人工知能班活動報告書

立命館コンピュータクラブ
2016 年度後期プロジェクト活動
山田知葉
音石朋恵
河村和紀
迫佑樹
城野大河
西澤佳祐
福井尚卿
松永樹
水野佑哉
吉川尚吾

2017/02/09

目 次

1　はじめに

文責:山田知葉

昨今は第 3 次人工知能ブームと言われ，人工知能の技術の進歩は著しいものとなっており，自動運転や医療診断など様々な分野での活用を期待され，研究されている．我々人工知能班は幅広く使われている人工知能の技術について学ぶことを目的として活動を行った．

1.1　活動内容

文責:山田知葉

人工知能に関する基礎的な知識を理解するために，谷口忠大著の書籍『イラストで学ぶ人工知能概論』を用いた．人工知能概論は人工知能で動く仮想的なロボットであるホイールダック 2 号がダンジョンを進み，出口にいるスフィンクスを倒すという内容になっている．

我々の班は書籍から学んだ成果としてホイールダック 3 号の作成を目指し，迷路探索，画像認識，自然言語処理の三つの分野に分かれてさらに各分野を深く学習し，それぞれの成果物を作り上げた．

2　迷路探索

文責:西澤佳佑

迷路探索班では，ホイールダック 3 号における迷路探索機能の実装をおこなった．

2.1　実装

文責:西澤佳佑

今回作成したプログラムについて述べていく．ホイールダック 2 号は,「迷路になっているダンジョンを，宝箱に入ったアイテムや財宝を手に入れながら，出口に早くたどり着かなければならない」という仕様となっている．これを参考に仕様を決定した．使用したプログラミング言語は C 言語である．

2.2　目的

文責:西澤佳佑

目的地への移動回数が最小となる経路を見つける．

2.3　仕様

文責:西澤佳佑

まず，縦がM，横がNの大きさのマップがあり，スタート地点とゴール地点が決まっている．マップには壁や行き止まりがあり，迷路となっている．スタートからゴールまで，マップ内を移動しながら探索を行い，最短経路を導き出す．移動する度に，移動した座標の位置をコンソール上に出力する．その後，導き出した最短経路を出力するというプログラムである．

2.4　前提条件

文責:西澤佳佑

このプログラムを構築する上で，このロボットに対する前提条件を設定した．

- スタート地点とゴール地点の座標のみ知っている
  探索をする上で最低限の情報のみ知っている．
- マップデータを書き換えることができる
  探索する際，マップに探索済みかどうかなどの情報を残すことができる．

2.5　結果

文責:福井尚卿

今回使用した6*6の大きさのマップを図1に示す．

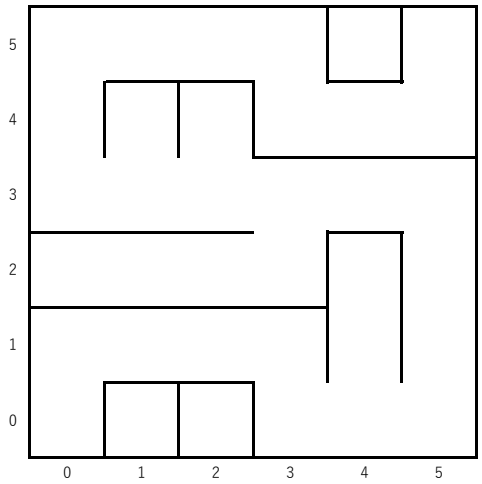

図 1: マップ

スタート地点の座標を x:0,y:0 とし，ゴール地点については 2 種類用意した．1 つ目は x:5,y:5，2 つ目は x:0,y:2 である．それぞれについて述べていく．

- ゴール座標が x: 5,y: 5 のとき

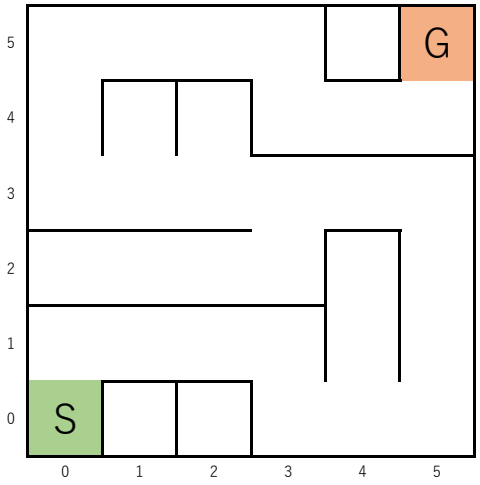

図 2: ゴール座標が x: 5,y: 5 のとき

実行結果を以下に示す．まず，マップデータの読み込み結果を表示する．次に，1 回目の探索の移動経路を表示し，最後に最短経路と移動回数を表示している．また，左側の数字を x 座標，右側の数字を y 座標として移動経路を表示している．
1

