# DP マッチングレポート

平成 28 年 6 月 19 日
未来ロボティクス学科
B3 1426015
今井 良佑

## 1 目的

今回のレポートでは DP マッチングのアルゴリズムについて理解，考察し実際の音響特徴量を用いて認識実験を行う．

## 2 理論

DP マッチングとは長さの異なる 2 つの系列の類似の度合いを知るための手法である．具体的には系列の要素同士の距離を計算したマトリックスを算出．このマトリックスから単語間距離を求める．各点のコストを以下の式 (1)〜(3) の最小値と定義し算出する．

$$cost[i, j] = cost[i-1, j] + mat[i, j] \quad (1)$$

$$cost[i, j] = cost[i-1, j-1] + 2 * mat[i, j] \quad (2)$$

$$cost[i, j] = cost[i, j-1] + mat[i, j] \quad (3)$$

$cost[i, j]$ は (i, j) におけるコスト，$mat[i, j]$ は (i, j) のマトリックスの値．以上で算出されたコストをそれぞれの系列の長さの和で正規化したものを単語間の距離とする．

## 3 実験内容

DP マッチングのプログラムを作成し，テンプレートの 100 単語すべてについて単語間距離を計算し，その中で最小の距離を与えた単語を正解とした時の認識率について考察する．

斜めの遷移の場合は局所距離を 2 倍から $\sqrt{2}$ 倍，または 1 倍に変えて実験し結果を考察する．

## 4 環境・使用機器

プログラムは C と Python で書く．理由としてはグラフとして可視化するうえで Python は慣れているためであり，C は実行の速度を考慮してである．
ソースコードは以下の通り

ソースコード 1: DP_Matching.py

```python
#coding:utf-8
#-----------------------------------------------------------------------------
# Name:  DP_Matching.py
# Author:  R.Imai
# Created: 2016 / 05 / 20
# Last Date: 2016 / 05 / 20
# Note:
#-----------------------------------------------------------------------------
import numpy as np
import sys
import os
import matplotlib.pyplot as plt
import argparse
import csv
import glob
import math


slantCoe = 1 #math.sqrt(2) #斜めの倍率

def parse_arguments():
    parser = argparse.ArgumentParser()
```

```
    parser.add_argument("dirName", help = "Directory there is a feature data")
    parser.add_argument(
           "-p", "--plot",
           type = str,
           dest = "plot",
           default = None,
           nargs = 4,
           help = "plot result graph[teach, label, check, label]"
    )
    parser.add_argument(
           "-s", "--save",
           type = str,
           dest = "save",
           default = None,
           help = "save result graph name"
    )
    parser.add_argument(
           "-t", "--test",
           type = int,
           dest = "test",
           nargs = 2,
           default = [0, 1],
           help = "set check user point [teach, check]"
    )


    return parser.parse_args()

def getDirList(dirName):
    fileList = []
    dirList = []
    for elem in glob.glob(dirName +"/*"):
        fileList.append(glob.glob(elem + "/*"))

    return fileList

def importFile(path):
    try:
        fp = open(path, 'r')
        reader = csv.reader(fp,delimiter=' ')
    except IOError:
        print (path + "cannot be opened.")
        exit()
    except Exception as e:
        print('type'+ str(type(e)))
        print(path)
        exit()
    data = []
    for i,elem in enumerate(reader):
        if i == 1:
            label = elem[0]
        elif i >= 3:
            data.append([float(i) for i in elem[:-1]])
    return data,label


def mkMatrix(teach, test):
    matrix = [[0 for i in range(len(teach))] for j in range(len(test))]
    for j, testData in enumerate(test):
        for i, teachData in enumerate(teach):
            matrix[j][i] = np.linalg.norm(np.array(testData) - np.array(teachData))
    #print(matrix)
    return matrix

def calcMinRoute(matrix):
    row = len(matrix[0])
    col = len(matrix)
    routeMatrix = [[0 for i in range(row)] for j in range(col)]
    routeMatrix[0][0] = matrix[0][0]
    for i in range(1, row):
        routeMatrix[0][i] = routeMatrix[0][i - 1] + matrix[0][i]
    for j in range(1, col):
        routeMatrix[j][0] = routeMatrix[j - 1][0] + matrix[j][0]

    for j in range(1, col):
        for i in range(1, row):
            calcSpace = []
            calcSpace.append(routeMatrix[j - 1][i] + matrix[j][i])
            calcSpace.append(routeMatrix[j - 1][i - 1] + slantCoe*matrix[j][i])
```

```
            calcSpace.append(routeMatrix[j][i - 1] + matrix[j][i])
            routeMatrix[j][i] = min(calcSpace)

    return routeMatrix[col - 1][row - 1]/(col + row), routeMatrix


def DP_matching(teachData, testData):
    num = []
    minNum = 99999999
    for checkData in teachData:
        mat = mkMatrix(checkData, testData)
        minRoute, routeMatrix = calcMinRoute(mat)
        if minRoute < minNum:
            minMat = mat
            minRouteMat = routeMatrix
            minNum = minRoute
        num.append(minRoute)
    #print(np.array(num))
    return np.argmin(np.array(num)), minMat, minRouteMat

def plotMat(mat, locus, save):
    x = np.arange(len(mat[0])+1)
    y = np.arange(len(mat)+1)
    X, Y = np.meshgrid(x, -y)
    plt.pcolor(X, Y, np.array(mat), cmap=plt.cm.binary)
    plt.colorbar()
    plt.plot(locus[0],locus[1])
    plt.xlim(0, len(mat[0]))
    plt.ylim(-len(mat),0)
    if save is None:
        plt.show()
    else:
        plt.savefig(save + ".png")

def plotRoute(mat):
    y = len(mat) - 1
    x = len(mat[0]) - 1
    locusX = [x + 0.5]
    locusY = [-y - 0.5]
    locus = []
    while x != 0 or y != 0:
        if x == 0:
            y -= 1
        elif y == 0:
            x -= 1
        else:
            #print(str(x) + ", " + str(y))
            num = np.argmin(np.array([mat[y - 1][x], mat[y - 1][x - 1], mat[y][x - 1]]))
            if num == 0:
                y -= 1
            elif num == 1:
                x -= 1
                y -= 1
            elif num == 2:
                x -= 1
        locusX.append(x + 0.5)
        locusY.append(-y - 0.5)
    locus.append(locusX)
    locus.append(locusY)
    """
    plt.plot(locusX,locusY)
    plt.xlim(0, len(mat[0]) - 1)
    plt.ylim(-len(mat) + 1)
    plt.show()
    """
    return locus


if __name__ == '__main__':
    arg = parse_arguments()
    fileList = getDirList(arg.dirName)
    feature = []
    labelList = []
    for dataList in fileList:
        elemList = []
        labelElem = []
        for fileName in dataList:
            data, label = importFile(fileName)
            elemList.append(data)
```

```
            labelElem.append(label)
        feature.append(elemList)
        labelList.append(labelElem)

    if not(arg.plot is None):
        for label, data in zip(labelList[int(arg.plot[0])], feature[int(arg.plot[0])]):
            if label == arg.plot[1]:
                check = data
                break
        for label, data in zip(labelList[int(arg.plot[2])], feature[int(arg.plot[2])]):
            if label == arg.plot[3]:
                teach = data
                break
        mat = mkMatrix(teach, check)
        minRoute, routeMatrix = calcMinRoute(mat)
        print(minRoute)
        plotMat(mat, plotRoute(routeMatrix), arg.save)


    else:
        cnt = 0
        for i,elem in enumerate(feature[arg.test[1]]):
            n, mat, Rmat = DP_matching(feature[arg.test[0]], elem)
            if i == n:
                cnt += 1
            else:
                print(labelList[arg.test[0]][n] + "::" + labelList[arg.test[1]][i])
            plotMat(mat, plotRoute(Rmat))
            #plotRoute(mat)
            if i % 10 == 0:
                print (str(i) + "%")
        print(cnt)
```

