ST456 - FINAL PROJECT: OCR for information extraction for semi-structured scanned documents

> MOTIVATION

As an increasing number of companies across various industries focus more heavily in digital transformation, and as a result, the demand for highly accurate OCR technology is at its all-time peak. This drives the motivation of the project, which is to develop a basic data pipeline to extract key information from scanned printed documents and label the corresponding text with its entities. Enabling automation of data analysis, reducing storage costs, easing the process of dealing with administrative paperwork, improving data management, and finally adopting a more eco-friendly approach is the end goal of digitalisation. We will be focusing on OCR, one of the key technologies required to enable such transformation.

> DESCRIPTION OF OCR

Optical Character Recognition (OCR), also known as Optical Character Reader, involves taking an input of scanned printed documents and outputting the entities (e.g. address, name, price, etc.). This approach uses a combination of hardware and software to digitalise physical printed documents into machine readable text. This innovative approach takes advantage of advanced AI methods to identify different types of fonts and languages, ultimately broadening the scope of its utilisation.

> METHODOLOGY

This will be achieved by splitting the work into three separated tasks, all working subsequently. The first task, called ‘Text Localisation’, implements a deep learning-based OCR model to scan through images, extracting coordinates where printed information appears.The second task, called ‘Text Recognition’, utilises the coordinates to crop the line-based images and recognize the text within the image. The third task, called ‘Key-Information Extract’, uses the recognized text and coordinates to label the corresponding entities with their desired categories and outputs a file grouping those specific information. Traditionally models are developed to tackle the entire challenge of OCR, which combine Text Localisation and Text Recognition, but we believe dealing with each task separately can lead to a more specific model architecture to be developed in the future.

↳ 1 cell hidden

DATA DESCRIPTION

> Our Data comes from the publicly available dataset called SROIE (scanned receipts OCR and key information extraction). The data consist of jpg images for receipts, a txt file containing the bounding boxes' coordinates together with the text, and a json file which contains the entities labels for the company name, address, date, and total amount. The dataset should contain 626 images, txt files, and json files, but the dataset contains some duplicates which needs to be removed, this leads to the data cleaning stage.

> The txt file contains the bounding box in the following manner: (x-y)-s of the corners of the bboxes and the corresponding text.

DATA CLEANING

Deduplication for Task 1 & 2 (Create a new folder "DeduplicatedTask1")

[ ] ↳ *25 cells hidden*

Deduplication for Task 3 (Create a new folder "DeduplicatedTask3")

[ ] ↳ *26 cells hidden*

TASK 1 - TEXT LOCALISATION

> Localizing and recognizing text from scene images captured by a camera is a conventional step to data extraction. This task is proposing the implementation of a Tesseract optical recognition character engine to create boxes around the identified text from printed invoices. Its version 4.00 includes a new neural network subsystem configured as a text line recognizer, originating from LSTM neural network. This one was pre-trained on different types of datasets to recognise different types of fonts and languages. A Convolutional Neural Network is performed to recognise an image that contains any type of character. Following this one, RNNs, and more specifically under the form of LSTM, plays the role of detecting text sequence of characters. Therefore, this particular version of tesseract responds to the goal of our first task, that is detecting text at a word-level.

> Using pytesseract, the model was utilised to execute the task at hand. All jpg images are being pre-processed and converted into a RGB mode as well as resized to improve data quality, essential to improve the model's accuracy. The following step consists in creating bouding boxes around texts detected from tesseract. Also known as bbox, these were solely visualised on a single file as display them for all images would slow the process grantly. Finally, getting the data out of those boxes provided with the resultant outcome, and provide task 2 with the necessary outcome to proceed: the coordinates of the texts within the images. Testing this model consisted in creating a matching analysis comparing both txt files and the resultant bounding boxes achieved from running tesseract on jpg files.

```
#enable access to google drive files
from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir('/content/drive/MyDrive/ST456-Final-Project')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

