OCR based Implementation

(Step by step explanation for logo matching)

Description:

- The implementation only works for the image with some text written (text may be the name of the company or any letter to resemble uniqueness of the company)
- Furthermore, this can be used as an extra utility to show the name of the company and the existing company (if exits) from which the logo is resembling.

PART 1: Creation of the training data (saved as text files in the save_data folder)

Step 1: Creating training data set for each letter (26 lowercase + 26 upper case + 9 numerals)

- Choosing fonts frequently occurring in the logos and converting .ttl file to .jpeg

- Extracting the text between the two lines (this has all the letters and numbers)

  - This image is manually corrected to detach and letter if joined
  - Function **regionOI_extractor(numberOfFonts)** in file **Extractor_dataset.py** does this for all the .jpeg images in a folder

- Every letter is cropped out form the image and saved in the folder data_set1 as-

  - The wrong detections if any must be deleted (I have kept best 10 cropped out images for each sample)
  - Function **letterExtractor()** in file **Extractor_dataset.py** does this for all the letters in the image and uses following:
    I) **Letter_Extractor.py** file for all the extracting letters in an order form left to right.
    II) **save_data.py** file to save cropped letter automatically after the previously existing samples with proper name.
  - This creates 62 folder in the data_set1 folder containing 10 samples of every letter.

- Next, the above cropped letters are threshed inverted as files are saved as text files(.txt)

  - Class **createTextFile()** in **Training.py** is used for this purpose
  - The resultant text files is saved in flat_text folder having 52 folders for each sample

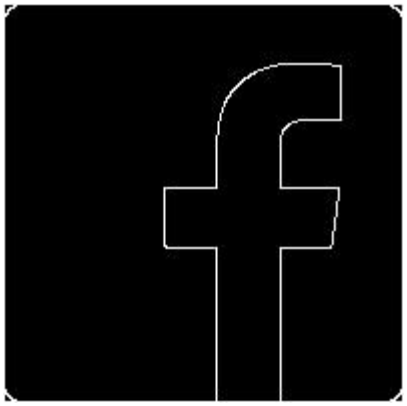
- Now the all the text files are combined as a single text file for creating a set of data for the trainer finally saved in saved_data folder.
  - **Classification.txt:** contains the letter with which the sample matches.

    eg. matrix: [97. 97. 97. 97. 97. 97. 97. 97. 97. 97. 96. 96. 96. 96.……0. 0. 0. 0. 0. 0 .0.]
    resembling a a a a a a……A A A A A…… 0 0 0 0 ….

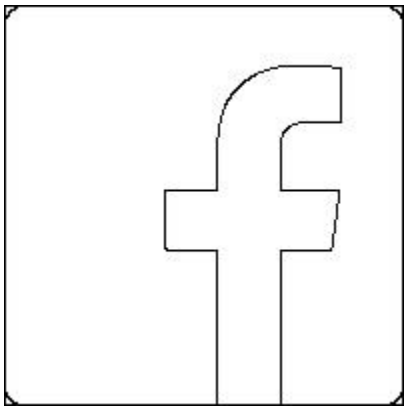
  - **flattenedImages.txt:** contains combined text files created from above procedure.
    eg. matrix: [[0. 0. 0. 255.…….0. 0. 255.]
    [255. 0. 0……0.]
    ……..
    [0. 0. 0. 0. 0. 0. 0. 0.]]
  - Matrix are float32 form with shape (620,) and (620,576) as all images are cropped with 24x24 size.
  - These files can be used directly and above process need not be repeated.
  - But the files are flexible enough to automate the process for implementing more efficient learning procedure in future.

PART 2: Letter extraction (General Process)

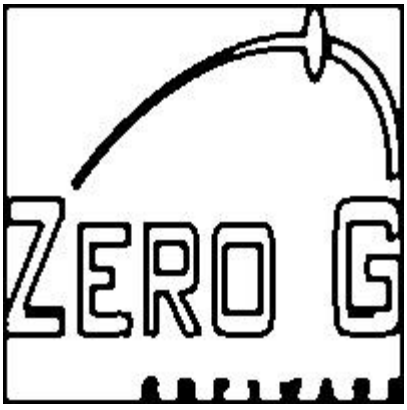
- This is the most imp part of the implementation to detect the letters from any general image. As logos are highly variable in colors, text, background and other shapes, so this makes this process tough to generalize.
- Implementation is still in process but work done till date is shown below.
- File **General_Letter_Extractor.py** has series of functions which perform this task.
- Step 1: finding edges using canny edge detection algorithm.