ファイル読み込み結果

```
0x01
0x03
0x03
0x00
0x00
0x01
0x02
0x02
0x02
0x03
0x01
0x01
```

[1] S がスタート地点，G がゴール地点を表している．

```
0x02
0x02
0x02
0x01
0x03
0x01
0x00
0x00
0x00
0x02
0x02
0x03
0x01
0x03
0x03
0x00
0x02
0x01
0x02
0x02
0x02
0x03
0x03
0x03
```

一回目移動座標

```
0 1
1 1
2 1
3 1
3 0
4 0
5 0
5 1
5 2
5 3
4 3
3 3
2 3
2 4
2 3
1 3
1 4
1 3
0 3
0 4
0 5
1 5
2 5
3 5
3 4
4 4
5 4
5 5
count: 28 移動回数#
```

二回目移動座標

```
0 1
1 1
```

```
2 1
3 1
3 0
4 0
5 0
5 1
5 2
5 3
4 3
3 3
2 3
1 3
0 3
0 4
0 5
1 5
2 5
3 5
3 4
4 4
5 4
5 5
count: 24 移動回数#
```

結果より，最短経路が導き出されている．

- ゴールが座標 x: 0,y: 2 のとき

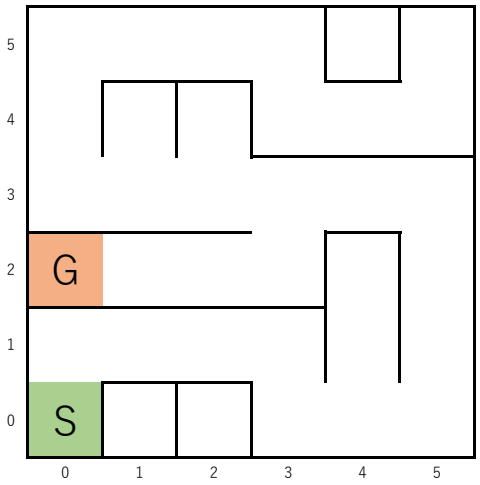

図 3: ゴールが座標 x: 0,y: 2 のとき

実行結果を以下に示す．

```
ファイル読み込み結果
```

```
0x01
0x03
0x03
0x00
0x00
0x01
0x02
0x02
0x02
0x03
0x01
0x01
0x02
0x02
0x02
0x01
0x03
0x01
0x00
0x00
0x00
0x02
0x02
0x03
0x01
0x03
0x03
0x00
0x02
0x01
0x02
0x02
0x02
0x03
0x03
0x03
```

一回目移動座標

```
0 1
1 1
2 1
3 1
3 0
4 0
4 1
4 2
4 1
4 0
5 0
5 1
5 2
5 3
4 3
3 3
2 3
1 3
0 3
0 4
0 5
1 5
```

```
2 5
3 5
3 4
4 4
5 4
5 5
5 4
4 4
3 4
3 5
2 5
1 5
0 5
0 4
0 3
1 3
1 4
1 3
2 3
2 4
2 3
3 3
3 2
2 2
1 2
0 2
count: 48 移動回数#
```

二回目移動座標

```
0 1
1 1
2 1
3 1
3 0
4 0
5 0
5 1
5 2
5 3
4 3
3 3
3 2
2 2
1 2
0 2
count: 16 移動回数#
```

結果より，最短経路が導き出されている．

2.6 実装方法

文責:西澤佳佑

今回実装方法を以下の 2 点に着目して説明する．

- マップ

- 探索アルゴリズム

2.6.1 マップ

マップは unsigned char 型の配列とし，シフト演算やビット演算を用いて計算している．そして，各配列要素のビット列に以下のルールを設定している．

マップデータのビット列 (unsigned char)
00000000(ビット列)
87654321(ビット桁番号)

ビット桁目:説明
8:上方向に移動できるか
7:下方向に移動できるか
6:左方向に移動できるか
5:右方向に移動できるか
4:仮想壁があるかどうか
3:既知かどうか
2:上に壁があるかどうか
1:右に壁があるかどうか

2.6.2 探索アルゴリズム

探索の手順を図 4 のフローチャートに示す.