```
!pip install gitpython
!sudo apt install tesseract-ocr
!pip install pytesseract
!pip install Pillow==9.0.0
```

```
    Requirement already satisfied: gitpython in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.7/dist-
    Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/pyth
    Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.7/dist-
    Reading package lists... Done
    Building dependency tree
    Reading state information... Done
    tesseract-ocr is already the newest version (4.00~git2288-10f4998a-2).
    0 upgraded, 0 newly installed, 0 to remove and 41 not upgraded.
    Requirement already satisfied: pytesseract in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.7/dist-pa
    Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.7/dist-
    Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python
    Requirement already satisfied: Pillow==9.0.0 in /usr/local/lib/python3.7/dist-pa
```

```
import cv2
import glob
import os.path as osp
import pandas as pd
import numpy as np
from os import listdir
from os.path import isfile, join
import tensorflow as tf
import glob
```

```
from PIL import Image
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
import pytesseract
from pytesseract import Output
from pathlib import Path
import os
import glob
import json
import random
from pathlib import Path
from difflib import SequenceMatcher
import shutil
from PIL import Image, ImageDraw, ImageFont
import cv2
import pandas as pd
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
from IPython.display import display
import matplotlib
from matplotlib import pyplot, patches
```

DATA EXTRACTION

```
def identify_files_in_directory(path):
  """
  Identifing all files in the given directory

  :param path: String, the path
  :return: list, list of all objects in the given path
  """
  return sorted([f for f in listdir(path) if isfile(join(path, f))])
```

```
## Function to plot multiple images
def plot_img(images, titles):
  fig, axs = plt.subplots(nrows = 1, ncols = len(images), figsize = (15, 15))
  for i, p in enumerate(images):
    axs[i].imshow(p, 'gray')
    axs[i].set_title(titles[i])
    #axs[i].axis('off')
  plt.show()
```

```
#make a list of only jpeg files from folder
import numpy as np
import csv
```

```
path = "/content/drive/MyDrive/ST456-Final-Project/SROIE2019/DeduplicatedTask1"
jpg_txt_list = identify_files_in_directory(path)

jpg_file = []
txt_file = []
for i in jpg_txt_list:
  if i[-3:] == "txt":
    txt_file.append(i)
  else:
    jpg_file.append(i)
```

```
pytesseract. get_tesseract_version()
```

```
    <Version('4.0.0')>
```

SINGLE FILE PRE-PROCESSING

```
img_path = '/content/drive/MyDrive/ST456-Final-Project/SROIE2019/DeduplicatedTask1/X0
img = cv2.imread(img_path, 0)
# Simple thresholding
ret, img_binary = cv2.threshold(img,230,255,cv2.THRESH_BINARY) #THRESH_BINARY
# Plot the images
images = [img, img_binary]
titles = ['Original image', 'Binary image']
plot_img(images, titles)
```

```
#data extraction for a single file using pytesseract
import cv2
import pytesseract
import sys

if len(sys.argv) < 2:
  print('Usage: python ocr_simple.py image.jpg')
  sys.exit(1)

img_path = '/content/drive/MyDrive/ST456-Final-Project/SROIE2019/DeduplicatedTask1/X0
img = cv2.imread(img_path,0)
# Adding custom options
config_cust = ('-l eng --oem 1 --psm 3')
#pytesseract.image_to_data(img, config=custom_config)
text=pytesseract.image_to_data(img_binary, config=config_cust,lang="eng", output_type
print(text.keys())
```

```
dict_keys(['level', 'page_num', 'block_num', 'par_num', 'line_num', 'word_num',
```

TEXT EXTRACTION FROM BOXES

```
#visualisation of boxes
n_boxes = len(text['line_num'])
#n_boxes = len(text['level'])
coordinates=[]
boxes=[]
```