ソースコード 2: DP_Matching.c

```
/*------------------------------------------------*/
// Name:  DP_Matching.c
// Author:  R.Imai
// Created: 2016 / 06 / 09
// Last Date: 2016 / 06 / 09
// Note:
/*------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <dirent.h>
#include <string.h>

#define slantCoe 2

struct melCepst{
    char label[20];
    int length;
    float data[150][15];
};
struct people{
    struct melCepst mel[100];
};

struct melCepst import(char *path, int num){
    FILE *fp;
    int j;
    char ff[20];
    char fname[30];
    struct melCepst peo;
    sprintf(fname,"data/%s/%s_%03d.txt",path,path,num);
    fp = fopen(fname, "r");
    fscanf(fp,"%s",ff);
    fscanf(fp, "%s", peo.label);
    fscanf(fp, "%d", &peo.length);
    j = 0;
    while (fscanf(fp, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f", &peo.data[j][0],&peo.data[j][1],&peo.
        data[j][2],&peo.data[j][3],&peo.data[j][4],&peo.data[j][5],&peo.data[j][6],&peo.data[j][7],&peo.data[
        j][8],&peo.data[j][9],&peo.data[j][10],&peo.data[j][11],&peo.data[j][12],&peo.data[j][13],&peo.data[j
        ][14]) != EOF){
        j += 1;
    }
    fclose(fp);