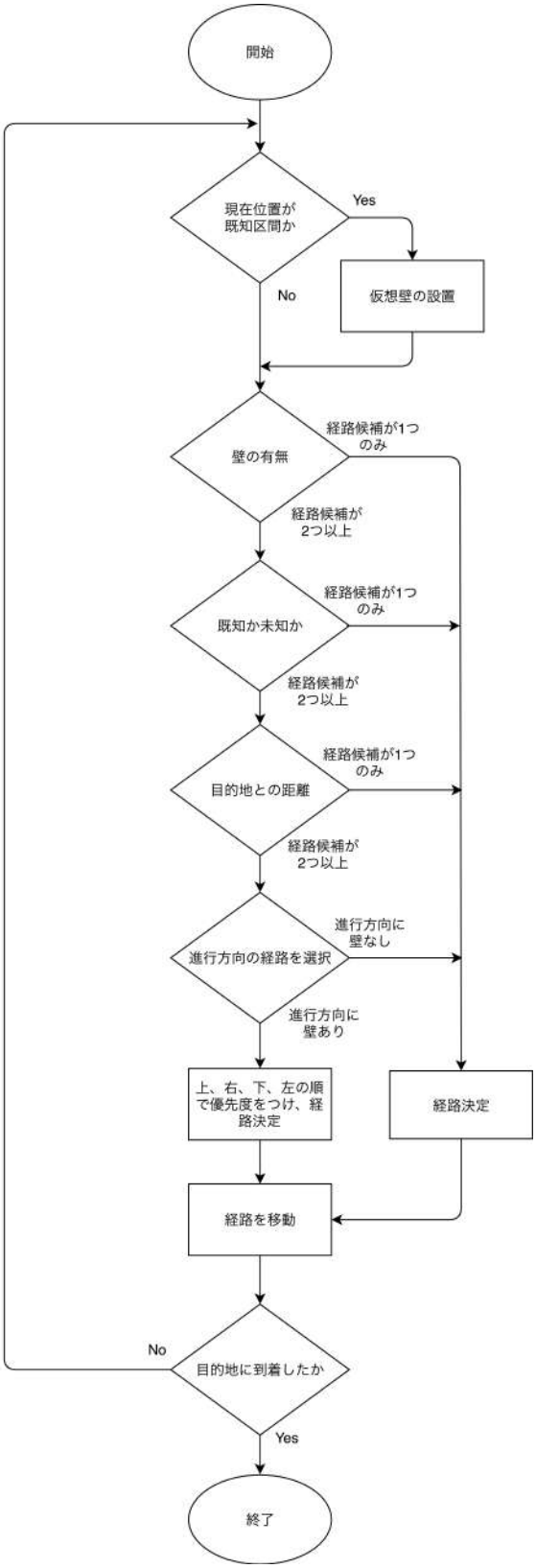
開始
現在位置が
既知区間か
Yes
No
仮想壁の設置
壁の有無
経路候補が1つ
のみ
経路候補が
2つ以上
既知か未知か
経路候補が1つ
のみ
経路候補が
2つ以上
目的地との距離
経路候補が1つ
のみ
経路候補が
2つ以上
進行方向の経路を選択
進行方向に
壁なし
進行方向に
壁あり
上、右、下、左の順
で優先度をつけ、経
路決定
経路決定
経路を移動
目的地に到着したか
No
Yes
終了

図 4: フローチャート

この手順では，現在地点から上下左右の座標を参照し，フローチャートの流れに沿って次に進むべき座標を決定している．次に，具体的な説明をする．

1. 現在位置が既知区間か

文責:福井尚卿

まず初めに，現在位置が既知がどうか，つまり今まで通ったことがあるかという条件から仮想壁を設置するか判定する．次に，仮想壁というシステムについて説明する．探索するにあたって，袋小路のような迷路上においてゴールに辿り着くまでに移動しても利得がないエリアがある．不必要なエリアを仮想壁というシステムを使い立ち入り禁止にすることで袋小路を探索対象から排除している．具体的には，迷路データの左から 5 ビット目のフラグが立っていたら（仮想壁があれば）そのエリアへは立ち入らないという処理をする．このプログラムでは現在探索しているマスが既知の場合，1 つ前に通過したマスに仮想壁のフラグを立てている．

2. 壁の有無

文責:西澤佳佑

壁があるかどうかの判定を行う．壁がない方向を列挙し，通行できるかできないかを迷路データに保存する．この時点で迷路の進行方向が 1 つの場合はその方向に経路を決定する．
また，仮想壁があるかどうかを判定する．仮想壁が発見された場合，その方向のフラグを 0 にする．この時点で迷路の進行方向が 1 つの場合はその方向に経路を決定する．