```
  for i in range(n_boxes):
    (x, y, w, h) = (text['left'][i], text['top'][i], text['width'][i], text['height'][i
    words=text['text'][i]
    if not (words == '') or words.isspace():
      img = cv2.rectangle(img_binary, (x, y), (x + w, y + h), (0, 0, 255), 2) #threshol
      boxes.append((x,y,w,h))
      coordinates.append((x,y,w,h,words))
  cv2_imshow(img)
```

MULTIPLE FILES EXTRACTION

```
th_list=[]
fin_img=[]
#threshold = 200

for im in sorted(jpg_file[0:3]):
  new_path =  path + '/' + im
  #print(new_path)
  img_th = cv2.imread(new_path, 0)
  #arr_img
  ret, img_bb = cv2.threshold(img_th,230,255,cv2.THRESH_BINARY) #THRESH_BINARY
  #append img_bb np arrays
```

```
    fin_img.append(img_bb)
    #rgb_img = (Image.open(new_path).convert("RGB")).resize((300, 600))
    th_img = (Image.open(new_path)).resize((300, 600))
    #convert into tif file
    imm = im.replace('.jpg', '.tif')
    #save into data_temp folder temporarily
    th_img.save("/content/drive/MyDrive/ST456-Final-Project/tesstrain/data_temp/"+ imm,
    #append to a list of images
    th_list.append(th_img)

  #get the text for each image
  config_cust = ('-l eng --oem 1 --psm 3')
  list_text=[]
  list_boxes=[]
  for every_img in th_list: #or jpg_file??
    #adding custom options
    text=pytesseract.image_to_data(every_img, config=config_cust,lang="eng", output_typ
    list_text.append(text) #create a list with dicts of every text being extracted of e
    #for n in list_text:
      #numb_boxes = len(n['line_num'])
    list_boxes.append(text['line_num']) #get a list of ints == number of lines each tex


  words_box=[]
  boxes_coord=[]
  text_coord=[]
  for txt in list_text:
    temp=[]
    box_temp=[]
    for i in range(0, len(txt['line_num'])):
      (x, y, w, h) = (txt['left'][i], txt['top'][i], txt['width'][i], txt['height'][i])
      words=txt['text'][i]
      if not (words=='' or words.isspace()):
        temp.append((x,y,w,h,words))
        box_temp.append((x,y,w,h))
    text_coord.append(temp)
    boxes_coord.append(box_temp)
```

TASK 1 EVALUATION - INTERSECTION OF UNION

```
  #A function that opens and reads all the text files in the path provided and then sto
  def read_bbox_and_words(path: Path):
    bbox_and_words_list = []

    with open(path, 'r', errors='ignore') as f:
      for line in f.read().splitlines():
        if len(line) == 0:
          continue

        split_lines = line.split(",")
        bbox = np.array(split_lines[0:8], dtype=np.int32)
        width = bbox[2]-bbox[0]
        height = bbox[5]-bbox[1]
        new_bb = (bbox[0], bbox[1], width, height)
        text = ",".join(split_lines[8:])

        # From the splited line we save (filename, [bounding box points], text line).
        # The filename will be useful in the future
        bbox_and_words_list.append([path.stem, *new_bb, text])

    #Stores the file in a dataframe with columns: 'filename', 'x0', 'y0', 'x1', 'y1', '
    dataframe = pd.DataFrame(bbox_and_words_list, columns=['filename', 'left', 'top', '
    #dataframe = dataframe.drop(columns=['x1', 'y1', 'x3', 'y3'])

    return dataframe


  #access first for elements of the txt file
  path_txt = []
  for subdir, dirs, files in os.walk(Path(path)):
    for file in files:
      if file.endswith(".txt"):
        t=(file)
        path_txt.append(t)
  #bbox_txt = read_bbox_and_words()


  path_all_txt = []
  for i in sorted(path_txt):
    t = (Path(path)/i)
    path_all_txt.append(t)


  list_coortxt =[]
  for k in path_all_txt:
    bbox_txt = read_bbox_and_words(k)
    list_coortxt.append(bbox_txt)