```

```
41
42      return peo;
43  }
44
45  float dist(float a[15], float b[15]){
46      float sum = 0;
47      int i;
48      for(i = 0; i < 15; i++){
49          sum += (a[i] - b[i])*(a[i] - b[i]);
50      }
51      sum = sqrtf(sum);
52      return sum;
53  }
54
55  float DP_length(struct melCepst data1, struct melCepst data2){
56      float mat[150][150];
57      float cost[150][150];
58      float distance;
59      int i, j;
60      float calcSpace[3];
61      for(i = 0; i < data1.length; i++){
62          for(j = 0; j < data2.length; j++){
63              mat[i][j] = dist(data1.data[i], data2.data[j]);
64          }
65      }
66      cost[0][0] = mat[0][0];
67      for(i = 1; i < data1.length; i++){
68          cost[i][0] = cost[i - 1][0] + mat[i][0];
69      }
70      for(j = 1; j < data2.length; j++){
71          cost[0][j] = cost[0][j - 1] + mat[0][j];
72      }
73      for(i = 1; i < data1.length; i++){
74          for(j = 1; j < data2.length; j++){
75              calcSpace[0] = cost[i - 1][j] + mat[i][j];
76              calcSpace[1] = cost[i - 1][j - 1] + slantCoe*mat[i][j];
77              calcSpace[2] = cost[i][j - 1] + mat[i][j];
78              cost[i][j] = fminf(calcSpace[0], fminf(calcSpace[1], calcSpace[2]));
79          }
80      }
81      distance = cost[data1.length - 1][data2.length - 1]/(float)(data1.length + data2.length);
82      return distance;
83  }
84
85
86  int main(int argc, char *argv[]){
87      struct melCepst data[2];
88      float minDist, dist;
89      char dirName[4][8];
90      char labelA[20], labelB[20];
91      int i, j, k, l, match;
92      int matchMat[4][4];
93      float trueLen = 0.0, falseLen = 0.0;
94      int trueCnt = 0, falseCnt = 0;
95      strcpy(dirName[0], "city011");
96      strcpy(dirName[1], "city012");
97      strcpy(dirName[2], "city021");
98      strcpy(dirName[3], "city022");
99
100     for(k = 0; k < 4; k++){
101         for(l = 0; l < 4; l++){
102             //printf("%s vs %s\n",dirName[k],dirName[l]);
103             //printf("%s で%s を認識\\\\\n",dirName[k],dirName[l]);
104             //printf("\\begin{itemize}\n\\setlength{\\parskip}{0cm}\n\\setlength{\\itemsep}{0cm}\n");
105             match = 0;
106             for(i = 1;i < 101; i++){
107                 minDist = 99999999.9;
108                 for(j = 1;j < 101; j++){
109                     //printf("%d\n",j);
110                     data[0] = import(dirName[k], j);
111                     data[1] = import(dirName[l], i);
112                     dist = DP_length(data[1], data[0]);
113                     if(dist < minDist){
114                         minDist = dist;
115                         strcpy(labelA, data[0].label);
116                         strcpy(labelB, data[1].label);
117                     }
118                 }
119                 if(strcmp(labelA, labelB) == 0){
```

```
                    match += 1;
                    trueLen += dist;
                    trueCnt += 1;
                }else{
                    falseLen += dist;
                    falseCnt += 1;
                    //printf("%s :: %s\n",labelA,labelB);
                }
            }
            matchMat[l][k] = match;
            //printf("\\end{itemize}\n");
            //printf("\t\tmatting rate is %d\n", match);
        }
    }
    printf("\t|%3d|%3d|%3d|%3d|\n", matchMat[0][0],matchMat[0][1],matchMat[0][2],matchMat[0][3]);
    printf("\t|%3d|%3d|%3d|%3d|\n", matchMat[1][0],matchMat[1][1],matchMat[1][2],matchMat[1][3]);
    printf("\t|%3d|%3d|%3d|%3d|\n", matchMat[2][0],matchMat[2][1],matchMat[2][2],matchMat[2][3]);
    printf("\t|%3d|%3d|%3d|%3d|\n", matchMat[3][0],matchMat[3][1],matchMat[3][2],matchMat[3][3]);
    printf("true dist mean: %f",trueLen/trueCnt);
    printf("false dist mean: %f",falseLen/falseCnt);


}
```

## 5 結果

### 5.1 斜め倍率 : 2

斜め倍率を 2 とした時の認識率は以下の表 1 のようになった.

表 1: 斜め倍率 2

| | | 教師データ | | | |
|---|---|---|---|---|---|
| | | city_011 | city_012 | city_021 | city_022 |
| テストデータ | city_011 | 100 | 99 | 90 | 84 |
| | city_012 | 100 | 100 | 92 | 86 |
| | city_021 | 83 | 91 | 100 | 99 |
| | city_022 | 86 | 94 | 100 | 100 |

また，誤認識した組み合わせを以下に示す．左が誤認識結果，右が実際の答えである．city011 で city011 を認識

なし

city011 で city012 を認識

なし

city011 で city021 を認識

- MARUNOUCHI :: KUNITACHI
- TENRYUU :: SENJU
- ZOOSHIGAYA :: SOSHIGAYA
- DOOTONBORI :: ROPPONGI
- YOTSUYA :: WASEDA
- SHINAGAWA :: GINZA
- TOOKYOO :: BAKUROCHOO
- TOOKYOO :: BUNKYOO
- TOOKYOO :: BOOSOO
- TENRYUU :: CHANPION
- RUMOI :: NYORAI
- EDOGAWA :: GYOODA

- RUMOI :: JARI
- YAESU :: JOOETSU
- RUMOI :: BYOOIN
- HARAJUKU :: HAPPYAKU
- GUNMA :: PYUUMA

city011 で city022 を認識

- MARUNOUCHI :: KUNITACHI
- HYOOGO :: SUGAMO
- ZOOSHIGAYA :: SOSHIGAYA
- MEJIRO :: HONJO
- BEPPU :: YAESU
- SHINAGAWA :: GINZA
- TOOKYOO :: BAKUROCHOO
- TOOKYOO :: BUNKYOO
- TOOKYOO :: BOOSOO
- TENRYUU :: CHANPION
- KOGANEI :: NYORAI
- RUMOI :: JUUMONJI
- RUMOI :: BYOOIN
- GYUUBA :: PYUUMA

city012 で city011 認識

- BOOSOO :: TOOKYOO

city012 で city012 を認識
なし
city012 で city021 を認識

- ZOOSHIGAYA :: SOSHIGAYA
- POPURA :: NOGATA
- BEPPU :: YAESU
- EDOGAWA :: GINZA
- RISHIRI :: ZUSHI
- HYUUGA :: GYOODA
- SHUZENJI :: JARI
- RYOOGOKU :: JOOETSU
- HYUUGA :: PYUUMA

city012 で city022 を認識

- MARUNOUCHI :: KUNITACHI
- SHINAGAWA :: GINZA
- RISHIRI :: ZUSHI
- BIZEN :: NYUUZEN
- BEPPU :: RYAKUZU
- GYUUBA :: PYUUMA

city021 で city011 を認識

- UENO :: SUGAMO
- HIBIYA :: NERIMA
- MYAKUHAKU :: RAUSU
- NERIMA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- BOOSOO :: BAKUROCHOO
- BEPPU :: MYAKUHAKU
- CHIYODA :: GYOODA
- KICHIJOOJI :: JUUMONJI
- YAESU :: JOOETSU