3. 未知か既知か
   未知の座標を優先して選択する．この時点で迷路の進行方向が 1 つの場合はその方向に経路を決定する．

4. 目的地との距離
   目的地に近い方を優先して選択する．この時点で迷路の進行方向が 1 つの場合はその方向に経路を決定する．

5. 進行方向の座標を選択
   進行方向の座標を優先して選択する．この時点で迷路の進行方向が 1 つの場合はその方向に経路を決定する．

6. 上下左右の順で優先度を付け，経路を決定
   この処理は，進行方向に壁が存在し，さらに未だ候補が 2 つ以上の場合が存在するため実装している．

7. 移動
   決定した座標に移動する処理を行い，その後ゴールかどうか評価をする．

これらの 1 から 7 の処理をゴールにたどり着くまで繰り返すことにより，無駄な経路に仮想壁が立てられ，最短経路が導き出される．

2.7 考察

文責:西澤佳佑

最短経路を導き出し，さらに少ないメモリで高速に動くようなプログラムを実装することができた．しかし，当著書に記載されているアルゴリズムをほとんど使わずに実装しているという問題がある．これは，実装をする時間が充分に取れなかったことと，高速化を目指し，データ量減少を重視してしまったことが原因である．そのアルゴリズムの例として，A*アルゴリズムやベイズ理論などがあり，こちらを使用した方がより柔軟に適応できることが多いと推測される．

3 画像認識

画像認識分野は，全体で勉強会を行いつつ個々で成果物を作成した．

3.1 クラス分類

文責:水野佑哉

3.1.1 目標

画像認識において，画像内の物体を分類ごとに分別することは非常に重要である．今回，OpenCV と Python を使用して，クラス分類について調べた．

3.1.2 開発環境，必要なもの

開発環境は WinPython-2.7.10.3(64bit)，OpenCV-2.4.13 である．クラス分類を行うため 2 種類もしくは多種類の画像を多数用意する．

3.1.3　クラス分類について

まず，クラス分類というものは 2 種類もしくは多種類の画像を種類ごとに分類する物であり，機械学習の一つである．データを分ける技術としてクラスタリングがあるが，こちらはあるデータをプログラムに与え，これを類似点から班分けするものであり，いわゆる教師無学習である．反して，クラス分類は事前に分類を入力したデータを与えておきそれを基に入力されたデータに分類を付与する物であり，いわゆる教師有学習である．

3.1.4　クラス分類を行う

クラス分類を行う際，画像から特徴量を抽出する必要がある。RGB 値をそのまま使用する場合は解像度と同僚の 3 次元ベクトルとなるため学習時間がかかってしまう．そのため次元削減を行う．他には二値化閾値やぼかし，エッジ検出なども有効である．これらの作業で特徴量を抽出した後，用意した画像の一部だけを使用し分類を学習させる．そして、残りの画像でクラス分類を行う．

3.1.5　実践

Python 初心者のため自作するのは困難と考え，インターネット上の情報を模して作成してみたがエラーにより進めず，エラーを理解できなかったことと時間がなかったことから実践は断念した．

3.2　パーセプトロンとニューラルネットワーク

文責:吉川尚吾

3.2.1　得られた知見

DeepLearning の基礎であるパーセプトロンとニューラルネットワークについて勉強した．深層学習の基礎となる

$$y = \begin{cases} 0 & (b1 + w1 * x1 + w2 * x2 \leq 0) \\ 1 & (b1 + w1 * x1 + w2 * x2 > 0) \end{cases}$$

といったパーセプトロンの重みとバイアスをパラメータとして設定する方法を学んだ．これらは AND や OR などの論理回路でも表現できる．しかし単層では XOR は表現できず，2 層必要となる．これらをグラフで表すと単層では線形領域だけだが，多層では非線形が表せられる．

パーセプトロンでは活性化関数として，ステップ関数を使うが，ニューラ

ルネットワークでは，シグモイド関数や ReLu 関数などを利用される．また，出力層で使用する活性化関数は

- 恒等関数
- ソフトマックス関数

が一般的に利用される．恒等関数は回帰問題[2] に，ソフトマックス関数は分類問題[3] に使われる．

入力データをバッチといい，バッチ単位で推論処理を行うことで，計算を高速にできることを知った．ニューラルネットワークの学習アルゴリズムが

1. ミニバッチ（ランダムにデータを選ぶ）
2. 損失関数を減らすための勾配の算出 (偏微分)
3. パラメータの更新（重みパラメータを勾配方向に微小量だけ更新）
4. 1 〜 3 を繰り返す

ということを学んだ．またこれらを確率勾配降下法（stochastic gradient descent）という．もしくはそれらの頭文字をとって SGD という名前の関数で実装される．

3.3 アニメ画像の顔認識

文責:松永樹

3.3.1 目的

アニメのキャラ画像を顔認識する．またキャラクタの判別を行う方法を探る．

3.3.2 動機

顔認識を行うにあたりこれまでの活動で挙げられた OpenCV を用いることにした．またこれらを使い機械学習を施行してみたかった．アニメのキャラ画像にした理由はアニメ画像は識別しやすい（かもしれない）特徴が多々あるためと感じたからである．実際は難しかった．