  def bb_intersection_over_union(boxA, boxB):
    x0, y0, width, height = boxA
    left, top, w, h = boxB
    bottom = y0-height
    right = x0+width
    b = top-h
    r = left+w
    #boxA = (x0,y0,bottom,right)
    #boxB = (left,top,b,r)
      # determine the (x, y)-coordinates of the intersection rectangle
    xA = max(x0, left)
    yA = max(y0, top)
    xB = min(bottom, b)
    yB = min(right, r)
      # compute the area of intersection rectangle
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
      # compute the area of both the prediction and ground-truth
      # rectangles
    boxAArea = width * height
    boxBArea = w * h
      # compute the intersection over union by taking the intersection
      # area and dividing it by the sum of prediction + ground-truth
      # areas - the interesection area
    iou = interArea / float(boxAArea + boxBArea - interArea)
      # return the intersection over union value
    return iou


  final_matched = []
  for id, files in enumerate(text_coord):
    matched_box = []
    for ix, line in enumerate(files):

      #print(type(line[4]))
      text_fing = line[4]
      box_a=line[:4]
      #print(type(box_a))
      list_search = list_coortxt[id]['line']
      #coords_search = list_coortxt[id]['line']["left","top","width","height"]
      #list_coortxt[id][[line[4] in lines for lines in list_coortxt[id]['line']]]["left
      #print(type(line[4]))
      #print(type(list_coortxt[id]['line']))
      res = [text_fing.lower() in [i.lower() for i in words.split()] for words in list_
      #print(list_coortxt[id]line[4])
      #if any(line[4] in word for word in list_coortxt[id]['line']):
      matched_line = list_coortxt[id][res][["left","top","width","height"]]
      #print(type(matched_line))
      if not matched_line.empty:
        iou=[]
        for i in range(matched_line.shape[0]):
          #print(i)
          iou_temp = bb_intersection_over_union(box_a,matched_line.iloc[i,:])
          print(iou_temp)
          iou.append(iou_temp)
        max_val = max(iou)

        matched_box.append(max_val)

    final_matched.append(np.array(matched_box))
```

PRE-PROCESSING FOR TRAINING

```
#add files to the intended folder for training
path3 = '/content/drive/MyDrive/ST456-Final-Project/tesstrain/data_temp'
path4= '/content/drive/MyDrive/ST456-Final-Project/tesstrain/data/MODEL_NAME-ground-t

texts_f = [f for f in os.listdir(path3)]

for file in texts_f:
  file_path = os.path.join(path3,file)


  if file_path.endswith('.tif')==True:
    shutil.copy(file_path, path4)

def read_bbox_and_words(paths, rootdir):
  bbox_and_words_list = []
  for path in paths:
    with open(rootdir+path, 'r', errors='ignore') as f:
      for line in f.read().splitlines():
        if len(line) == 0:
          continue

        split_lines = line.split(",")

        bbox = np.array(split_lines[0:8],dtype=np.int32)
        text = ",".join(split_lines[8:])

        # From the splited line we save (filename, [bounding box points], text line).
        # The filename will be useful in the future
        bbox_and_words_list.append([path[:-4], *bbox, text])

  dataframe = pd.DataFrame(bbox_and_words_list, columns=['filename', 'x0', 'y0', 'x1',
  dataframe = dataframe.drop(columns=['x1', 'y1', 'x3', 'y3'])

  return dataframe


#create .gt.txt
df = read_bbox_and_words(paths = txt_file,rootdir = "./SROIE2019/DeduplicatedTask1/")
max_len = -1000
for line in df["line"]:
  if len(line)>max_len:
    max_len = len(line)
print(f"max_length: {max_len}")
df
```

```
df['line'] = df['line'].astype(str) + str('/t')
df['line']
```

```
    0                                  TAN WOON YANN/t
    1              BOOK TA .K(TAMAN DAYA) SDN BND/t
    2                                        789417-W/t
    3            NO.53 55,57 & 59, JALAN SAGU 18,/t
    4                                      TAMAN DAYA,/t
                                 ...
    33621                                      TOTAL/t
    33622                                      11.32/t
    33623                                       0.68/t
    33624                                  THANK YOU/t
    33625    FOR ANY ENQUIRY, PLEASE CONTACT US:/t
    Name: line, Length: 33626, dtype: object
```