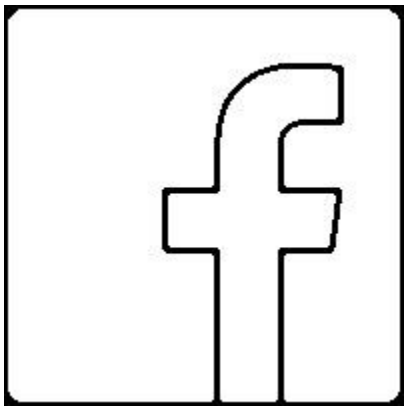


- Step 2: Threshing the image for properly detecting contours.



- Step 3: blurring to remove small dots and erosion of the edged to get significant edges





  - A rectangle around the boundaries is intentionally made to detect the letters which are not fully closed contours or half cut by the edges (as in Facebook logo)

- Step 4: finding contours and removing unwanted contours.

  - Unwanted contours are removed using the area of the contour, hierarchy methods and few image matrix reconstructions.
  - Still the unwanted contours between the letters or near the edges are to be removed.
  - The ‘ f ’ from Facebook is perfectly detected as it is simplest logo with no designing.

- This process requires lots of further additions for generalization but works well for extremely simple images.

PART 3: Recognizing the letters detected by the **General_Letter_Extractor.py**

- File **ML_Testing.py** has the classes having implementation of K-nearest and SVM machine learning methods for initial testing.
- Run **database_matching.py** file to view the search results.
- Step 1: Extract the letters bounding rectangle and pass into trainer to get the output.

  - These are resized into 24x24 as trainer requires the same shape as that of saved data in flattenedImages.txt

- Step 2: The result is stores in and array as [‘ f ’]
  - Results from every image in the database in again stored in different array as [‘ a ’, ‘ p ’, ‘ p ’, ‘ l ’, ‘ e ’]
  - These results are matched for similar sets.
  - If the number is good (greater than a threshold) then the output is displayed with the company name as printed on the logo.

Example result for Facebook logo having text ‘f ’: (tested with 50 images in test_cases folder)

Letters recognised from source image are: ['f']

Checking input querry in dataset data set...
file name : data (1).jpg
Letters recognised: ['B', 'D', 'E', 'D', 'E', 'd', 'I', 'U', 'I', '0', 'z', '0', 'I', 'k', 'i', 'I', 'A']

file name : data (2).jpg
Letters recognised: ['g', 'E', 'g', 'E', '7', 'd', 'E']

file name : data (3).jpg
Letters recognised: ['n']

file name : data (4).jpg
Letters recognised: ['u', 'L', 'I', '6', 'r', 'D', '\t', 'n', 'D', 'I', 'E', 'm', 'F']

file name : data (5).jpg
Letters recognised: ['1', 'N', '2', '3', '7', 'n', '7', '7', '\t']

file name : data (6).jpg
Letters recognised: ['8', 'I', '7']

file name : data (7).jpg
Letters recognised: ['5', 'i', 'c']

file name : data (8).jpg
Letters recognised: ['J', 'L', 'I', 'I', 'E', 'E', '8', 'F']

file name : data (9).jpg
Letters recognised: []

file name : data (10).jpg
Letters recognised: ['V', '7', 'H']

file name : data (11).jpg
Letters recognised: ['f']
Your company named ['f'] may resemble this logo

file name : data (12).jpg
Letters recognised: ['I']

file name : data (13).jpg
Letters recognised: ['7', '0']

file name : data (14).jpg
Letters recognised: ['7', 'm', '3', 'Q']

file name : data (15).jpg
Letters recognised: ['I', 'I', 'I', 'I']

file name : data (16).jpg
Letters recognised: ['J', 'Y', '3', '4', 'o']

file name : data (17).jpg
Letters recognised: ['W', '5']

file name : data (18).jpg
Letters recognised: []

file name : data (19).jpg
Letters recognised: ['V', 'D', '6', 'D', '\t', 'm', 'W', 'M', 'P', 'M', 'R', 'x', 'm']

file name : data (20).jpg
Letters recognised: ['3', 'M', '3', 'M']