3.3.3 結論

進捗は出なかった．よって学習したものを挙げることにする．

[2] 予測を行う問題．例:人の写った画像からその人の体重を求める問題
[3] 例:人の写った画像からその人が男性か女性かを見分ける問題

3.3.4 機械学習の概要

まずは機械学習とは何かについて学ぶことにした．機械学習とは人工知能を実現するための手法のことで，例としては顔認識，迷路探索，自然言語処理など様々なものがあげられる．また機械学習には強化学習，教師なし学習，教師あり学習の主に 3 つの学習に分類される．今活動では教師なし学習と教師あり学習の違いについて調べた．

- 教師あり学習
  例えば顔認識を行う際，人工知能は人間の顔の画像を読み込んでも ”これが顔なのだ ”と判別することが難しい．そこで，人工知能に ”目と口と鼻があり，輪郭の付いているものが顔 ”と言った風に教える．人工知能はそれをもとに人間の顔を学習し，未知の画像に対応するといったものになる．教師あり学習でよく用いられるのは OpenCV であり，これには分類分けに使用される学習器が既存で導入されている他，画像認識に役立つモジュールが備わっているためである．中でも Haar-like 特徴分類器が有名であり，これは現実の人間の顔に対し，眼や顔（正面）などに特徴をもたせる事ができる．また，アニメ画像検出のための学習器なども有志によって作られている．

- 教師なし学習
  上述とは異なり，この学習では特徴点を人工知能に与えず，学習させる．例えば，顔認識では人間の顔はこうであるという答えを教えさせず，人工知能にどうにかして何らかの法則があるかを見出してもらうようにする．しかし，人工知能側もそれではどう分けられるのかは変わってくるためある程度示唆をして学習してもらう．また，教師なし学習の中には深層学習（Deeplearning) と言うものがあり，これは学習するごとに重みをつけていき，特徴点を分類させるといったものである．

3.3.5 実験

目的
4.1 節で述べたアニメ画検出の特徴分類器と OpenCV 標準の特徴分類器（実在の人物を主に対応）では検出の差が出るのかを確認する．この実験により，アニメ顔では実在の人物の顔認識と同等にできるのかを見分けられる事ができる．

実験材料
この施行に用いられるものは

- OpenCV 標準の Haar-like 特徴分類器

- haarcascade_frontalface_alt_tree.xml
- OpenCV 対応のアニメ顔特徴分類器
- lbpcascade_animeface.xml
- サンプル画像
  単体画像（一種のキャラが描写されているアイコンサイズの画像）
  集合絵 (複数のキャラが描写されている画像)

  である．この実験では単体画像と集合絵の二つを用いて検出し，二つの特徴分類器の検出精度を比較する．サンプル画像として単体画像は下記に記すサイトからキャラ画像のデータセットを取得し，2 人以上写っているものや見切れ過ぎているもの等を削除した．またこのデータセットは数量が多いので，186 枚に厳選した．また集合絵に用いた画像は ’けいおん !’ のキャラ画像である．Google 検索から無作為に取り出し，同様に厳選した．計 186 枚である．

結果（単体画像）

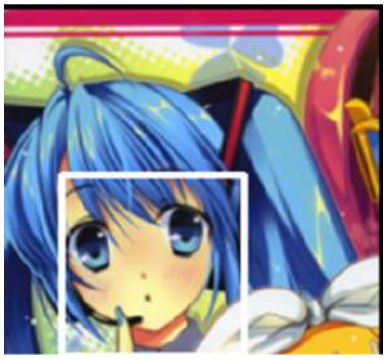

図 5: アニメ顔検出（アニメ顔検出分類器のとき）

アニメ顔検出分類器で分類したところ 186 枚中，誤差（顔でないものに顔判定）が 8 枚，誤差（顔であるはずだが取得できない）が 2 枚検出された．また Haar-like 特徴分類器では顔検出された画像は一枚もないと判断された．

結果（集合絵）

図 6: アニメ顔検出器

図 7: OpenCV 標準の検出器

アニメ顔検出で分類したところ，元画像 186 枚中 181 枚検出され，誤差（顔でないものに顔判定）が 36 枚，誤差（顔であるはずだが取得できない）が 67 枚であった．また Haar-like 検出器では元画像 186 枚中 4 枚検出され，誤差はともに 4 枚全てにあった．以上より，現実の顔の検出とアニメ顔の検出では検出材料としての特徴点の違いが見受けられることがわかった．

考察

なぜ OpenCV 標準の認識とアニメ顔認識ではこれほどまでに認識で異なるのか．いくつか上げてみると

- OpenCV 標準の分類器は正面対応より斜めに映るものに弱い．
- 人間の顔とは違い，輪郭がきっちりしていないものが多い
- アニメと現実の顔ではそもそも特徴点が違いすぎる（顔，髪型，目，髪色など）