```
one_line_df = df[["filename","line"]].groupby(["filename"])["line"].transform(lambda :
one_line_df
#print(one_line_df.shape[0])
```

```
    0        TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SDN BND...
    1        TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SDN BND...
    2        TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SDN BND...
    3        TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SDN BND...
    4        TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SDN BND...
                                    ...
    33621    3180303/t LIAN HING STATIONERY SDN BHD/t (1627...
    33622    3180303/t LIAN HING STATIONERY SDN BHD/t (1627...
    33623    3180303/t LIAN HING STATIONERY SDN BHD/t (1627...
    33624    3180303/t LIAN HING STATIONERY SDN BHD/t (1627...
    33625    3180303/t LIAN HING STATIONERY SDN BHD/t (1627...
    Name: line, Length: 33626, dtype: object
```

```
one_line_df = one_line_df.to_frame().merge(df, left_index=True, right_index=True)[["l:
one_line_df = one_line_df.drop_duplicates()
```

```
for i in range(one_line_df.shape[0]):
  row = one_line_df.iloc[i,:]
  text = row[0]
  filename = row[1]

  text_file = open("/content/drive/MyDrive/ST456-Final-Project/tesstrain/data_temp/"+

  #write string to file
  text_file.write(text)

  #close file
  text_file.close()


#move .gt.txt to intended folder
folder1 = '/content/drive/MyDrive/ST456-Final-Project/tesstrain/data_temp'
folder2= '/content/drive/MyDrive/ST456-Final-Project/tesstrain/data/MODEL_NAME-ground

texts_f = [f for f in os.listdir(folder1)]

for file in texts_f:
  file_path = os.path.join(folder1,file)

  if file_path.endswith('.gt.txt')==True:
    shutil.copy(file_path, folder2)
```

TRAINING TESSERACT MODEL

```
os.chdir('/content/drive/MyDrive/ST456-Final-Project')
```

```
#get in the github repository
os.getcwd()
```

```
cd tesstrain/
```

```
    /content/drive/MyDrive/ST456-Final-Project/tesstrain
```

```
!ls
```

```
    data                            generate_wordstr_box.py  plot
    data_temp                       LICENSE                  README.md
    generate_gt_from_box.py         Makefile                 requirements.txt
    generate_line_box.py            normalize.py             shuffle.py
    generate_line_syllable_box.py   ocrd-testset.zip         src
```

```
#create data folder
!make training MODEL_NAME=MODEL_NAME
```

```
    find -L data/MODEL_NAME-ground-truth -name '*.gt.txt' | xargs paste -s > "data/M
    unicharset_extractor --output_unicharset "data/MODEL_NAME/unicharset" --norm_mod
    Bad box coordinates in boxfile string! TAN WOON YANN/t BOOK TA .K(TAMAN DAYA) SD
    Extracting unicharset from plain text file data/MODEL_NAME/all-gt
    Other case a of A is not in unicharset
    Other case n of N is not in unicharset
    Other case w of W is not in unicharset
    Other case o of O is not in unicharset
    Other case y of Y is not in unicharset
    Other case b of B is not in unicharset
    Other case k of K is not in unicharset
    Other case m of M is not in unicharset
    Other case d of D is not in unicharset
    Other case s of S is not in unicharset
    Other case j of J is not in unicharset
    Other case g of G is not in unicharset
    Other case u of U is not in unicharset
    Other case h of H is not in unicharset
    Other case c of C is not in unicharset
    Other case e of E is not in unicharset
    Other case p of P is not in unicharset
    Other case i of I is not in unicharset
    Other case q of Q is not in unicharset
    Other case f of F is not in unicharset
    Other case x of X is not in unicharset
    Other case v of V is not in unicharset
    Other case z of Z is not in unicharset
    Wrote unicharset file data/MODEL_NAME/unicharset
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469612.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469619.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469620.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469622.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469623.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
    PYTHONIOENCODING=utf-8 python3 generate_line_box.py -i "data/MODEL_NAME-ground-t
    + tesseract data/MODEL_NAME-ground-truth/X00016469669.tif data/MODEL_NAME-ground
    Tesseract Open Source OCR Engine v4.0.0-beta.1 with Leptonica
    Page 1
    Warning. Invalid resolution 0 dpi. Using 70 instead.
```

