.

[2] 三好 辰明, 太田 学, “ツイートに出現する顔文字等の文字と記号に着目した感情分類”The 5th Forum on Data Engineering and Information Management. 2013. 3.