と言った風に両者の違いが存在するためだと考える．また，アニメ絵でのキャラの見分け方は様々であるが，誰しも経験から，髪型や髪色，輪郭の形（同一アニメでは見分けがつきにくい），目の色，体型と言ったものがあげられる．よってアニメ絵の顔検出にはこれらの特徴点を踏まえた学習器を使って教師あり学習を行う必要が有るだろう．

3.3.6 展望

今活動を通して人工知能の学習についての知見が深まった．この知見を活かし，次のことを行えるように努力しようと思う．それは”あるキャラクタの画像取得”である．キャラクタの顔認識は上述の通り先人の技術により，行えることがわかった．ならば，キャラクタの顔認識の情報を用いて，キャラクタの判別はできるのだろうか．例えば，顔認識により髪型や特徴をおさえ，教師あり学習で学ばせる事ができれば，複数キャラの集合絵にキャラ判別を行うことができるだろう．また，ディープラーニングでも同様にできることが参考文献にあるサイトにより可能であると記されている．これをやってみたいと考える．そのためにさらに教師あり学習の分類器，また教師なしの深層学習における学習器についてより良い理解を深めていこうと思う．

3.3.7 付録

5 節で述べた実験のソースコード

```
import glob
import cv2
import random, string

# サンプル顔認識特徴量ファイル（アニメ顔分類器，Haar-特徴分類器）like
cascade_path = "lbpcascade_animeface.xml"
#cascade_path = "haarcascade_frontalface_alt_tree.xml"フォルダ内にある画像全てを参照し終えるまでループを回す

#
for folder in glob.glob
('/home/mattu/opencv-3.0.0/build/k-on/*.jpg'):
    color = (255, 255, 255)  パス指定

#
    filePath=folder
    print("PATH:"+filePath)  画像の読み取り

#
    image=cv2.imread(folder)  グレースケール化（情報量を減らす）

#
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  分類器を作成

#
    cascade = cv2.CascadeClassifier(cascade_path)  顔認識の実行

#
    facerect = cascade.detectMultiScale(gray, scaleFactor=1.1,
 minNeighbors=1, minSize=(1, 1))  顔認識が出来れば四角の枠で囲う，できなければ”

#no ”と表示するface
    if len(facerect) > 0:
      for rect in facerect:
        cv2.rectangle(image, tuple(rect[0:2]),
```

```
tuple(rect[0:2]+rect[2:4]), color, thickness=2)画像を指定ファイルに書き込み
      #
      fname = "/home/mattu/opencv-3.0.0/build/img_mix_haar/"
      fname += ''.join([random.choice(string.ascii_letters
+ string.digits)
 for i in range(7)])
      outputPath=fname+".jpg"
      cv2.imwrite(outputPath,image)
    else:
      print("no face")
```

3.4 顔認識

文責:河村和紀

3.4.1 目的

OpenCV に用意されている haar-like 特徴を用いるカスケード分類器の精度を比較し，顔認識においてある条件下において最も精度の良い分類器を選択する．

3.4.2 理論

OpenCV は haar-like 特徴を用いる分類器を提供している．その中には例えば顔，目，笑顔検出のための検出器などがある．これらは opencv/data/haar-cascades/ フォルダ内に保存されている XML ファイルに保存されている．

HaarDetectObjects($image, scaleFactor = 1.1, minNeighbors = 2, minSize = (0,0)$)[4]

- imge：検出対象の画像
- scaleFactor：スキャン毎にスケーリングされる探索窓のスケーリングファクタ
- minNeighbors：オブジェクトを表す矩形を構成する近傍矩形の最小数
- minSize：探索窓の最小サイズ

[4] より詳細は http://opencv.jp/opencv-2svn/py/objdetect_cascade_classification.html?highlight=cascade

3.4.3 実験 1

OpenCV に実装されている Haar-like 特徴分類器のうち顔の正面を顔認識する分類器の 1 人のみが写った画像に対する認識精度を調べる．

#方法

1. 図 8 のように 1 人のみが画像に写っている画像 [5] を 400 枚用意する
2. 図 9 のように Haar-like 特徴分類器 4 種類 (default，alt，alt2，alt_tree) を用いそれぞれ顔認識する
3. 図 10 のように顔認識を誤検知したものの数を計上する

※ Haar-like 特徴分類器 4 種類のファイル名は以下のようである．

- haarcascade_frontalface_default.xml
- haarcascade_frontalface_alt.xml
- haarcascade_frontalface_alt2.xml
- haarcascade_frontalface_alt_tree.xml