```
list = os.listdir('/content/drive/MyDrive/ST456-Final-Project/tesstrain/MODEL_NAME-gr
number_files = len(list)
print(number_files)
```

```
    1252
```

```
folder2= '/content/drive/MyDrive/ST456-Final-Project/tesstrain/data_temp/'

texts_f = [f for f in os.listdir(folder2)]

for file in texts_f:
  #print(file)
  file_path = os.path.join(folder2,file)
  #print(file_path)
  if file_path.endswith('.tif')==True:
    continue
  else:
    os.remove(os.path.join(folder2, file))
```

```
!pip install -r requirements.txt
```

```
    Requirement already satisfied: Pillow>=6.2.1 in /usr/local/lib/python3.7/dist-pa
    Collecting python-bidi>=0.4
      Downloading python_bidi-0.4.2-py2.py3-none-any.whl (30 kB)
    Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/
    Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dis
    Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-pac
    Installing collected packages: python-bidi
    Successfully installed python-bidi-0.4.2
```

```
!apt-get install bc
```

```
    Reading package lists... Done
    Building dependency tree
    Reading state information... Done
    The following NEW packages will be installed:
      bc
    0 upgraded, 1 newly installed, 0 to remove and 41 not upgraded.
```

```
Need to get 86.2 kB of archives.
After this operation, 223 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 bc amd64 1.07.1-2 [86.2
Fetched 86.2 kB in 0s (1,639 kB/s)
Selecting previously unselected package bc.
(Reading database ... 155561 files and directories currently installed.)
Preparing to unpack .../archives/bc_1.07.1-2_amd64.deb ...
Unpacking bc (1.07.1-2) ...
Setting up bc (1.07.1-2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

```
!make training MODEL_NAME=MODEL_NAME
```

```
!make clean MODEL_NAME=MODEL_NAME
```

```
find -L data/MODEL_NAME-ground-truth -name '*.box' -delete
find -L data/MODEL_NAME-ground-truth -name '*.lstmf' -delete
rm -rf data/MODEL_NAME
```

We realize Task 1 would be difficult to do on its own, as a lot of pretrained model will combine Task 1 and 2, namely Text Localisation and Text Recognition. Due to the difficulty of the task, we decide to combine the objective of Task 1 and Task 2 together.

Task 2 - TEXT RECOGNITION

> Since we are combinging Task 1 and Task 2, the main input of this model is the bounding boxes which gives the relative position of the line text.

> Traditional approach uses a CNN backbone with a LSTM model to recognize the image patterns to predict the texts. In this section the base model TrOCR is implemented and fine-tuned to our dataset SROIE.

> This model uses a transformer architecture, having a encoder-decoder structure which the encoder's weights are initialized with VeiT type models, and the decoder's weights are initialized with BERT type models. More specifically, the model chosen to be initialized in our implementations are BEiTBASE for the encoder and RoBERTaBASE for the decoder. Ref: TrOCR Paper

[ ] ↳ *43 cells hidden*

TASK 3 - KEY INFORMATION EXTRACTION

> Using the output of Task 2 - Text Recognition, the aim of this task is to extract texts of a number of key fields from given receipts, and save the texts for each receipt image in a JSON file.

> For each test receipt image, the extracted text is compared to the ground truth. An extract text is marked as correct if both submitted content and category of the extracted text matches the ground truth. An F1 score is used for ranking.