city021 で city012 を認識

- HIBIYA :: NERIMA
- BIZEN :: REBUN
- SHINAGAWA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- BEPPU :: PINPON
- CHIYODA :: GYUUBA
- CHIYODA :: GYOODA
- KICHIJOOJI :: JUUMONJI

city021 で city021 を認識

なし

city021 で city022 を認識

なし

city022 で city011 を認識

- UENO :: SUGAMO
- SENJU :: TENRYUU
- DATE :: RAUSU
- PYUUMA :: GUNMA
- CHIYODA :: JIYUUGAOKA
- TSUKIJI :: ZUSHI
- TAKAO :: BOOSOO
- SENJU :: PURIZUMU
- NOGATA :: POPURA
- GYANGU :: NYORAI
- PYUUMA :: HYUUGA
- TAKAO :: HYOOGO
- HYAKKATEN :: MYAKUHAKU
- CHIYODA :: GYUUBA
- CHIYODA :: GYOODA
- KICHIJOOJI :: JUUMONJI

city022 で city012 を認識

- UENO :: SUGAMO
- SENJU :: TENRYUU
- DATE :: RAUSU
- BIZEN :: REBUN

- DATE :: WASEDA
- KESENNUMA :: GUNMA
- CHIYODA :: JIYUUGAOKA
- SOSHIGAYA :: ZOOSHIGAYA
- BEPPU :: DENENCHOOFU
- BEPPU :: PINPON
- BIZEN :: PURIZUMU
- RYAKUZU :: MYAKUHAKU
- PYUUMA :: GYUUBA
- KICHIJOOJI :: JUUMONJI

city022 で city021 を認識

- ZOOSHIGAYA :: SOSHIGAYA

city022 で city022 を認識
なし

### 5.2 斜め倍率 : $\sqrt{2}$

斜め倍率を $\sqrt{2}$ とした時の認識率は以下の表 2 のようになった.

表 2: 斜め倍率 $\sqrt{2}$

| | | 教師データ | | | |
|---|---|---|---|---|---|
| | | city_011 | city_012 | city_021 | city_022 |
| テストデータ | city_011 | 100 | 99 | 94 | 92 |
| | city_012 | 100 | 100 | 95 | 94 |
| | city_021 | 92 | 98 | 100 | 100 |
| | city_022 | 94 | 95 | 100 | 100 |

また，誤認識した組み合わせを以下に示す．左が誤認識結果，右が実際の答えである．city011 で city011 を認識
なし
city011 で city012 を認識
なし
city011 で city021 を認識

- TENRYUU :: SENJU
- YAESU :: RAUSU
- TOOKYOO :: BAKUROCHOO
- TOOKYOO :: BOOSOO
- RUMOI :: NYORAI
- EDOGAWA :: GYOODA
- YAESU :: JOOETSU
- HYUUGA :: PYUUMA

city011 で city022 を認識

- MARUNOUCHI :: KUNITACHI

- TOOKYOO :: BOOSOO
- KOGANEI :: NYORAI
- YAESU :: JOOETSU
- RUMOI :: BYOOIN
- HYUUGA :: PYUUMA

city012 で city011 を認識

- BOOSOO :: TOOKYOO

city012 で city012 を認識
なし
city012 で city021 を認識

- BEPPU :: YAESU
- RISHIRI :: ZUSHI

city012 で city022 を認識

- NIHONBASHI :: KUNITACHI
- TENRYUU :: SENJU
- RISHIRI :: ZUSHI
- BIZEN :: NYUUZEN
- GYUUBA :: PYUUMA

city021 で city011 を認識

- UENO :: SUGAMO
- HIBIYA :: NERIMA
- NERIMA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- CHIYODA :: GYOODA
- SHINAGAWA :: JOOETSU

city021 で city012 を認識

- HIBIYA :: NERIMA
- NERIMA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- HYUUGA :: GYUUBA
- CHIYODA :: GYOODA

city021 で city021 を認識
なし
city021 で city022 を認識
なし
city022 で city011 を認識

- UENO :: SUGAMO

- PYUUMA :: GUNMA
- TAKAO :: BOOSOO
- OCHANOMIZU :: PURIZUMU
- DATE :: NYAKUSEI
- PYUUMA :: HYUUGA
- PYUUMA :: GYUUBA
- KICHIJOOJI :: JUUMONJI

city022 で city012 を認識

- SENJU :: TENRYUU
- KESENNUMA :: GUNMA
- CHIYODA :: JIYUUGAOKA
- BIZEN :: PURIZUMU
- PYUUMA :: GYUUBA
- KICHIJOOJI :: JUUMONJI

city022 で city021 を認識
なし
city022 で city022 を認識
なし

### 5.3 斜め倍率 : 1

斜め倍率を 1 とした時の認識率は以下の表 3 のようになった.

表 3: 斜め倍率 1

| | | 教師データ | | | |
|---|---|---|---|---|---|
| | | city_011 | city_012 | city_021 | city_022 |
| テストデータ | city_011 | 100 | 99 | 95 | 94 |
| | city_012 | 100 | 100 | 96 | 98 |
| | city_021 | 92 | 99 | 100 | 100 |
| | city_022 | 95 | 98 | 100 | 100 |

また，誤認識した組み合わせを以下に示す．左が誤認識結果，右が実際の答えである．city011 で city011 を認識
なし
city011 で city012 を認識
なし
city011 で city021 を認識

- GUNMA :: NERIMA
- YAESU :: RAUSU
- TOOKYOO :: BAKUROCHOO
- TOOKYOO :: BOOSOO
- EDOGAWA :: GYOODA
- YAESU :: JOOETSU
- RUMOI :: BYOOIN

