POLITECHNIKA ŚLĄSKA W GLIWICACH

Wydział Matematyki Stosowanej

Kierunek Informatyka

Inżynierskie, stacjonarne, semestr V

Dokumentacja

Projekt: Zliczanie liter na zdjęciu

Data: 22.01.2018r.

Autorzy: Beniamin Stecuła

Damian Szala

Michał Kloc

Założenia projektu:

Projekt ma na celu rozpoznanie liter za zdjęcia planszy do gry Scrabble a następnie zliczenie liter użytych podczas gry i porównanie ich do zestawu startowego w celu sprawdzenia kompletności płytek.

Realizacja:

Projekt został wykonany przy użyciu języka Python i kompilatora Python 3.6.2 (64-bit) i poniżej prezentujemy stworzony kod:

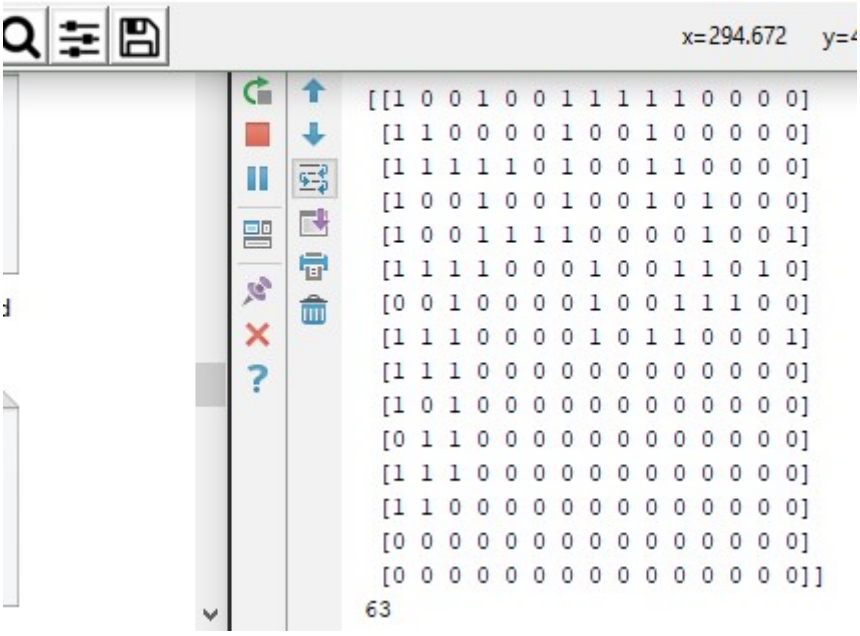
```
import os
import numpy as np
from PIL import Image
from sklearn import svm
from sklearn import neighbors
from scipy.ndimage import measurements
import matplotlib.pylab as plt
from sklearn import cross_validation

wymiar = 15

def prepare_feature(im):
    #resize
    resize_im = Image.fromarray(np.uint8(im))
    norm_im = np.array(resize_im.resize((30, 30)))

    #remove border
    norm_im = norm_im[3:-3, 3:-3]
    return norm_im.flatten()

def find_sudoku_edges(im, axis = 0):
    """Finds the cell edges for an aligned sudoku image."""

    #threshold and sum rows and columns
    trim = 1 * (im < 128)
    s = trim.sum(axis = axis)

    #find center of strongest lines
    s_labels, s_nbr = measurements.label(s > (0.5 * max(s)))
    m = measurements.center_of_mass(s, s_labels, range(1, s_nbr + 1))
    x = [int(x[0]) for x in m]

    #if only the strong lines are detected add lines in between
    if len(x) == 4:
        dx = np.diff(x)
        x = [x[0], x[0] + dx[0] / 3, x[0] + 2 * dx[0] / 3,
            x[1], x[1] + dx[1] / 3, x[1] + 2 * dx[1] / 3,
            x[2], x[2] + dx[2] / 3, x[2] + 2 * dx[2] / 3, x[3]]
    if len(x) == wymiar+1:
        return x
    else:
        raise RuntimeError("Edges not detected.")

def find_sudoku_edges2(im, axis = 0):
    if axis == 0:
        leng = len(im[0])
    else:
```

```
        leng = len(im)
    x = [1, leng/wymiar, 2*leng/wymiar, 3*leng/wymiar, 4*leng/wymiar,
         5*leng / wymiar, 6*leng/wymiar, 7*leng/wymiar, 8*leng/wymiar,9*leng/wymiar,
         10*leng / wymiar,11*leng/wymiar, 12*leng/wymiar, 13*leng/wymiar, 14*leng/wymiar,
         leng-1]
    return x

def load_ocr_data(path):
    """
    Return labels and ocr features for all images
    inpath.
    """

    #create list of all files ending in.jpg
    imlist = [os.path.join(path, f) for f in os.listdir(path) if f.endswith(".jpg")]
    #create labels "1" "A"-nazwa litery
    labels = [int(imfile.split("/")[-1][0:2]) for imfile in imlist]
    print(labels)
    #create features from the images
    features = []
    for imname in imlist:
        im = np.array(Image.open(imname).convert("L"))

        norm_im = prepare_feature(im)
        features.append(norm_im)
    return np.array(features), labels

#DATA
data,target=load_ocr_data("data/")
features, test_features, labels, test_labels = cross_validation.train_test_split(data,
target, test_size=0.3, random_state=0)

#Create SVM classificator
clf = svm.SVC(C=1, kernel='linear')
#clf = svm.SVC(C=0.1, kernel='sigmoid')
#clf = svm.SVC(C=0.1, kernel='rbf')

#Create kNN classificator
#clf = neighbors.KNeighborsClassifier(10, weights='uniform')
#clf = neighbors.KNeighborsClassifier(10, weights='distance', p=2)

#Fitting
clf.fit(features, labels)

#Checking
pred = clf.predict(test_features)

print(np.count_nonzero((np.array(test_labels) - pred)))
print(100*((len(test_labels) - np.count_nonzero((np.array(test_labels) - pred))) /
(1.0*len(test_labels))))

#Reading sudoku
imname = "sudoku/sudoku23.JPG"
vername = "sudoku/sudoku23.sud"

im = np.array(Image.open(imname).convert("L"))

ax = plt.subplot(111)
ax.imshow(im, cmap=plt.cm.Greys_r)
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

#find the cell edges
x = find_sudoku_edges2(im, axis = 0)
y = find_sudoku_edges2(im, axis = 1)
```

```
#crop cell sand classify
crops = []
for col in range(wymiar):
    for row in range(wymiar):
        crop = im[int(y[col]):int(y[col+1]), int(x[row]):int(x[row+1])]
        crops.append(prepare_feature(crop.copy()))

pred_sudoku = clf.predict(crops)
sudoku = pred_sudoku.reshape(wymiar,wymiar)

print("Result:")
print(sudoku)

jest = 0
for i in range(len(sudoku)):
    for j in range(len(sudoku[0])):
        if (sudoku[i][j]>0 and sudoku[i][j]<31):
            jest+=1
print(jest)

plt.show()
```

Wynik końcowy:

Program nam pokazuje w których miejscach znalazł litery i zlicza ich ilość.



Podział ról:

Beniamin Stecuła – koncepcja, project manager, rozwój projektu, testing,

Damian Szala – obróbka graficzna, dokumentacja,

Michał Kloc – stworzenie podstaw projektu, obróbka graficzna.

System kontroli wersji:

Prowadziliśmy kontrolę wersji przy użyciu platformy GitHub pod adresem:

https://github.com/IanTebis/rakontante-literojn-en-foto