> To complete this task we will be using a model called LayoutLM. The LayoutLM model was proposed in the paper "LayoutLM: Pre-training of Text and Layout for Document Image Understanding" by Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou.

> LayoutLM is a simple but effective multi-modal pre-training method of text, layout and image for visually-rich document understanding and information extraction tasks, such as receipt understanding.

Link to paper: https://arxiv.org/pdf/1912.13318.pdf

The model that we will be using is taken from a github repository:https://github.com/microsoft/unilm.git

Link to Kaggle Notebook that part of the code is based on: https://www.kaggle.com/code/urbikn/layoutlm-using-the-sroie-dataset/notebook

[ ] ↳ *49 cells hidden*

Conclusion

The presented project applies OCR machine to localise and extract key information from invoices. The data was collected from an open-source platform, where newly developped programming tools were available to the wide network of developers to work on improving OCR techniques. Using these resoures enabled the performance of the three tasks at hand, dicussed above.

> Task 1 deals with localising the text using bbox coordinates, tesseract was used to detection the text in the image and draw boxes around them.

> Task 2 deals with recognising the text given the cropped images using the bboxes, TrOCR was used to recognize the text. (F1:testing=84%)

> Task 3 deals with labelling the extracted text with either company name, address, date, or total amount, the model used was LayoutLM. (F1:testing=95%)

Executing these tasks brought our project to a successful end, where by evaluating, subsequently, the outputs of our models we could come to this conclusion.

We hope that readers can utilize this notebook as an template to build on and eventually create a complete data pipeline to use on any printed document, allowing them to store their physical documents in a more accessible form.

Limitations and Future Work

For future research, we will further explore the models architecture as well as the pre-training strategies. This project limitations were identified respectively to each task, ultimately responding to our belief: conducting tasks individually will improve the accuracy of our model.

> Task 1: Localising and extrating coordinates using Tesseract-OCR engine showed limitations when it came to images affected by artifacts, but more specifically is not capable of recognising handwriting. Although we did not require the extraction of such font, this could represent our algorithm limited to other data type. However, a future work could be to train our tesseract model on custom data to improve the accuracy of this particular task as well as training it for various fonts and languages. Furthermore, pre-processing jpg files to remove noise would avoid any boxes being created intending to detect texts. As a result, it increased considerably the number of coordinates stored compared to the amount of text within a file.

> Task 2: The limitations we found during this task were that we have only implemented one model, more model could be easily implemented by substituing the model name in Model Configuration section in Task 2. For the TrOCR model, we have trained the model for 3 epochs, and it took 5 hours using Google Colab's GPU. The performance of the model can be further improved if the epochs number increases. Readers are welcomed to perform further training without the time constraint of our project.

> Task 3: A limitation of using LayoutLM is that the model relies more on manual labelling of images and does not fully explore the possibility of using large-scale unlabeled training samples. For future work with LayoutLM we will investigate different versions of the model that expand the language detected to make the LayoutLM model available for different languages, mainly the non-English regions across the globe.

Bibliography

- S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development," in Proceedings of the IEEE, vol. 80, no. 7, pp. 1029-1058, July 1992, doi: 10.1109/5.156468.
- Gregory Vial, Understanding digital transformation: A review and a research agenda, The Journal of Strategic Information Systems, Volume 28, Issue 2, 2019, Pages 118-144, ISSN 0963-8687
- Patel, Chirag & Patel, Atul & Patel, Dharmendra. (2012). Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. International Journal of Computer Applications. 55. 50-56. 10.5120/8794-2784.
- T. C. Wei, U. U. Sheikh and A. A. -H. A. Rahman, "Improved optical character recognition with deep neural network," 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), 2018, pp. 245-249, doi: 10.1109/CSPA.2018.8368720.

Contribution

> Candidate Number: 27416 - Contribution = 33.33%

> Candidate Number: 36805 - Contribution = 33.33%

> Candidate Number: 26089 - Contribution = 33.33%