- HYUUGA :: PYUUMA

city011 で city022 を認識

- TOOKYOO :: BOOSOO
- KOGANEI :: NYORAI
- YAESU :: JOOETSU
- RUMOI :: BYOOIN
- HYUUGA :: PYUUMA

city012 で city011 を認識

- BOOSOO :: TOOKYOO

city012 で city012 を認識
なし
city012 で city021 を認識

- RISHIRI :: ZUSHI

city012 で city022 を認識

- RISHIRI :: ZUSHI
- CHIYODA :: GYOODA

city021 で city011 を認識

- HIBIYA :: NERIMA
- NERIMA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- NOGATA :: GYOODA
- SHINAGAWA :: JOOETSU

city021 で city012 を認識

- HIBIYA :: NERIMA
- NERIMA :: GUNMA
- SOSHIGAYA :: ZOOSHIGAYA
- HYUUGA :: GYUUBA

city021 で city021 を認識
なし
city021 で city022 を認識
なし
city022 で city011 を認識

- UENO :: SUGAMO
- TOOKYOO :: BOOSOO

- SENJU :: PURIZUMU
- EDOGAWA :: POPURA
- PYUUMA :: HYUUGA
- PYUUMA :: GYUUBA

city022 で city012 を認識

- KESENNUMA :: GUNMA
- PYUUMA :: GYUUBA

city022 で city021 を認識
なし
city022 で city022 を認識
なし

### 5.4 結果まとめ

今回の斜め倍率を 2 から $\sqrt{2}$, $\sqrt{2}$ から 1 へと変化させたときの誤認識関連のデータを以下の表 4 に記す.

表 4: 認識率の推移

| | 変わらず誤認識 | 誤認識の結果が変わった | 正しく認識された | 新たに誤認識になった |
|---|---|---|---|---|
| $2 \rightarrow \sqrt{2}$ | 36 | 7 | 53 | 4 |
| $\sqrt{2} \rightarrow 1$ | 28 | 3 | 21 | 4 |

### 5.5 グラフ

DP マッチングを行った際に通ったルートのグラフ化行う．まず，正しく認識できていた単語で，正しい組み合わせとそれとは異なる組み合わせのグラフを以下の図 1，2 に記す.

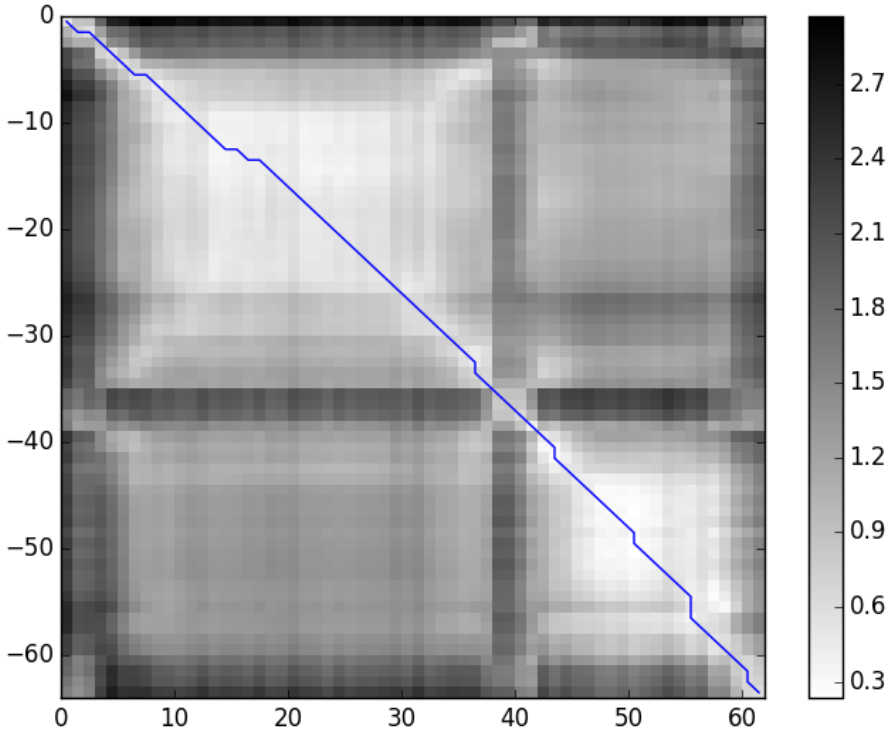


図 1: 同じ単語の DP の例 (011GYOODA，012GYOODA，距離 0.280)