file name : data (21).jpg
Letters recognised: ['e', 'D', 'e', 'D', 't', 'K', 'A', 'D', 's', '8', 'x', '7', '7']

file name : data (22).jpg
Letters recognised: ['M', 'M', '7', 'M', 'k', '0', 'H', 'k', 'H', 'M', 'm']

file name : data (23).jpg
Letters recognised: ['r', 'M', 'M', 'm', '7', 'o', 'L', 'M', '7', 'M', 'M', 'E', 'M', 'm']

file name : data (24).jpg
Letters recognised: ['n', 'R', 'H', 'K', 'A', 'H', '7', '7', 's']

file name : data (25).jpg
Letters recognised: ['M', 'E', 'B', 'E', 'D', 'H', 'N', 'D', 'L', 'z', 'H', 'A', '0']

file name : data (26).jpg
Letters recognised: ['M', 'y', 'T']

file name : data (27).jpg
Letters recognised: ['J', 'L', 'A', 'n', 'N', 'F']

file name : data (28).jpg
Letters recognised: ['P', 'n', 'E', 'P', 'n', 'o', 'H', 'A', 'A', 'B', 'n', '0', 'A', 'C', 'N', 'O']

file name : data (29).jpg
Letters recognised: ['I']

file name : data (30).jpg
Letters recognised: ['7', 'M', 'R', '7', 'M', 'B', 'E', 'D', 'I', 'c', 'D', 'E', '7', 'A', '3', 'M', '6']

file name : data (31).jpg
Letters recognised: ['M', 'E', 'D', 'F', 'm', 'I', 'n']

file name : data (32).jpg
Letters recognised: []

file name : data (33).jpg
Letters recognised: ['5', '5', 'D']

file name : data (34).jpg

Letters recognised: ['3', 'z']

file name : data (35).jpg
Letters recognised: ['I', 'I', 'E', '7', '6', 'N', '3', 'I', 't', 'B']

file name : data (36).jpg
Letters recognised: ['f']
Your company named ['f'] may resemble this logo

file name : data (37).jpg
Letters recognised: ['b', 'A', 'D', '7', 'f', 'f', '\t']
Your company named ['f'] may resemble this logo

file name : data (38).jpg
Letters recognised: ['5']

file name : data (39).jpg
Letters recognised: []

file name : data (40).jpg
Letters recognised: ['z', '7', '7', '\t', '6', 'N', 'c', 'B', 'm', 'M']

file name : data (41).jpg
Letters recognised: ['r', 'R', 'D', 'T', '7', '7', '7', '7', 'm', '7', 'E', 'D', 'K', 'D']

file name : data (42).jpg
Letters recognised: ['E', '7', 'D', '7', 'E', 'f', 'D', 'D', 'D', 'I', 'M', 'J', '\t', 'D', '8', 'H', '2', 'r']
Your company named ['f'] may resemble this logo

file name : data (43).jpg
Letters recognised: ['J', 'L', 'I', 'F']

file name : data (44).jpg
Letters recognised: ['\t']

file name : data (45).jpg
Letters recognised: ['E', 'I', 'F']

file name : data (46).jpg
Letters recognised: ['\t', 'H', 'O', 'i', 'c', 'K', 'D', 'm', 'c', 'K', 'o', 'm', 'o', 'm', 'o', 'm', '\t', 'm', '\t', 'n', '\t', 'm', '\t', 'm', '\t', 'H', '\t', 'H']

file name : data (47).jpg
Letters recognised: ['3', 'K', 'Z', '7']

file name : data (48).jpg
Letters recognised: ['D', 'I', 'B', 'E', 'E', 'M', '3', 'K', 'A', 'A', '7']

file name : data (49).jpg
Letters recognised: ['J', 'L', 'H', 'E', 'N', 'F']

file name : data (50).jpg
Letters recognised: ['0', 'A', 'j', '7', '7', '7', '4', 'm']

Expected Detection:

Detection Obtained:

Problems:

- Wrong contours detected as letters.
- Wrong detection of letter f.
- Main reason for the wrong detection is the highly inaccurate cropped images of letters. The general letter detection algorithm is still to be figured out which can work on all types of images.
- K-nearest trainer correctly recognizes ‘ f ‘ from the input image which shows that trainer works perfectly if the letter is cropped accurately.

Solution: Develop general Algorithm for cropping out letter from all types of logos.