以下上から順に default，alt，alt2，alt_tree とする．

図 8: サンプル画像の例

[5] https://www.vision.caltech.edu/Image_Datasets/Caltech101/

図 9: サンプル画像の例

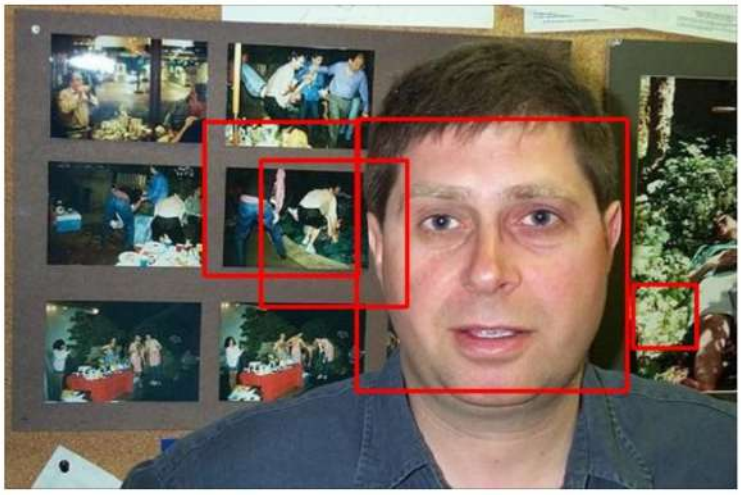

図 10: サンプル画像の例

#結果

誤検知には 2 種類あり，本来顔であるものを顔であると判別できなかったもの (誤検知 1 とする)，顔でないものを顔であると判別したもの (誤検知 2 とする) がある．それぞれについて表 1 に示す．

表 1:

	default	alt	alt2	alt_tree
誤検知 1	2	0	1	0
誤検知 2	260	60	154	0

#考察
alt_tree に関しては 400 枚の内誤検知は 0 箇所であり，続いて alt，alt2，default の順に精度がよいことが分かる．よって基本的に 1 人のみを認識すればよい場面では alt_tree を用いるのがよい．

3.4.4 実験 2

OpenCV に実装されている Haar-like 特徴分類器のうち顔の正面を顔認識する分類器の複数人が写った画像に対する認識精度を調べる．

#方法実験 1 と同様に今度は図 11 のように 1 枚の画像に複数人写ってる画像について haar-like 特徴を用いるカスケード分類器の精度を比較する．1 枚当たり平均 5.4 人が写っている 80 枚のサンプル画像を用いる．

図 11: サンプル画像の例

#結果
実験 1 と同様に誤検知した数を表 2 に示す．

表 2:

	default	alt	alt2	alt_tree
誤検知 1	224	106	108	394
誤検知 2	185	175	175	0

#考察
1 人のみを認識している場合と違い alt_tree は顔を認識できる精度が最低値をしめしてる．複数人の顔認識に関しては alt,alt2 は同程度の精度を示しており，それらの精度は default より上回っているため複数人を認識するような場面では alt，alt2 を用いるのが適切であると考えられる．

4　自然言語処理

4.1　目的

文責:迫佑樹

昨今，SNS 及びブログサービスの普及，発展により多くの人が様々な情報を発信できるようになった．そんな中,「炎上」と呼ばれる事象が多発している．炎上とは，インターネット上での発言に対して非難や中傷が続くことである．炎上は，発信した情報の内容によって起こるものであり，炎上しやすい記事，発言の内容を解析することにより，炎上の特徴を捉えることで，事前に炎上が防げるのではないかと考えた．　そこで今回，私たちは自然言語処理と呼ばれる手法を用いて，炎上する記事内容の特徴を捉え，投稿を行う記事内容と過去の炎上記事の内容を比較することで記事が炎上する確率を教えてくれる Web アプリケーションを作成することとした．

4.2　炎上記事の分類について

文責:迫佑樹

まず，教師データとなる炎上記事の分類について説明する．株式会社はてなが運営するサービス，はてなブックマークを使って炎上記事を集めた．はてなブックマークでは，様々な Web ページへコメントを残す機能がある．そのはてなブックマークにつけられたコメントをネガティブ・ポジティブ判断器 (http://qiita.com/moroku0519/items/e6352d31311418f38227) にかけ，付いているコメントがネガティブだと判断された Web ページは炎上記事，付いているコメントがポジティブだと判断された Web ページは非炎上記事として分類した．そして，その記事本文を取得し，炎上記事及び非炎上記事の内容をテキストデータとして保存した．

4.3　炎上記事の特徴解析について

文責:迫佑樹

炎上記事の特徴を解析するため，テキストデータとして保存した炎上記事の本文を形態素解析した．形態素解析とは，文章を意味のある単語に区切り，辞書を利用して品詞や内容を区別することである．形態素解析で得た単語群のうち，最も文章の特徴を示す名詞や形容詞を使用して後述するテキスト分類を行った．