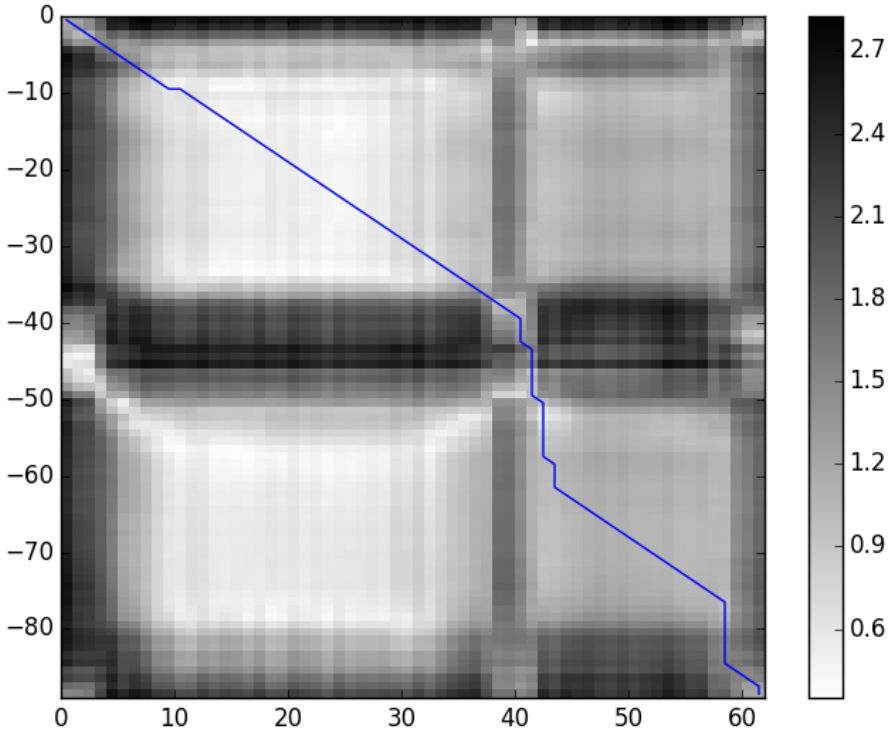


図 2: 違う単語の DP の例 (011TOOKYOO，012GYOODA，距離 0.589)

次に同一話者で唯一認識できなかった city012 のデータで city011 の “TOOKYOO” の認識の誤認識結果のルートと正しいルートを以下の図 3〜8 に記す.

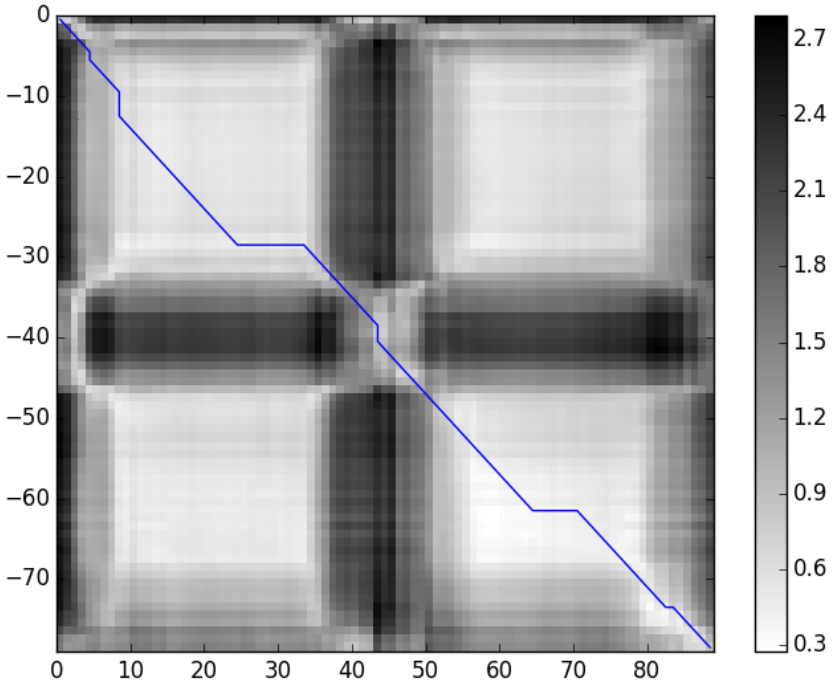


図 3: 斜め 2 での 012 BOOSOO と 011 TOOKYOO の DP(距離 : 0.603)

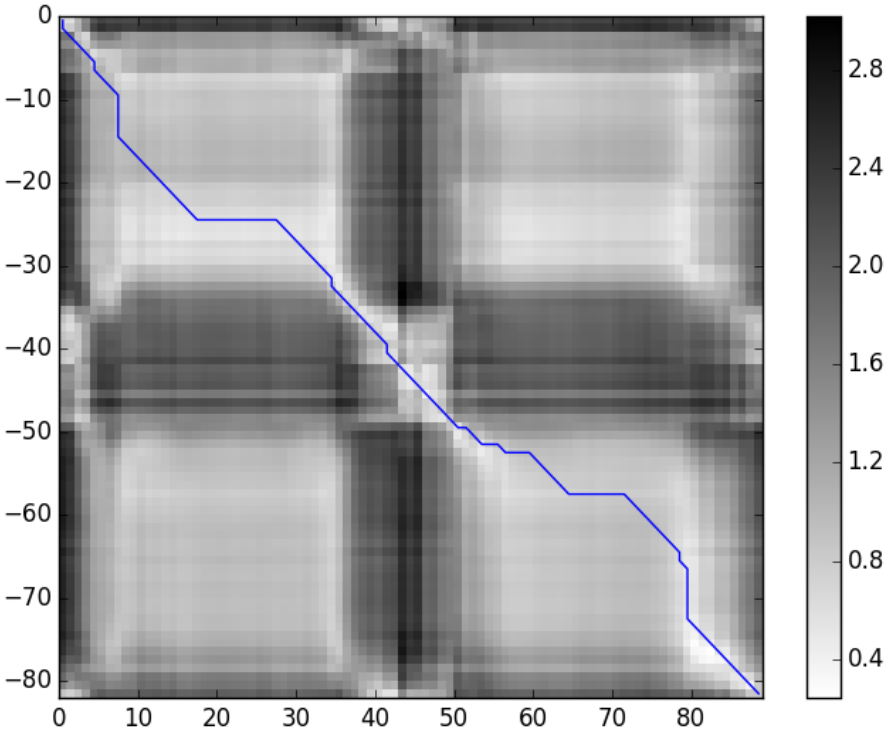


図 4: 斜め 2 での 012 TOOKYOO と 011 TOOKYOO の DP(距離 : 0.611)

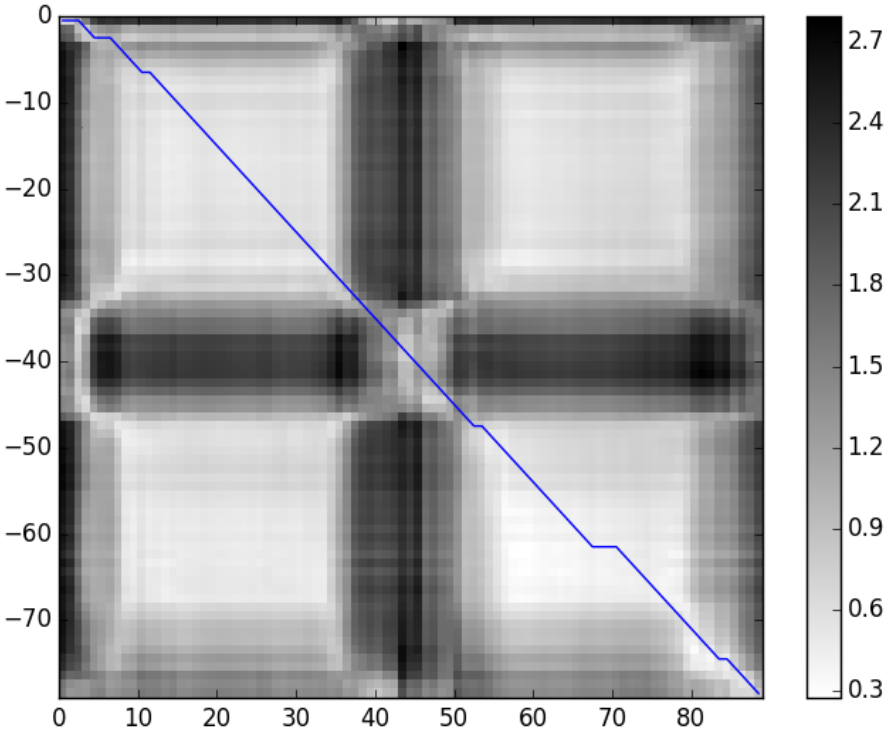


図 5: 斜め $\sqrt{2}$ での 012 BOOSOO と 011 TOOKYOO の DP(距離 : 0.479)

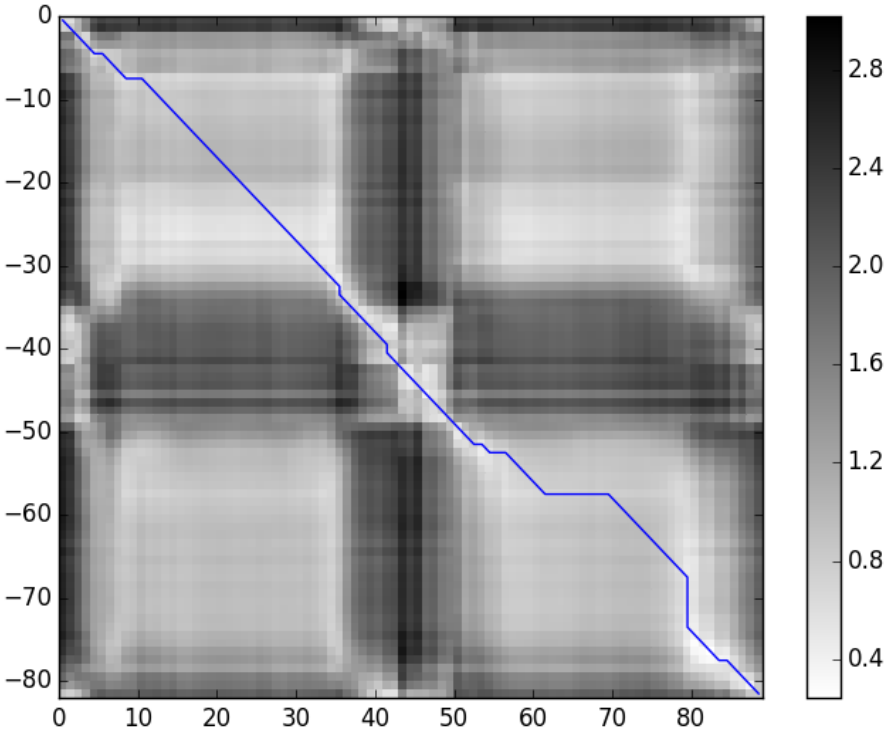


図 6: 斜め $\sqrt{2}$ での 012 TOOKYOO と 011 TOOKYOO の DP(距離 : 0.510)