4.4　ナイーブベイズを用いたテキスト分類について

文責:音石朋恵

テキスト分類とは，与えられた文書をあらかじめ与えられたいくつかのカテゴリに自動分類することである．今回の場合は炎上する記事であるか否かを分類する．まず最初に人間が分類器を訓練し，炎上した記事はどのようなものなのかを教える (教師あり学習)．その訓練データをもとに，分類器は炎上するような文書の特徴を自動学習していく．そして，実際にテキスト分類する際に，一般によく使われており，かつ高速なナイーブベイズを用いる．ナイーブベイズでは先程の学習データをもとに，文書 doc が与えられたときカテゴリ cat である事後確率 P(cat—doc) を求めることでテキスト分類を行う．

4.5　作成した Web アプリケーション

文責:迫佑樹

以上の流れで，炎上を判定する事が可能となった．Web アプリケーションフレームワーク Ruby on Rails を使用し，誰でもテキストを入力するだけで簡単に炎上判断をすることが出来る Web アプリケーションを作成した．作成した Web アプリケーションを以下に示す．

図 12: 炎上判定

4.6　反省，展望

文責:迫佑樹

今回，炎上する確率を 0%から 100%の間で算出していたが，極端に低いか極端に高いかのどちらかになってしまった．炎上するかしないかは，同じ内容でも口調により決まる側面もある．今回の炎上記事特徴付けでは，名詞や形容詞のみを評価基準として使用したため，正しい炎上判断ができなかったと考えられる．

5　おわりに

文責:山田知葉

人工知能班は，人工知能を作るための知識を獲得することを活動目的とし，得た知識を反映させた成果物の作成に取り組んだ．　各分野で成果物を作成することができたが，使用した書籍は大学の講義や人工知能の入門書として使われているものであったため，実際に人工知能を作る場面では勉強に使った書籍では深く触れられていない知識を要した．そのため開発に時間を割くべきであったかもしれない．

参考文献

[1] 谷口忠大『イラストで学ぶ人工知能概論』講談社 (講談社 2014 年 9 月 25 日)

[2] 森川幸人『マッチ箱の脳 (AI) 使える人工知能のお話』(新紀元社 2000 年 12 月)

[3] 『人工知能 Advent Calendar 2015』
http://qiita.com/advent-calendar/2015/ai
（最終閲覧日:2017 月 2 月 7 日）

[4] 『python を使って簡単な画像分類を実現する stMind』
http://stmind.hatenablog.com/entry/2014/01/15/012418
（最終閲覧日:2017 月 2 月 7 日）

[5] 『画像処理入門講座：OpenCV と Python で始める画像処理 — プログラミング — POSTD』http://postd.cc/image-processing-101/
（最終閲覧日:2017 月 2 月 7 日）

[6] 『クラス分類とクラスタリングの違い:コンテクストクリエーションによるプラットフォームビジネス:オルタナティブ・ブログ』
http://blogs.itmedia.co.jp/takafumi/2015/09/post_3.html
（最終閲覧日:2017 月 2 月 7 日）

[7] 斎藤康毅『ゼロから作る Deep Learning -Python で学ぶディープラーニングの理論と実装』（株式会社オライリー・ジャパン 2016 年 12 月 22 日）

[8] 『OpenCV でアニメ顔検出』
http://opencv.blog.jp/python/anime_face_detect
（最終閲覧日:2017 月 2 月 8 日）

[9] 『ご注文は Deep Learning ですか？』
http://kivantium.hateblo.jp/entry/2015/02/20/214909
（最終閲覧日:2017 月 2 月 8 日）

[10] 『Deep Learning でラブライブ！キャラを識別する』
http://christina.hatenablog.com/entry/2015/01/23/212541（最終閲覧日:2017 月 2 月 8 日）

[11] 『OpenCV によるアニメ顔検出の分類器』
http://ultraist.hatenablog.com/entry/20110718/1310965532
（最終閲覧日:2017 月 2 月 8 日）

[12] 『キャラ画像データセット』
http://www.nurs.or.jp/ñagadomi/animeface-character-dataset/
（最終閲覧日:2017 月 2 月 8 日）

[13] 『深層学習でアニメ顔を分類する』
http://qiita.com/hogefugabar/items/312707a09d29632e7288
（最終閲覧日:2017 月 2 月 8 日）

[14] 岡谷貴之『深層学習』 (講談社 2015 年 4 年 8 日)

[15] 『人工知能に関する断創録 ナイーブベイズを用いたテキスト分類』
http://aidiary.hatenablog.com/entry/20100613/1276389337
（最終閲覧日:2017 月 2 月 7 日）