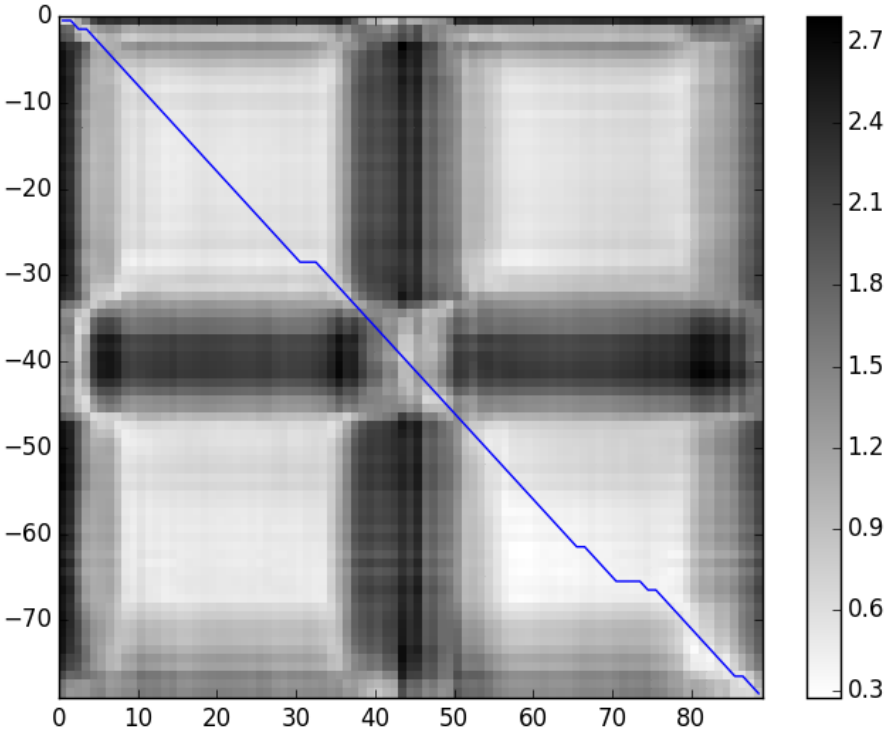


図 7: 斜め 1 での 012 BOOSOO と 011 TOOKYOO の DP(距離 : 0.353)

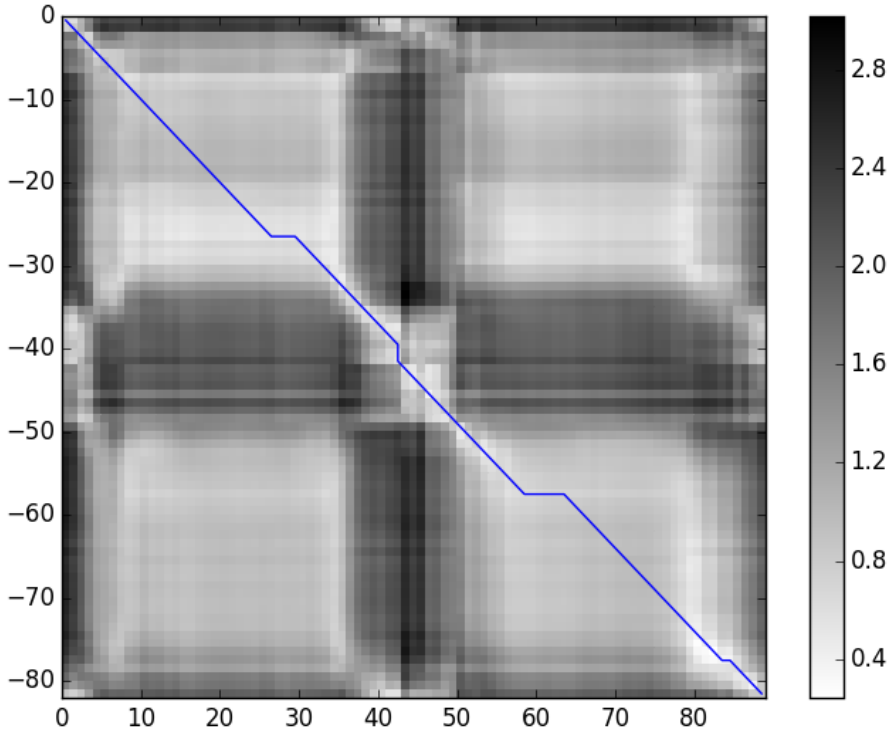


図 8: 斜め 1 での 012 TOOKYOO と 011 TOOKYOO の DP(距離 : 0.378)

### 5.6 正しい認識時と誤認識時の距離平均

ここまでを通して誤認識時の距離平均は正しい認識時の時の距離平均より長いのではないかと仮説を立てそれぞれの距離平均を算出した．以下の表 5 の通りになった

表 5: 平均距離

| | 正しい認識 | 誤認識 |
|---|---|---|
| 斜め倍率:2 | 1.030 | 1.060 |
| 斜め倍率: $\sqrt{2}$ | 0.911 | 0.954 |
| 斜め倍率:1 | 0.7489 | 0.7916 |

## 6 考察

今回の実験を通して，まず DP マッチングは特徴量の時系列の長さが異なる場合でも高い認識率を得るということである．これに関しては最後に長さで正規化しているため長さよる距離の差が出ないからであると考える．

次に斜めの倍率に関して 2 倍，$\sqrt{2}$ 倍，1 倍とやってきたが，それぞれの意味として，2 倍は斜め先のマスにたどり着くためには横に一つ，下に一つ動かなければならないため一回で斜めに移動す場合は 2 倍するという意味であり，$\sqrt{2}$ 倍はユークリッドノルムの考え方で一辺よりも $\sqrt{2}$ 倍であることからである．最後に 1 倍は 8 近傍すべて隣り合ったマスであるからということであると考える．

ただ倍率が下がるほど斜めのつながりが強くなりより斜めに移動することが多くなっていることがわかる．本実験の場合単純に認識率だけで見たら斜め倍率が 1 倍のほうが良いしかし図 4 に記すように新たに誤認識となったものもあることから一概に 1 倍がよいとは言い切れない．

また，今回の認識結果では同一話者の認識率は非常に高かった．これより今回使用されたメルケプストラム特徴量は同一話者に対しては非常に強力な特徴量であることがわかる．

次に正しい認識時と誤認識時の距離平均を見ると予想通り誤認識時のほうが距離平均が長くなっている．また誤認識が起こったのがほとんど違う話者である．つまりこれは，誤認識された単語が正しい単語に似ているというよりは正しい単語の発生の仕方に個人個人の癖が出やすいものなのではないかと考える．